Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

António José Pinto dos Santos

dezembro| 2016

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO

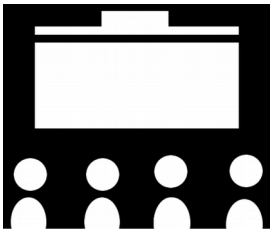
ANTÓNIO JOSÉ PINTO DOS SANTOS

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Dezembro / 2016

Gesp.010.03



Área curricular: Projeto de Informática

Ano letivo: 2015 / 2016

Gestão e controlo de recursos / materiais escolares

Orientador: Prof. José Alberto Quitério Figueiredo

Autor do documento: António José Pinto dos Santos

Resumo

Este documento representa o trabalho de investigação e implementação de um programa informático, um documento criado no âmbito da unidade curricular de Projeto de Informática, do curso de Engenharia Informática no Instituto Politécnico da Guarda. Um documento que descreve as etapas de projeção de uma aplicação, através de linguagem de modelação, com suas formas de identificação de requisitos e modelação de tarefas e dados, juntando a descrição e a identificação de testes.

A gestão de recursos escolares como salas de aula, projetores e computadores, entre outros, foi o tema escolhido para a constituição / desenvolvimento deste documento. Um tema escolhido porque o registo de entregas e levantamentos (sobretudo chaves de sala de aula) é, na Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda, ainda feito, com o recurso a fichas de papel. Os funcionários não possuem aplicações informáticas próprias que os auxiliem nas tarefas de registo e confirmação de empréstimos, tudo dependente de anotações ou outros auxiliares tradicionais de memória.

Para preencher aquela lacuna, idealizou-se uma aplicação que pudesse servir de apoio à atividade dos funcionários do IPG, uma aplicação de cariz tradicional, para ambiente em computadores pessoais, desenhada com diversas funcionalidades como registar utilizadores e materiais / recursos, efetuar reservas, confirmar e alterar reservas, automatizar processos de alocação de recursos com horários de trabalho (sobretudo, horários de professores e salas de aula) e mecanismos de consulta e pesquisa. Uma aplicação desenvolvida em Java, fazendo uso dos recursos existentes no ambiente de desenvolvimento JDK, na sua versão mais recente (à data), a versão 8. Uma aplicação distribuída pela sua interface, pela sua base de dados e documentos de auxílio à construção, gravação e formulação de dados.

Abstract

This document represents the work of a research and implementation of a computer program, a document created within the curricular unit of Computer Science Project in the course of Computer Engineering at the Polytechnic Institute of Guarda. A document that describes the steps of projecting an application, through a modeling language, with its forms of identification of requirements and modeling of tasks and data, joining the description and identification of tests.

The management of school resources such as classrooms, projectors and computers, among others, was the theme chosen for the constitution / development of this document. A chosen subject because the registration of deliveries and withdrawals (mainly classroom keys) is, at the School of Technology and Management of the Polytechnic Institute in Guarda, still done with the use of paper chips. Employees do not have their own computer applications to assist them with loan registration and confirmation tasks, all dependent on notes or other traditional memory helpers.

In order to fill this gap, an application was designed that could serve as a support for IPG employees' activity, a traditional application for a personal computers environment, designed with several functionalities such as registering users and materials / resources, make reservations, confirm and alter / replace reservations, automate resource allocation processes with work schedules (mostly, teachers schedules and classrooms), and search and query mechanisms. An application developed in Java, making use of the resources existing in the development environment JDK, in its latest version (to date), version 8. An application distributed by its interface, its database and documents to aid in the construction, recording and data formulation.

Índice

Índice de ilustrações

1. Introdução

O relatório aqui apresentado é fruto do projeto realizado pelo aluno António José Pinto dos Santos, no âmbito da unidade curricular de Projeto de Informática, componente curricular pertencente ao terceiro ano do curso de Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

1.1. Contextualização

Com o intuito de desenvolver o projeto final do curso de Engenharia Informática, a solução a desenvolver deverá compreender conteúdos obtidos durante os períodos de aprendizagem. Conceitos como programação orientada a objetos, base de dados, manipulação e tratamento de documentos devem existir num trabalho global e abrangente como este. Uma ferramenta de características semelhantes a um conjunto de aplicações cujo objetivo é a gestão de equipamentos enquadra-se naqueles moldes.

Atualmente, todo o registo de entregas das chaves de salas de aula é concretizado em papel, o funcionário da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda não possui uma ferramenta informática que o auxilie naquela tarefa, que lhe preste informação sobre a ocupação efetiva e a distribuição das atividades letivas, requisição ou troca de salas e consulta. Para além da falta existente, todo o processo de registo de entregas e empréstimos de material não está visível ao funcionário, não existe uma ferramenta próxima que lhe permita a gestão e manutenção de recursos. Assim, surgiu a ideia de criar uma aplicação informática para registo de empréstimos de recursos escolares e gestão / ocupação de salas de aula.

Com uma aplicação para organização e controlo de recursos, o funcionário passa a possuir informação complementar para monitorização / consulta de pessoas e das suas atividades. O funcionário tem, assim, correta informação sobre ocorrências escolares, localização daquelas e horários destinados.

Para facilitar o processo de criação de empréstimos, pretende-se que o processo de criação de empréstimos / requisições de salas de aula seja, em alguns casos, automático, o software pensado tem de ser capaz de associar recursos (salas de aula) com pessoas (neste caso, professores) consoante o horário daquelas. Desta forma caberá ao funcionário, apenas, o papel de validar a requisição / ocupação efetiva dos recursos, gerir recursos disponíveis (salas de aula) para alterações de horário e trocas imprevistas. O programa terá capacidade para conseguir a informação (horários de professores) necessária na plataforma escolar e criar registos na base de dados consoante aqueles, tendo em atenção as datas a gerar, consoante os períodos (semestres) letivos e os intervalos de interrupção.

Outro dos objetivos imaginados para o programa passa pela capacidade de prover informação acerca da calendarização das atividades letivas, associando pessoas, disciplinas e recursos. Com esta ideia o funcionário passa a ter informação onde ocorrerão determinadas atividades, quando ocorrerão e o período

destinado a cada. Esta condição vai permitir ao funcionário aumentar a sua capacidade de resposta em situações de consulta ou pedidos de informação e de elaboração de tarefas de manutenção e limpeza.

A aplicação possuirá todas as funcionalidades indispensáveis a um software de gestão de materiais como inserção, edição e catalogação de recursos, registo de pessoas e empréstimos, marcação de devoluções, criação de relatórios de atividades e elaboração do histórico. Com este último aspeto é ambição desejada a elaboração de uma componente estatística, componente que pode estabelecer relação entre a qualidade dos recursos (salas de aula) e a progressão das atividades letivas, quantificando a relação entre pessoas e recursos, podendo servir de base para a criação de horários ao evidenciar tendências e gostos e até identificar recursos menos usados e que necessitam de condições diferentes para uma agradável experiência letiva.

A aplicação deverá ser simples, uma das condições / objetivos deste projeto passa por não exigir demasiado tempo de aprendizagem ou tempo de controlo de aplicação uma vez que o funcionário deve estar liberto para outro tipo de atividades. A interface deve ser intuitiva, os recursos listados como objetos responsivos que permitam, rapidamente e eficazmente, obter informação detalhada do tipo de recurso, características e calendarização de atividades. Poderá, ainda, fornecer mecanismos de filtro e pesquisa, segundo características ligadas a cada recurso, facilitando o processo de requisição.

Algum desenvolvimento deste projeto foi também elaborado nos tempos didáticos da disciplina de Programação Avançada, pelo que uma boa parte do código utilizado foi reaproveitado para este atividade. Por isso, e como o público alvo tenderá a fazer uso da aplicação no seu local de trabalho em computadores próprios, não se achou necessário elaborar uma aplicação Web.

2. Estado da arte

De modo a definir o conjunto de funcionalidades e objetivos, pretende-se com este capítulo realizar a análise de aplicações que emprestem ideias e conceitos ao projeto. A Escola Superior de Tecnologia e Gestão da Guarda possui, já, um sistema informático que permite realizar reservas de recursos. Esta funcionalidade da plataforma Web do Instituto Politécnico da Guarda não permite elaborar relatórios e estatísticas de uso e não permite selecionar ou realizar pesquisas por traços distintivos de cada recurso, é uma aplicação simples cujo objetivo serve, apenas, o intuito de registo de empréstimos.

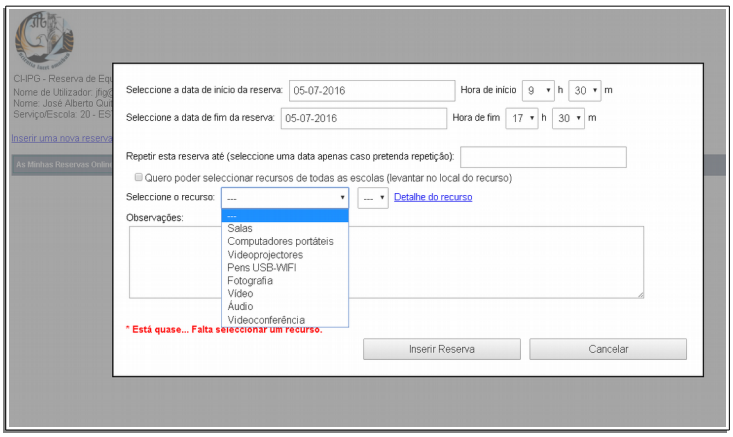


Ilustração 1: Ferramenta de reservas do IPG.

Como a aplicação para este projeto se baseia numa experiência muito particular, na qual se pretende conjugar a organização e controlo das atividades letivas com a gestão de recursos, as aplicações que foram analisadas dividiram-se em dois grupos: software para controlo de recursos e software para gestão de tempos de trabalho.

2.1. Gestão de recursos (empréstimos).

Começou-se por procurar soluções especializadas em gestão de recursos, em espaços educativos e com implemento de empréstimos. A maior parte das soluções encontradas são caracterizadas pela organização de bibliotecas, softwares que implementam a inserção, edição e catalogação de recursos, gestão de utilizadores, ferramentas de pesquisa e controlo de empréstimos.

2.1.1. Insignia Software.

O primeiro dos sistemas, *Insignia Software*, é uma ferramenta completa de gestão escolar, os módulos que incorpora são os seguintes: gestão de alunos, gestão de recursos digitais, gestão de bibliotecas e gestão de recursos. O último daqueles módulos tem interesse para os objetivos definidos uma vez que implementa funcionalidades que permitem a manipulação, gestão e calendarização de empréstimos. Eis algumas das utilidades daqueles módulos: catalogação e inventariação de recursos, gestão de utilizadores, ferramentas de controlo e circulação, sistema de notificações, aplicações para dispositivos móveis, base de dados disponível na Web para implementação OPAC (*Online Public Access Catalog*) e acesso a outros sistemas públicos através do uso do protocolo Z39.50.

Insignia Software é comercializada com duas versões, a versão mais completa, Interprise, possibilita a gestão de livros escolares, a integração RFID, reserva de salas, gestão de ativos e comércio eletrónico.

2.1.2. ResourceMate.

O seguinte sistema, que captou interesse, chama-se *ResourceMate*. É um produto da empresa Jaywil Software Development Inc.[1], uma instituição focada no desenvolvimento e automatização de pequenas e médias bibliotecas. O programa tem várias versões, com distintos preços e focada no tipo e tamanho da instituição a que se destina.

É uma aplicação desenvolvida em vários idiomas, bastante completa com todas as opções necessárias para gestão de recursos e gestão de utilizadores. Os registos associados a cada utilizador permitem o controlo de ações e registo de histórico (reservas, levantamentos e entregas de recursos (sobretudo livros), pagamentos e dados pessoais). Existem, mesmo, módulos adicionais que permitem o desenvolvimento de espaços pessoais para controlo e acesso àqueles dados.

É facultado um sistema de pesquisa online, que pode ser ou não acedido através de autenticação. O sistema de pesquisa inclui funcionalidades como procura fonética.

Na área destinada aos recursos, existe informação muito completa com imagens e outras características multimédia associadas, descrição física, assuntos relacionados a cada livro ou recurso, histórico de circulação e/ou reservas. Através do uso de equipamentos próprios, o sistema permite o uso do código de barras como fonte de informação para o registo de características dos recursos e assuntos relativos.

As ferramentas de gestão permitem definir os recursos disponibilizados para a pesquisa (catálogo online), fixar períodos de entrega por cada recurso ou grupos de recursos, estabelecer custos associados a atrasos e marcar assuntos/disciplinas relevantes por cada livro ou material.

Vários relatórios podem ser obtidos com sumários de reservas por utilizadores, grupos de utilizadores e tipo de recursos. Existem, ainda, módulos adicionais que permitem adicionar funções de promoção dos espaços e dos produtos.

2.1.3. CyBrosys School Library.

O sistema avaliado pertence à empresa Cybrosys Technologies, que é uma empresa indiana dedicada à produção de soluções informáticas. O sistema que desenvolvem para bibliotecas escolares surge com o intuito de centralizar os processos de empréstimo. É de fácil uso e navegação, com capacidade de gerir subscrições, catálogos e divulgação de relatórios. Possui as funcionalidades exigidas a um sistema deste tipo: registo de livros e utilizadores; pesquisa interativa do tipo OPAC; controlo de circulação; ambiente multi-utilizador; importação, backup e restauro de dados, criação e/ou edição de relatórios e informação estatística através de documentos em formato Excel, HTML ou OpenOffice.[2]

1 Ver: www.resourcemate.com.
2 Ver: www.cybrosys.com/School-library.

A segurança é um atributo associado ao sistema, este permite a definição de privilégios e gestão de grupos de utilizadores; permite ter um controlo dos movimentos realizados pelos administradores, existindo funcionalidades de deteção de ações inseguras e bloqueio de utilizadores.

2.1.4. Koha.

É um software de gestão integrada de bibliotecas. Um sistema Web, com código aberto e com uma comunidade de entusiastas e contribuidores bastante grande, o seu sucesso deve-se há inexistência de custos associados à licença com que é distribuído e às grandes possibilidades de expansão. Segundo a Wikipédia, foi criado em 1999 pela empresa Katipo Communications sediada na Nova Zelândia[3], está traduzido em mais de 100 línguas e está, hoje, implementado em mais de 900 bibliotecas públicas de todo o mundo.

Como os sistemas anteriores, a plataforma Web possui módulos que permitem registos de recursos e utilizadores, a criação de catálogos e pesquisa online (através do protocolo Z39.50 os seus catálogos fazem parte de uma rede global de pesquisa), módulos de gestão de recursos em circulação, módulos de gestão de utilizadores, sistema de compras e/ou aquisições, ferramentas de administração e relatórios detalhados de utilização.

Bastante configurável, o sistema permite gerir o ambiente de trabalho e perfis de utilizadores, definir regras e períodos de empréstimo, obter dados estatísticos para gestão de orçamentos, configurar métodos de pesquisa e unir catálogos.

Existem aplicações móveis relacionadas com o sistema Koha: uma simples pesquisa nas lojas de cada plataforma móvel permitiu encontrar, *Kiritaki Koha* (Android) que possibilita ao utilizador a pesquisa e a implementação de reservas, *my library koha em Android e Koha 3 Library Management System em* Iphone permitem a administração de bibliotecas através de um aparelho móvel.

2.1.5. Biblivre.

É um software livre para automatização de bibliotecas. Criado no ano de 2005, é um programa patrocinado pelo Instituto Itaú Cultural do Brasil e está instalado em cerca de 6000 bibliotecas daquele país e em 10 bibliotecas portuguesas. Possui versões em inglês, português e espanhol. As atualizações são permanentes e gratuitas. É compatível com Windows, Linux e outros sistemas Unix. Este sistema está desenvolvido em Java, utiliza o PostgreSql como gestor de base de dados e o Apache Tomcat como servidor Web para aplicações Java. [4]

Este sistema dá ênfase às rotinas e aos principais procedimentos de uma biblioteca. A sua área de administração é bastante completa, dividindo-se em quatro zonas de utilização: aquisição, como a definição e o registo de fornecedores, o estado dos pedidos e requisições de objetos; configuração, através da funções que permitem, por exemplo, configurações de páginas, incluir traduções, definir tipos de utilizadores, configurar

3 Artigo referenciado: https://en.wikipedia.org/wiki/Koha_(software).
4 Ver: http://biblivre.hemominas.mg.gov.br/Biblivre4/static/Manual_Biblivre_4.1.0.pdf.

permissões e senhas; catalogação, através da listagem de material bibliográfico, multimédia e objetos digitais (programas de computador); e circulação, através da inscrição de utilizadores, cadastro, definição da tipologia de utilizadores, permissões de acesso ao cadastro, registo de empréstimos ou reservas e definição de multas.

O sistema permite o acesso aos catálogos mundiais através do protocolo Z39.50, com uma pesquisa vasta e diversificada por: nome ou título, assunto, ano de publicação, autor, atributos, entre outros. Para além da possibilidade da reserva, o sistema permite a impressão de obras e textos distribuídos sob domínio público.

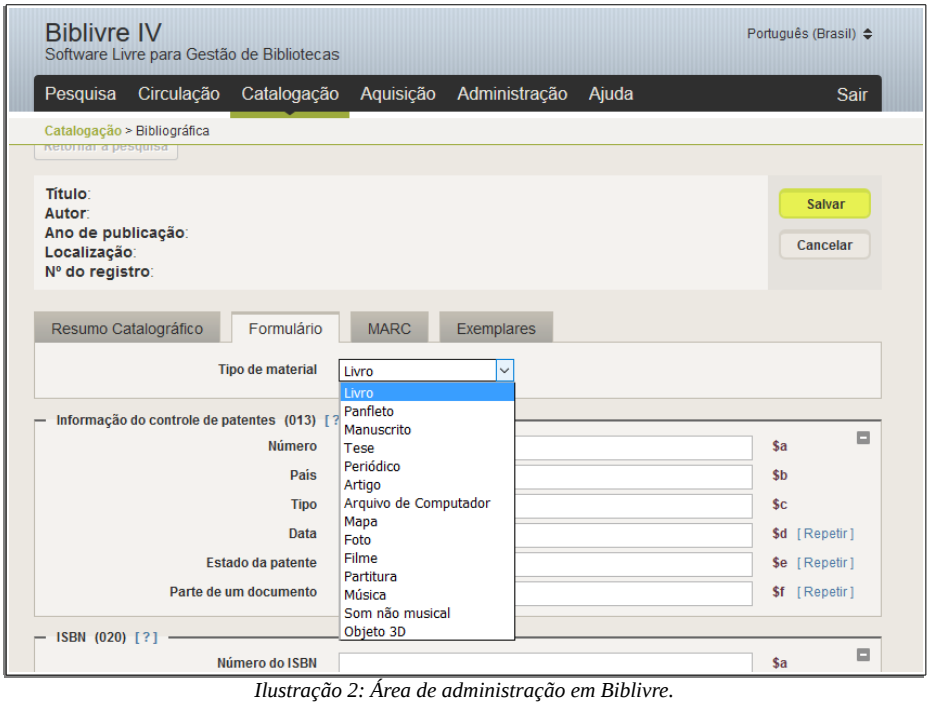


Ilustração 2: Área de administração em Biblivre.

2.2. Gestão de tempos de trabalho.

Através da relação entre salas de aula / recursos e horários escolares, o sistema pode (ou poderia) incorporar funções para controlo de tempos de trabalho e registo / consulta de períodos de atividade. Com vista àquele objetivo, as análises realizadas a seguir foram efetuadas sobre programas que permitem o controlo de momentos de trabalho e a gestão de recursos humanos nos seus períodos laborais.

2.2.1. DRoster.

É uma aplicação destinada à gestão de trabalhadores, atribuição de horários e controlo dos tempos de entrada e saída. É propriedade da empresa Kappix, fundada em 2004.[5] Surgiu com a ideia de cumprir vários requisitos para distintos tipos de empresas e diferentes atividades económicas. Hoje em dia, várias empresas como restaurantes, hotéis, centros médicos, instituições académicas, escolas, empresas de construção e até instituições militares usam esta solução.

A aplicação é comercializada em duas versões, uma versão *freeware* e uma versão paga.

5 Mais informações em: http://www.kappix.com/contact.htm

De entre as funcionalidades que fornece conta-se o planeamento de horários, associando empregados aos períodos laborais. A aplicação implementa todas as funções de inserção / edição de trabalhadores, criação de turnos, configuração detalhada das atividades e preparação / edição de relatórios. O número de relatórios disponíveis depende da versão do programa usada, que podem ser imprimidos ou exportados para formato Excel, CSV, Word e HTML.

Os empregados podem ser inseridos em grupos, esses grupos podem ser criados e editados. Os horários podem ser fixos, alternativos ou rotativos. A aplicação permite controlar a disponibilidade de cada utilizador em período parcial ou total, controlar e ter acesso a períodos de férias ou a ausências por motivo de doença. Possui um sistema de controlo na atribuição de horários que impede a sobreposição daqueles, impedindo, da mesma forma, a atribuição de horários a empregados ausentes ou se ultrapassam os tempos máximos diários de trabalho.

Para maior colaboração com os empregados, a aplicação disponibiliza funções de notificação através do correio eletrónico de cada trabalhador.

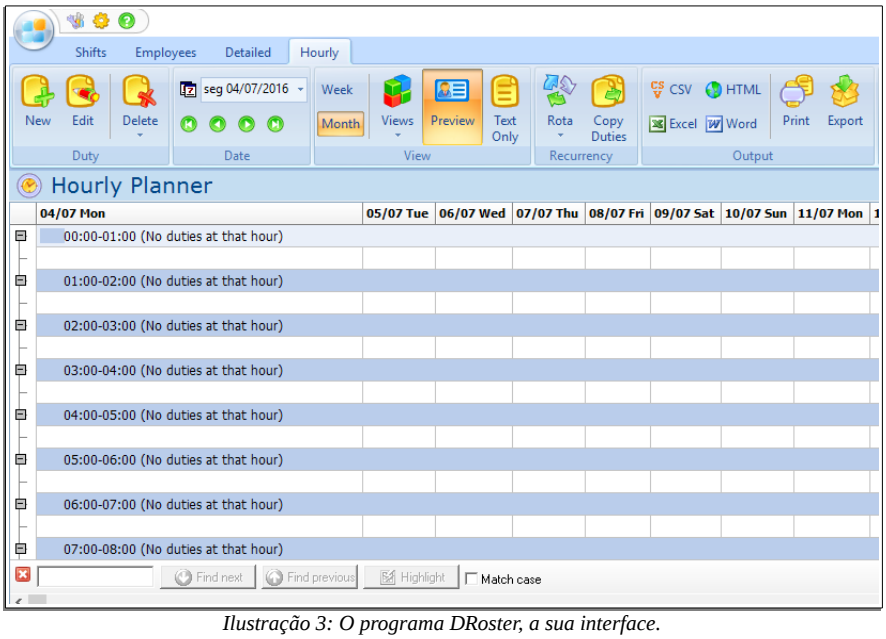


Ilustração 3: O programa DRoster, a sua interface.

2.2.2. Open Time Clock.

É um serviço Web que não exige instalação, pode ser usado em qualquer sistema ou plataforma e pode ser estendido ou acedido através de aplicações móveis para Windows Phone, Android e Iphone. Da mesma forma que o programa anterior, este permite o registo de empregados, atividades e grupos, tempos de trabalho e períodos de interrupção ou ausência. No entanto, esta é uma aplicação distinta; a aplicação anterior focava-se na organização e atribuição de horários, esta foca a sua utilidade no controlo de entradas e saídas.

Existe uma interface que apenas se destina à administração, serve a inserção, a edição e configuração de todos os dados da aplicação, é o centro de registo das atividades e dados relativos aos empregados.

Por cada empregado é criada uma conta com a qual o funcionário acede para marcar ou carimbar entradas e saídas em tempos trabalho. Como a aplicação gere os IPs e as coordenadas geográficas das máquinas de acesso do utilizador, só um número específico e identificado de máquinas pode servir aquela funcionalidade, garantindo confiança nos comportamentos e validade dos dados. Existem, também, módulos de reconhecimento de imagem e validação através de câmara.

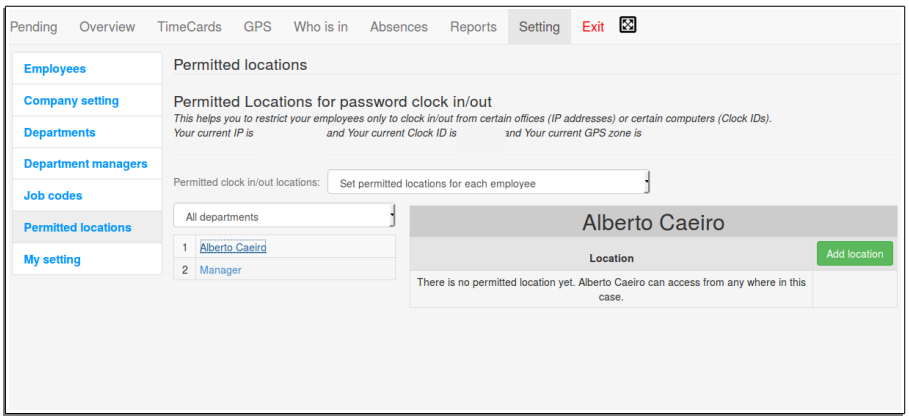


Ilustração 4: Página de definições em OpenTimeClock.

2.2.3. CKZ time clock.

É um sistema muito completo, implementa todas as funções dos programas anteriores às quais acrescenta algo mais. É distribuído em duas versões, uma versão para desktop e outra em plataforma Web. A versão para desktop está disponível sem encargos adicionais, as funcionalidades que acrescenta permitem monitorizar empregados em tempo real, controlo de assiduidade e horas extra, registo e controlo de períodos de férias e ausências por doença. É um sistema que permite um controlo do pagamento de salários, com relatórios completos por cada trabalhador.

Possui funções para criação e gestão de horários, tendo em atenção os tempos de entrada e saída dos empregados, assim como dos períodos de ausência daqueles. As atividades podem ser reajustadas em tempo real e a performance dos trabalhadores avaliada através do sucesso de execução daquelas.

Sem limite para o número de empregados, esta aplicação incorpora aspetos e direitos exigidos por lei laboral: os tempos de trabalho extra, os subsídios de férias e as ausências por doença são situações ou campos configuráveis por cada diferente trabalhador ou grupo de trabalhadores.

Com a versão da plataforma Web, não há exigência de instalação, manutenção ou preocupação com atualizações; no entanto, o sistema não é gratuito. Como pode ser acedido de qualquer lugar, este permite que o empregado utilize a sua conta para identificar períodos ativos de trabalho, em espaço próprio ou página pessoal da plataforma, ao qual é associado um histórico de presenças e realização de tarefas que pode servir para avaliar a performance e servir como base de promoções.[6]

6 Ver: http://www.ckzinc.com.

2.3. Funcionalidades prestadas.

Apesar da identificação de alguns objetivos, a análise do estado da arte permitiu consolidar aspetos técnicos e funcionais que podem servir à construção de uma aplicação mais completa. Em baixo apresenta-se um quadro de funcionalidades que existem nos sistemas analisados e que servem os propósitos deste projeto.

Gestão de recursos	Registo de recursos.
	Registo de utilizadores.
	Associação do tipo de material a assuntos / caraterísticas.
	Pesquisa por tipo de material, assunto e características.
	Requisição de material.
	Registo de empréstimos e devoluções.
	Relatórios de utilização por material e tipo de material.
	Análise estatística de utilização por características físicas e técnicas do recurso.
Gestão de tempos de trabalho	Associação de horários laborais a datas e recursos (salas de aula).
	Marcação de presenças através de empréstimos (levantamentos) e devoluções.
	Possibilidade de alteração de horário e espaço de atividade.
	Edição e controlo de tempos de paragem (férias e feriados).
	Relatórios de atividade por utilizador.
	Monitorização e pesquisa de horários por docente e por atividade letiva.

3. Planeamento e desenho.

Neste capítulo pretende-se fazer a análise das principais metodologias de produção de software, caraterizar o desenvolvimento em processo ágil, pretende-se definir a metodologia adotada e selecionar os recursos, pretende-se delinear etapas e apresentar objetivos.

3.1. Metodologia.

O desenvolvimento ágil, "camaleónico", rege-se pelo princípio iterativo, onde *as pessoas têm um papel fundamental no desenvolvimento do projeto*[7], todos os intervenientes do processo (arquitetos, designers, executivos, gestores de projeto e produto, programadores, escritores técnicos, responsáveis de recursos humanos e utilizadores)[8] são chamados a cada iteração, contribuindo com capacidade de comunicação, compreensão e imaginação. A planificação é dinâmica, a opinião do cliente / utilizador (*feedback*) conta e a adaptação a novos requisitos é condição essencial.

Dentro do panorama global da tendência ágil de produção de software, são várias as vertentes que se ajustam consoante a dimensão dos grupos de trabalho, características dos projetos, escolhas, formação pessoal e de grupo. Para este projeto, porque a equipa de produção e desenvolvimento é de dimensões reduzidas (constituída por aluno e orientador), porque as etapas de desenvolvimento e escrita são curtas, para que as sessões / reuniões de avaliação, testes e planificação possam ser frequentes (semanalmente ou quinzenalmente) escolheu-se a metodologia XP (eXtreme Programming).

A metodologia XP busca abordar práticas, tais como, reparação, programação em par, mudança, feedback, desenvolvimento iterativo e testes automatizados, entre outras[9]. Os processos iterativos são compostos por quatro atividades: planeamento, projeto, codificação e testes. Para este projeto, pretende-se que nas fases de planeamento se possa discutir e documentar os requisitos funcionais por cada incremento ao projeto; que em fases de projeto ou design se documente o projeto com modelos da linguagem UML (Unified Modeling Language) e requisitos técnicos; que em fases de codificação se desenvolva a aplicação com o apoio e a aprovação técnica do orientador; e que as fases de teste sejam fator de evolução do projeto e controlo de tempo.

7 Mário Rui Sampaio Tomás – *Métodos ágeis: características, pontos fortes e fracos e possibilidades de aplicação*, Faculdade de ciências e Tecnologia da Universidade Nova de Lisboa, Lisboa, 2009, página 5.

8 Ver: www.desenvolvimentoagil.com.br/xp.

9 Vinícius Manhães Teles – Um estudo de caso da adoção das práticas e valores Extreme Programming, dissertação de mestrado da Universidade Federal do Rio de Janeiro, Rio de Janeiro 2005, página 55.

3.2. Etapas.

Para a realização deste projeto foram reconhecidas as seguintes etapas de desenvolvimento, com os respetivos tempos de duração:

Análise de requisitos, 36 horas.

Estado da arte, 36 horas.

Estudo das tecnologias a utilizar, 24 horas.

Planeamento, 80 horas

Definição / desenho de estilos, 70 horas.

Codificação, 360 horas.

Testes e avaliação da aplicação (aperfeiçoamento), 72 horas.

Escrita do relatório. 96 horas.

Em baixo, apresenta-se o mapa de Gantt, com a distribuição de etapas e respetiva calendarização.

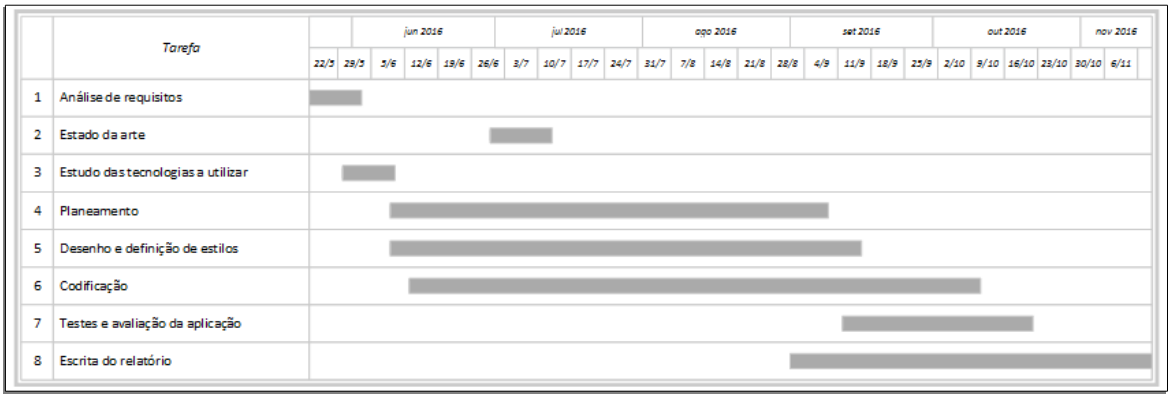


Ilustração 5: Mapa de Gantt.

3.3. Funcionalidades / Objetivos.

A aplicação deverá poder interagir com dois tipos de utilizadores, um administrador que deverá ter capacidade para criar utilizadores de tipo “Administrador” – nível elevado ou “Funcionário” - nível seguinte, tendo, por isso, possibilidade para gestão de permissões; e um utilizador do tipo funcionário que será o principal interveniente do sistema.

Utilizador – funcionário:

- Inserir e editar pessoas que realizam empréstimos.
- Inserir e editar recursos.
- Pesquisa de recurso por nome, assunto, características.
- Pesquisa de pessoa, com informação de atividade e histórico.

- Registar requisição.
- Confirmar empréstimo.
- Confirmar devolução.
- Atribuir nova requisição.
- Atribuir recurso distinto.
- Definir tempos letivos, períodos de férias e feriados.
- Visualizar estatísticas.
- Exportação / impressão de relatórios.

Utilizador – administrador:

- Todas as funcionalidades do utilizador – funcionário.
- Criação de contas de utilizador – funcionário e utilizador – administrador.
- Gestão de privilégios.

3.4. Desenho e modelação.

3.4.1. Diagrama de contexto.

Para compor os atributos descritos em cima, usaram-se práticas, diagramas e símbolos gráficos de modelação descritos na linguagem UML (Unified Modeling Language). É que, ao recorrer-se a símbolos padronizados, todo o projeto de Engenharia de Software se vê descrito por elementos comuns, facilitando a compreensão e a descrição dos objetos e a comunicação estabelecida entre eles.[10]

O primeiro desenho ou diagrama, que se descreve a seguir, pretende ser um elemento de análise às interações possíveis entre a aplicação e sistemas / indivíduos / elementos externos àquele. Um diagrama de contexto não se preocupa com a estrutura interna do sistema, o seu objetivo é, simplesmente, ligá-lo a fatores externos, de forma a identificar restrições e requisitos.[11]

Os objetivos descritos no capítulo anterior fornecem ações válidas àquelas interações, podendo ser descritos como fluxos de dados (entrada ou saída) que tornam a aplicação útil e funcional. É, por exemplo, requisito do sistema que o utilizador “funcionário” possa registar e editar dados de utilizadores “requerentes”, inserir e editar dados de recursos, marcar e substituir requisições, confirmar entregas e devoluções de material.

É, por exemplo, requisito do sistema que o utilizador “administrador” possa registar utilizadores com privilégios elevados, por forma a que o utilizador “funcionário” possa interagir com a aplicação. E, para isso, é necessário ou requisito que o utilizador “administrador” possa gerir privilégios. É, também, requisito que a

10 Em Wikipédia: https://pt.wikipedia.org/wiki/UML, última atualização a 16/08/2016.
11 Ver: https://es.wikipedia.org/wiki/Diagrama_de_contexto_de_sistema, última atualização a 21/02/2016.

aplicação apresente recursos de pesquisa e automatismos capazes de gerar requisições através do tratamento e decomposição do documento CSV.

É, ainda, requisito que a aplicação forneça mecanismos de apoio à análise e à avaliação dos dados registados, combinando-se àqueles, elementos de exportação e impressão.

Com isso, apresenta-se, a seguir, o diagrama de contexto onde se descrevem os requisitos enunciados, com a indicação da direção da comunicação, movimentos que mostram os fluxos de dados (entrada e saída) entre a aplicação e atores externos.

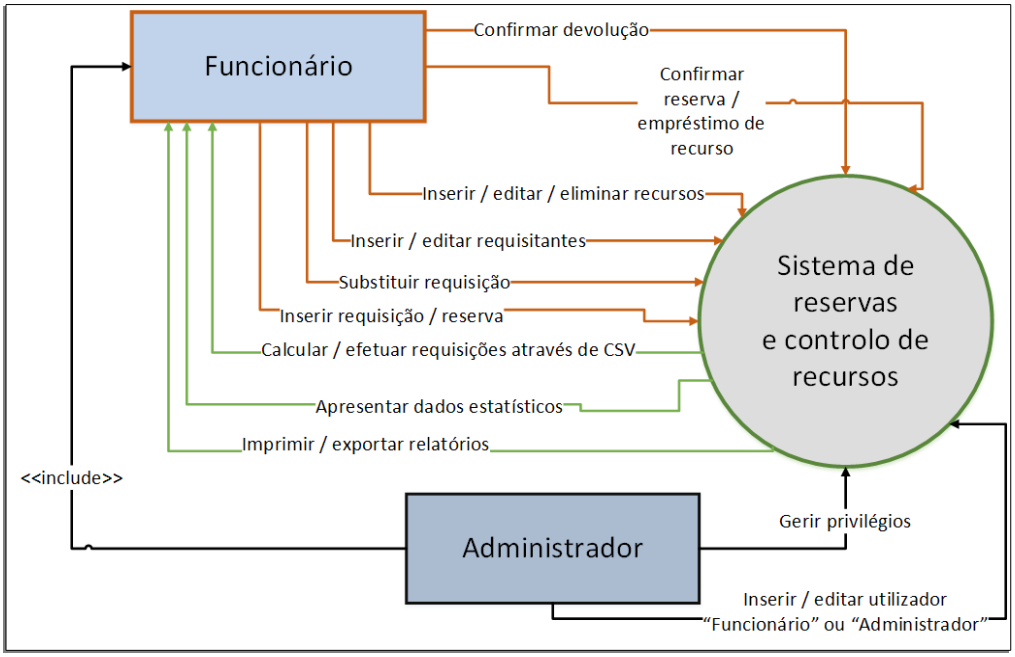


Ilustração 6: Diagrama de contexto.

3.4.2. Diagrama de casos de uso.

A linguagem UML distingue dois grupos de diagramas: diagramas de estrutura e diagramas de componente comportamental. O diagrama de casos de uso faz parte do último conjunto. O seu objetivo passa por identificar os principais momentos de interação com os utilizadores, casos funcionais, que derivam dos requisitos gerais identificados no diagrama anterior.

O diagrama de casos de uso é composto essencialmente por quatro partes: cenário ou palco da ação (normalmente, a aplicação), o ator (podendo ser um utilizador ou um qualquer sistema externo ao cenário), o caso de uso e o processo de comunicação.

Para esta aplicação, são dois os atores que podemos identificar: o administrador e o funcionário. Os casos de uso do administrador são os mesmos aplicados ao utilizador "funcionário", mas acresce-lhe a possibilidade de gerir contas e privilégios quando fazendo uso dos casos: "Inserir utilizador" e "Editar utilizador".

Alguns casos permitem "endereçar" o utilizador para diferentes casos de uso, nesta situação o processo de comunicação estende-se até àquele. É exemplo o caso de uso: "Editar recurso" que permite

eliminar o registo, é, também, exemplo o caso de uso “Visualizar atividades / estatística por recurso ou utilizador” que permite criar relatórios para exportação ou impressão.

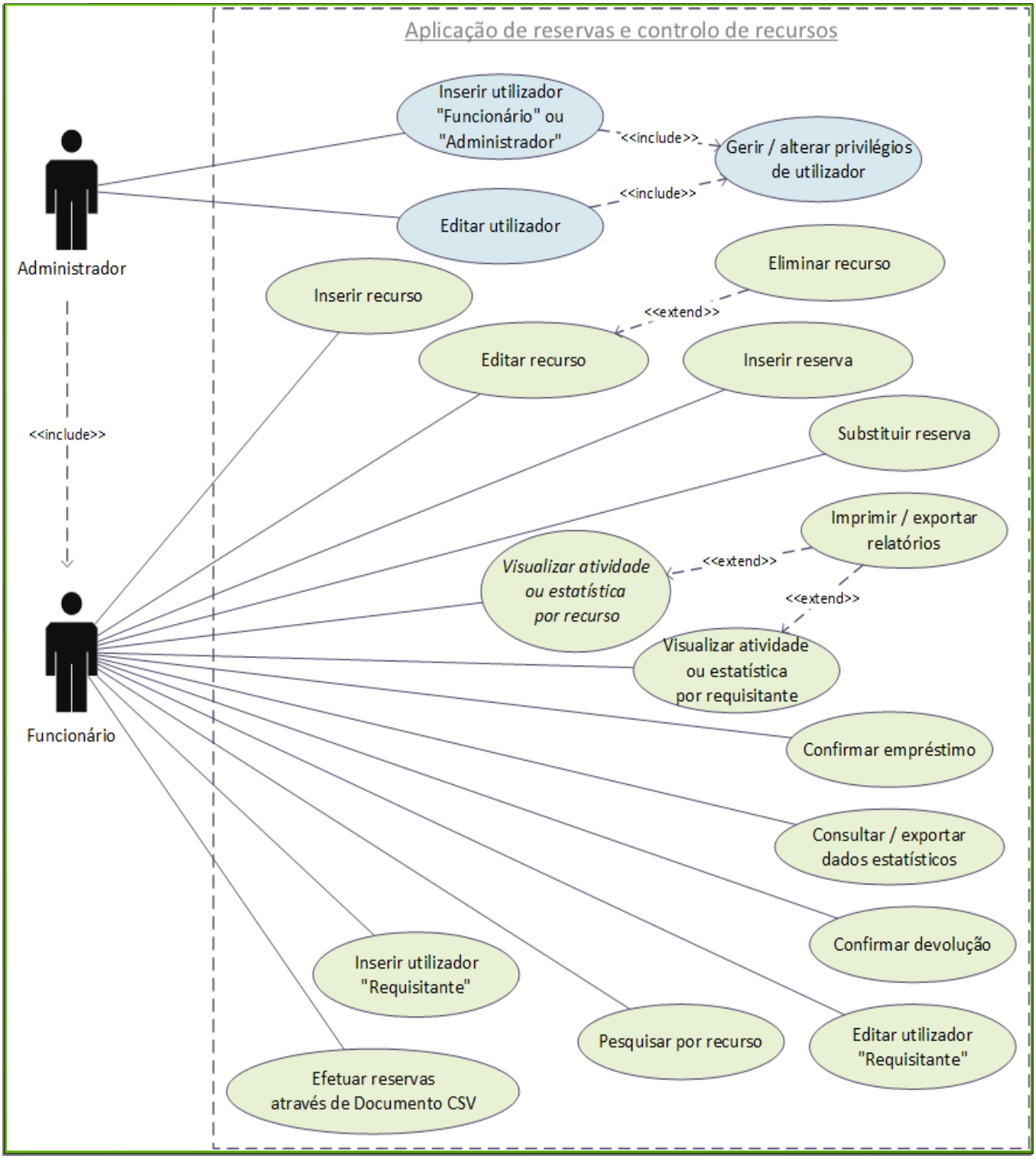


Ilustração 7: Diagrama de casos de uso.

Neste exemplo, podemos identificar os casos de uso mais relevantes (mais frequentes) como sendo: inserir recurso, editar recurso, inserir utilizador “requisitante”, inserir utilizador “administrador ou funcionário”, inserir reserva, confirmar empréstimo, confirmar reserva e substituir reserva.

Aqueles são os momentos principais da aplicação e que definem, neste caso, o processo de criação de classes. Classes, que à medida que vão sendo descritas, se tornam em objetos caraterizados. Para compor o diagrama de classes é necessário descrever os casos de uso, identificar os nomes que se convertem em identidades, os adjetivos que se transformam em elementos de classe e os verbos que distinguem momentos de comunicação entre utilizadores, interfaces e objetos.

O capítulo seguinte (descrição de casos de uso e diagramas de sequência) irá permitir identificar não só as tabelas essenciais à base de dados, mas, também, as interfaces necessárias para que o sistema possa interagir com os utilizadores.

3.4.3. Descrição de casos de uso / diagramas de sequência.

A descrição de casos de uso, para além do já mencionado, permite achar caminhos, cenários alternativos e momentos de exceção. A descrição sequencial possibilita a descoberta de pré-condições ou requisitos anteriores à ocorrência, permite, ainda, a descoberta de sucessos que podem ocorrer após o término do caso de uso. Por isso, existe a exigência da procura e da definição de respostas do sistema para diferentes estados, exigindo, também, o estabelecimento de prioridades de desenvolvimento e testes que acompanham o momento de codificação.

Os quadros a seguir, que descrevem os casos de uso referidos no capitulo anterior, estão divididos em dez secções: nome, objetivo, atores envolvidos, pré-condição, prioridade (desenvolvimento), fluxo principal, fluxos alternativos, fluxos de exceção, pós-condição e casos de teste.

3.4.3.1. Inserir recurso.

O titulo demonstra, claramente, o objetivo deste caso de uso. A primeira das interfaces que podemos fichar é “Inserir recurso”, a partir dali toda a interação do utilizador com o sistema permite desenvolver este caso de uso. Podem, já, distinguir-se algumas classes como: materiais, tipos de material e caraterísticas; e, na classe materiais, podem, já, adivinhar-se os seguintes atributos: designação, imagem, código ou identificação.

Nome	Inserir recurso
Objetivo	Registar novo recurso na base de dados.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Alta
Fluxo principal	**1.** O ator seleciona a opção “Inserir recurso” no menu principal da aplicação. **2.** O sistema apresenta nova janela onde o utilizador pode selecionar e introduzir: tipo de material, designação, imagem, código do produto, caraterísticas do produto. (2A) **3.** O ator seleciona o tipo de material. **4.** O sistema mostra caraterísticas de acordo com o tipo escolhido. **5.** O ator preenche os campos obrigatórios e, opcionalmente, escolhe a imagem, regista / escolhe caraterísticas do produto. **6.** Os campos designação e código do produto estão preenchidos, o sistema ativa a opção “Registar novo artigo”.

	7. O ator confirma o novo registo. **8.** O sistema apresenta a mensagem: “O produto foi registado”. (8A)
Fluxos alternativos	**8A** a) O sistema verifica a existência de um produto com o mesmo código, cancela a operação e apresenta a mensagem com a informação.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema apresenta mensagem do erro ocorrido: “Não existe ligação à base de dados”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar se as caraterísticas são alteradas de acordo com o tipo de material. **2.** Verificar se a opção “Registar novo artigo” só está ativa quando os campos obrigatórios estão preenchidos. **3.** Verificar se o sistema impede a inserção de produtos com o mesmo código .

Seguindo a análise anterior, podem, já, desenhar-se alguns recursos gráficos necessários à interface: desde campos para a seleção ou listagem de elementos, campos para entrada de dados, recursos para gestão e seleção de ficheiros, tabelas e listas, botões para confirmação ou anulação, janelas de mensagens que servem propósitos de comunicação entre utilizador e sistema.

O diagrama de sequência, que resulta da descrição do caso de uso, permite identificar outro objeto ou classe: medidas, uma tabela relacional entre caraterísticas e materiais que permitirá, ainda, acrescentar valor mensurável àquela relação. De notar que o sistema não deverá permitir a existência de produtos com o mesmo código na base de dados, os testes a implementar deverão tomar em conta esta situação.

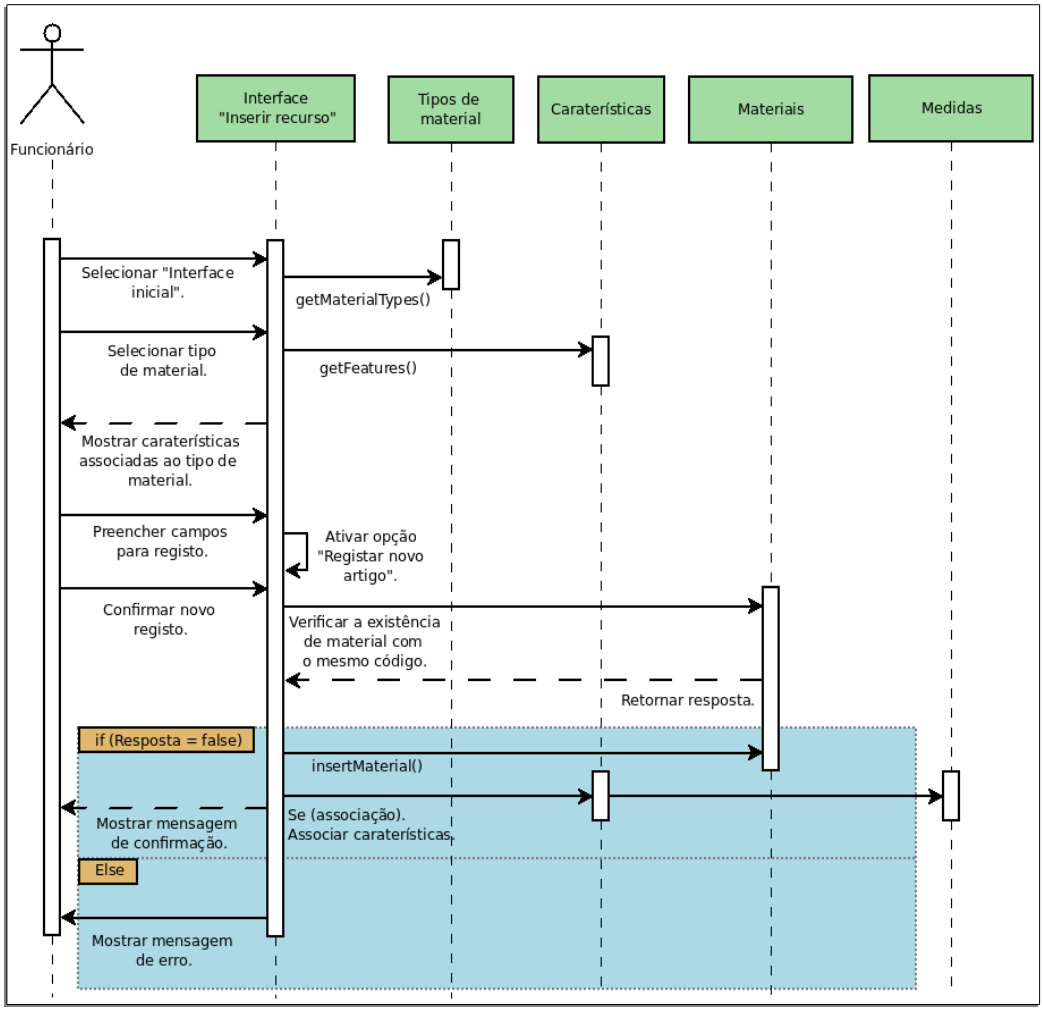


Ilustração 8: Diagrama de sequência: "Inserir recurso".

3.4.3.2. Editar recurso.

A descrição deste caso de uso permite distinguir mais três interfaces: “Ver recursos”, “Detalhes do recurso” e “Outros dados”. A primeira interface será uma janela de transição, o caso de uso deverá processa-se no plano seguinte, na interface “Ver detalhes”, onde o utilizador atuará com o componente “editar recurso” e o sistema reagirá permitindo a edição dos campos de texto. Esta interface irá permitir a passagem para outra zona ou janela onde o utilizador poderá associar caraterísticas e software (caso o recurso seja do tipo que o possua).

Nome	Editar recurso
Objetivo	Alterar os dados dos recurso.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Média
Fluxo principal	**1.** O ator dirige-se à opção “Ver recursos” do menu da janela principal. **2.** O sistema apresenta a lista de recursos por cada tipo de material. (2A) **3.** O ator seleciona o recurso pretendido. **4.** O sistema apresenta nova janela com detalhes do recurso e opções para edição. **4.** O ator seleciona a opção “Editar recurso”. (4A) (4B) **5.** O sistema ativa os campos com os dados do recurso, permitindo a sua edição. **6.** O ator edita os novos dados e confirma na opção “Gravar”. **7.** O sistema atualiza os dados do recurso.
Fluxos alternativos	**4A** a) O ator escolhe a opção “Outros dados”. b) O sistema abre nova janela onde o utilizador pode associar e remover associação de caraterísticas, software (recursos com software) ao recurso. c) O ator seleciona associação,remove ou insere uma nova. d) O sistema guarda as alterações.
	4B **a)** O ator escolhe a opção eliminar. (4BaA) **b)** O sistema apresenta mensagem de aviso: “Pretende eliminar recurso, esta ação não permite restauro”. **c)** O ator confirma a opção. (4BcA) **d)** O sistema encerra janela com detalhes do recurso e opções para edição.
	4BaA **a)** O sistema verifica que o recurso se encontra emprestado ou que existem requisições futuras associadas. A opção eliminar não está ativa.
	4BcA **a)** O ator seleciona a opção cancelar. **b)** O sistema cancela a operação.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema não permite a abertura da janela com a lista de recursos.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar se os novos valores são apresentados ou modificados na janela que lista todos os recursos e se a ordenação é realizada corretamente. **2.** Verificar se a possibilidade de encontrar o recurso por nome, caraterística e software existe.

Nesta descrição de caso de uso, existe um outro momento: “eliminar recurso”, achou-se que este cenário não representava um caso de uso independente e, por isso, é representado como fluxo alternativo.

O diagrama de sequência, a seguir, distingue mais duas classes para incorporar no diagrama respetivo, são eles: software e instalações (tabela relacional entre software e materiais, que acrescenta campos como a data da instalação, data de atualização ou estado da atualização).

Embora não esteja definido no diagrama de sequência em baixo, os campos de edição deverão estar formatados com elementos de validação, impedindo valores incorretos ou vazios nos campos obrigatórios. Como a interface “Ver recursos” é uma interface, sobretudo, informativa, pensa-se utilizá-la na listagem de recursos, introduzindo ferramentas de pesquisa e acrescentando campos informativos sobre reservas futuras dos materiais selecionados.

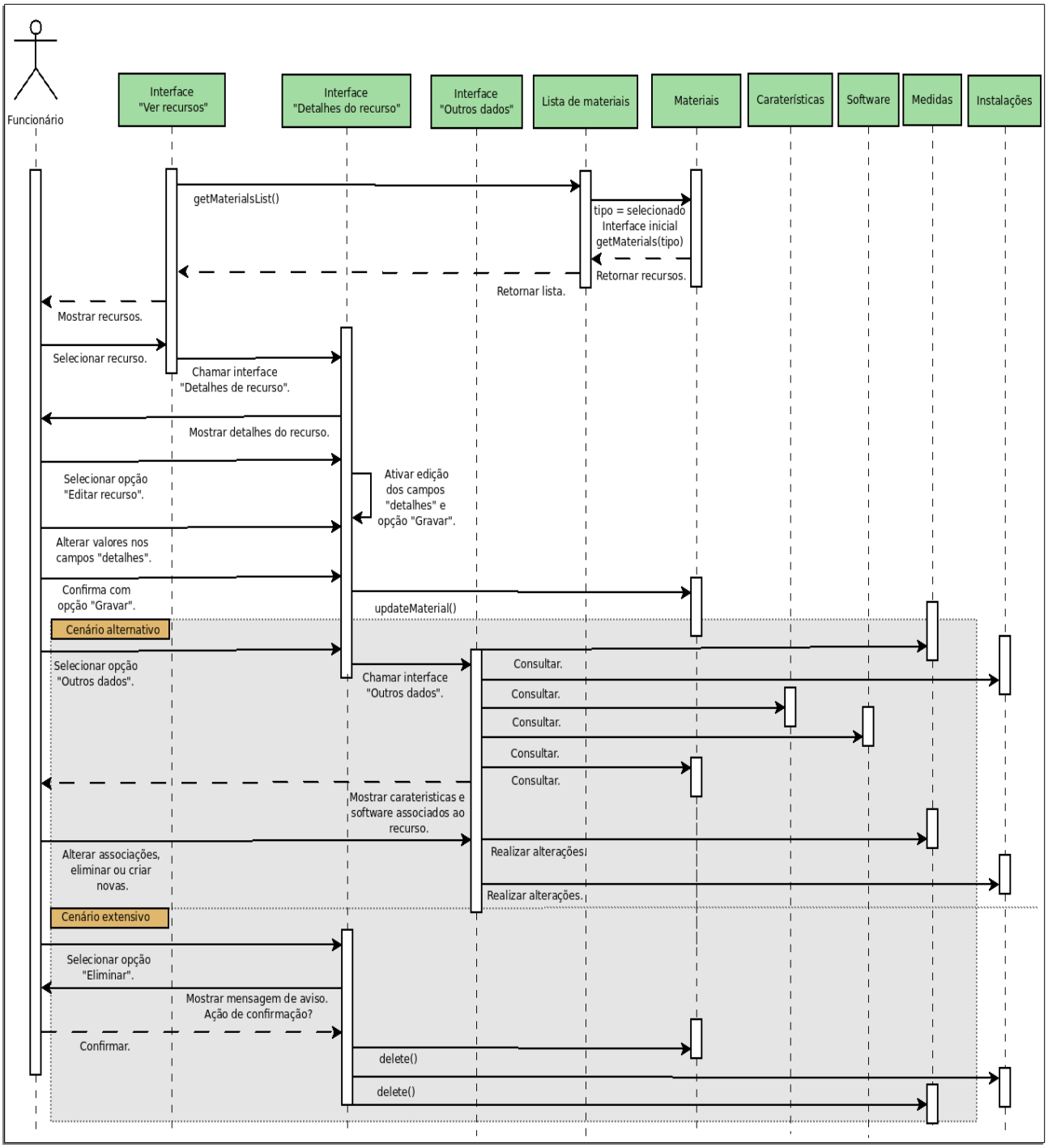


Ilustração 9: Diagrama de sequência: "Editar recurso".

3.4.3.3. Inserir utilizador “requisitante”.

Na descrição deste caso de uso identifica-se uma outra interface, “Utilizadores”. É uma interface que permitirá adicionar novos utilizadores, mas, também, a edição dos já existentes. Para isso, a interface possuirá uma lista de utilizadores. Quando um dos valores daquela lista se encontrar selecionado, a interface possuirá as ferramentas necessárias para a edição dos dados de utilizador, caso contrário, a interface ajustar-se-á ao cenário deste caso de uso.

Podemos identificar mais duas classes: utilizadores e funções. Os campos: nome, correio eletrónico, telefone e identificação na instituição serão elementos da classe utilizadores.

Nome	Inserir utilizador
Objetivo	Registo da pessoa / requisitante no sistema.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Alta
Fluxo principal	**1.** O ator seleciona a opção “Utilizadores” no menu da aplicação. **2.** O sistema apresenta a lista de utilizadores já registados e um formulário para nova inscrição. No formulário estão referenciados os seguintes dados: nome, correio eletrónico, telefone, função, identificação na instituição. (2A) **3.** O ator preenche todos os campos ou apenas aqueles obrigatórios (nome, correio eletrónico ou identificação na instituição) e confirma-os. (3A) (3B) (3C) **4.** O sistema regista os novos valores no sistema e mostra mensagem de concretização do registo, o utilizador é adicionado à lista.
Fluxos alternativos	**3A** **a)** O ator não preenche os campos ‘nome’, ‘correio eletrónico’ e / ou ‘identificação na instituição’. **b)** O sistema mostra mensagem de erro: “Preencha os campos ‘nome’ e, pelo menos, um destes campos: ‘correio eletrónico’ ou ‘identificação’”.
	3B **a)** O ator preenche erradamente o campo “correio eletrónico” e “telefone”. O campo correio eletrónico deve corresponder à seguinte forma: “[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}$”, o campo telefone à seguinte: “^(([+]?(\\d{3}))|(([+]?[(](\\d{3})[)])))?[- .]?((2\\d{2}[. -]?\\d{6})|(2\\d{2}[. -]?\\d{3}[. -]?\\d{3})|(9[1236]\\d{7})|(9[1236][. -]?\\d{3}[. -]?\\d{4})|(9[1236][. -]?\\d{4}[. -]?\\d{3})|(9[1236][. -]?\\d{3}[. -]?\\d{2}[. -]?\\d{2}))”. O ator confirma os dados. **b)** O sistema mostra mensagem de erro: “Os campos ‘correio eletrónico’ e / ou ‘telefone’ estão num formato incorreto”.
	3C **a)** O ator preenche os campo com um endereço de correio eletrónico ou uma identificação já existente na base de dados. **b)** O sistema mostra mensagem de erro: “Verifique validade e exclusividade dos dados de identificação”.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema apresenta mensagem do erro ocorrido: “Não existe ligação à base de dados”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar a inclusão do novo utilizador na lista de utilizadores. **2.** Verificar se a omissão de campos obrigatórios e formatos errados impedem o registo de utilizadores.

Os campos, correio eletrónico e identificação, serão os elementos da classe que permitirão identificar o utilizador, devido à sua singularidade. Como os requerentes poderão ser externos à instituição, a introdução do correio eletrónico garantirá a identificação ou reconhecimento, dessa forma, pelo menos um daqueles campos não poderá estar vazio. O sistema ou aplicação deverá ser capaz de assegurar a caraterística daqueles campos (unicidade) e impedir a inserção de valores que a contradigam.

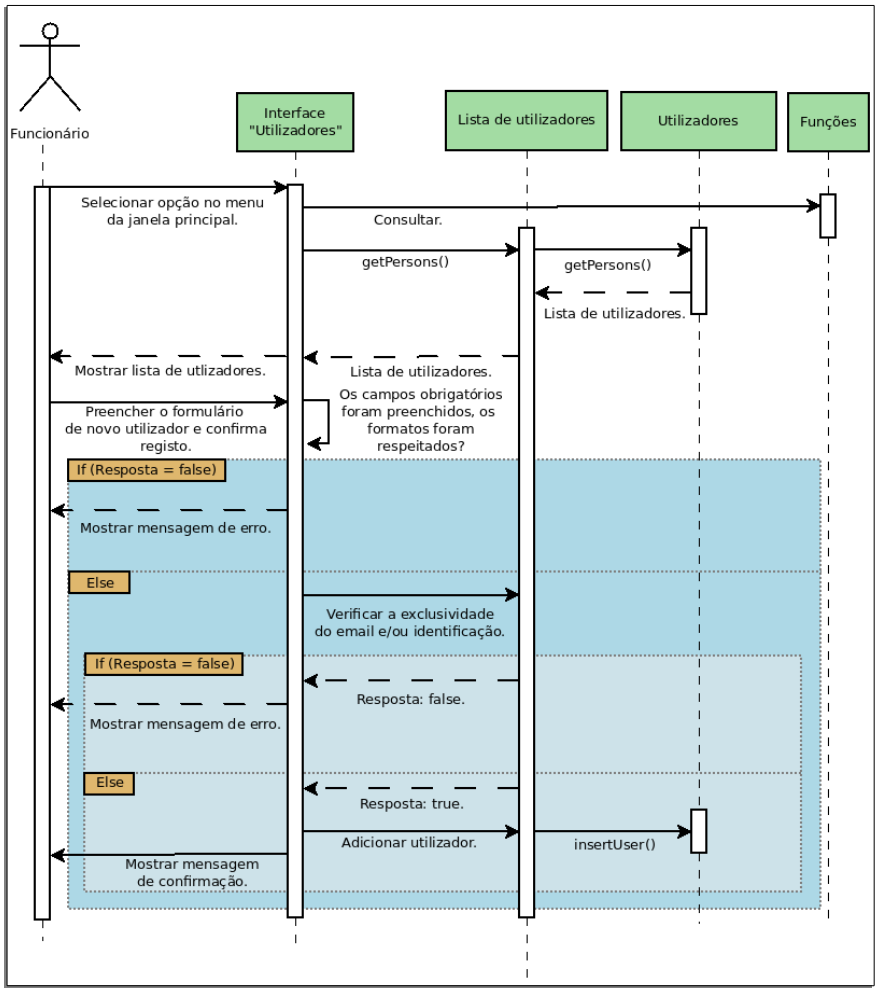


Ilustração 10: Diagrama de sequência: "Inserir utilizador".

3.4.3.4. Inserir utilizador "funcionário ou administrador".

Este caso de uso faz uso da interface anterior ("Utilizadores"), no entanto, sendo o sistema capaz de reconhecer e validar privilégios de utilização, permitirá que um utilizador do tipo administrador possa ter acesso a um conjunto de recursos para gestão de utilizadores e controlo dos seus direitos. É um caso de uso que se colocou à parte do anterior uma vez que define novos objetivos e objetos (classes). Como já dito, os próprios recursos colocados à disposição do utilizador são alargados.

As novas classes, agora achadas, são: privilégios e credenciais. Os tipos de privilégios identificados são: nível 1 (o mais elevado) usado na identificação de utilizadores do tipo "administrador", o nível 2 para identificar utilizadores do tipo "funcionário", o nível 3 para utilizadores com identificação / participação da instituição e o nível 4 para restantes utilizadores.

Os utilizadores dos dois primeiros níveis são os que apresentam relevância para este caso de uso, a sua criação originará o aparecimento de um novo objeto, credencial, onde serão registadas as referências de acesso à aplicação (palavra de acesso).

Nome	Inserir utilizador “Administrador” ou “Funcionário”
Objetivo	Inserir utilizadores com privilégios para execução e administração da aplicação.
Atores envolvidos	Administrador
Pré-condição	Login válido
Prioridade	Média
Fluxo principal	**1.** O ator executa o caso de uso “Inserir utilizador”. **2.** O sistema verifica a presença de um utilizador com privilégios elevados e para além das opções presentes no caso de uso “Inserir Utilizador”, exibe as opções “Editar privilégios” e “Senha de acesso”. (2A) **3.** O ator seleciona o privilégio de nível 2 para utilizador “Funcionário” ou de nível 1 para “Administrador”. Preenche o campo “Senha de acesso”. Confirma a opção. (3A) 4. O sistema regista a opção.
Fluxos alternativos	Todos os verificados para o caso de uso “Inserir utilizador”.
	3A a) O ator não preenche o campo “Senha de acesso”. b) O sistema impede o registo. O sistema Mostra mensagem: “Não preencheu o campo ‘Senha de acesso’ ”.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema apresenta mensagem do erro ocorrido: “Não existe ligação à base de dados”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar a a criação de utilizadores com aquele tipo de privilégios.

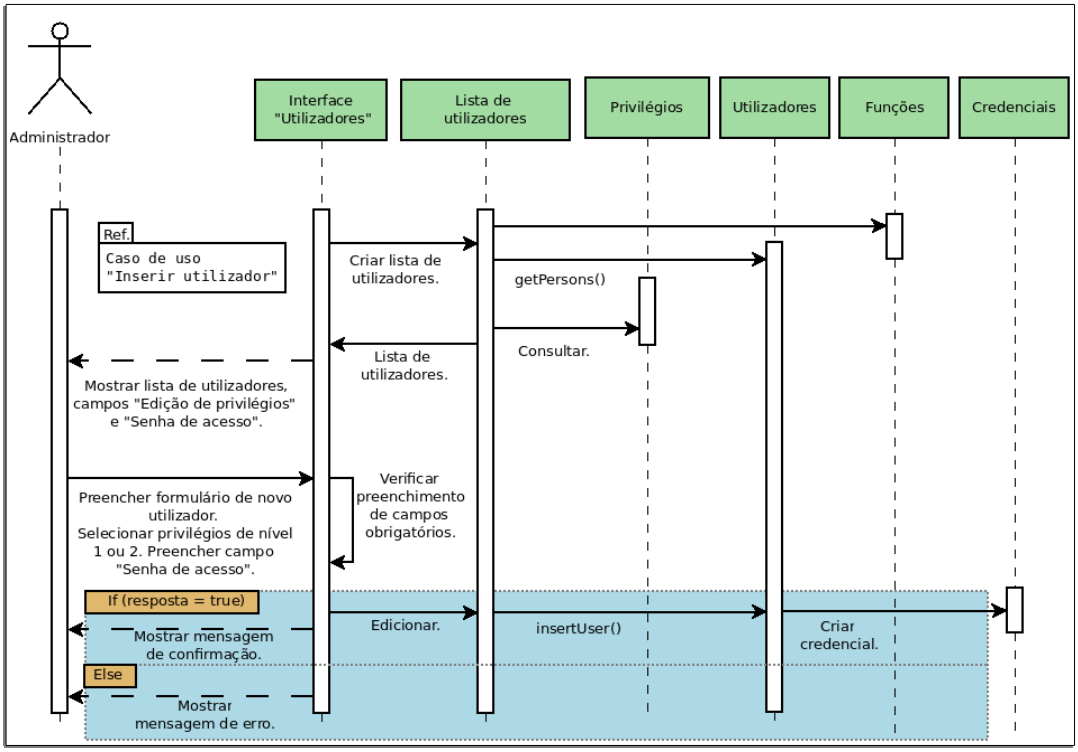


Ilustração 11: Diagrama de sequência “Inserir utilizador ‘Administrador’ ou ‘Funcionário’”.

3.4.3.5. Inserir reserva / requisição.

A classe requisição será a classe central da aplicação, com a descrição deste caso de uso, encontram-se mais três novas classes, são elas: turmas, atividades e disciplinas. Uma nova interface é, também, descrita: "Efetuar requisição".

A nova interface deverá ter opções para seleção de recursos e utilizadores, campos para escolha de datas e tempos de reserva, sendo o sistema capaz de atualizar a lista de recursos disponíveis por cada data ou hora escolhida. Nessa verificação deverá ter em conta a disponibilidade dos recursos, analisando requisições futuras e a sobreposição daquelas sobre os materiais selecionados.

Nome	Inserir reserva
Objetivo	Criar uma nova requisição.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Alta
Fluxo principal	**1.** O ator dirige-se à opção "Efetuar requisição" do menu da janela principal. **2.** O sistema apresenta uma janela com opções para seleção de recurso ou recursos, seleção de utilizador e tempos de reserva. (2A) **3.** O ator seleciona o tipo de recurso e o tempos associados à requisição. 4. O sistema verifica a disponibilidade dos recursos e apresenta as possibilidades. (4A) **5.** O utilizador seleciona o recurso ou recursos e preenche os campos destinados à escolha do utilizador. (5A) (5B) (5C) **6.** O sistema ativa a opção "Confirmar reserva(s)". **7.** O ator confirma a(s) reserva(s). **8.** O sistema apresenta a mensagem de confirmação. (8A)
Fluxos alternativos	O ator pode encerrar a janela a qualquer altura.
	4A **a)** O sistema não apresenta resultados, não existem recursos disponíveis para a ocasião. **b)** O ator escolhe outros momentos.
	5A **a)** O ator seleciona a opção que permite chamar o caso de uso "Pesquisar por recurso". **b)** O sistema atualiza a lista de recursos disponíveis com a pesquisa realizada.
	5B **a)** O ator seleciona a opção que permite chamar o caso de uso "Inserir utilizador". **b)** O sistema retorna novo utilizador.
	5C **a)** O ator seleciona a opção "Mais informação". **b)** O sistema apresenta nova janela onde o ator pode selecionar outra informação para agregar às informações da requisição (atividade, turmas e disciplinas). **c)** O ator seleciona a informação. (5CcA) **d)** O sistema guarda as opções.
	5CcA **a)** O ator cancela a operação. **b)** O sistema limpa qualquer associação existente.
	8A **a)** Um ou alguns dos recursos selecionados estão emprestados, embora a reserva

	tenha terminado. O sistema alerta o utilizador para a situação.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema mostra mensagem: “Não existe ligação à base de dados”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar se a requisição é efetuada com sucesso, testando requisições múltiplas. **2.** Verificar se a lista é atualizada de acordo com a disponibilidade de cada recurso. **3.** Verificar se a opção “Confirmar reserva” apenas está ativa quando as informações obrigatórias estão preenchidas (utilizador, recursos e balizas temporais). **4.** Verificar a validação dos tempos (tempo final não pode ser menor que tempo inicial). **5.** Verificar a correta associação de outras informações à requisição.

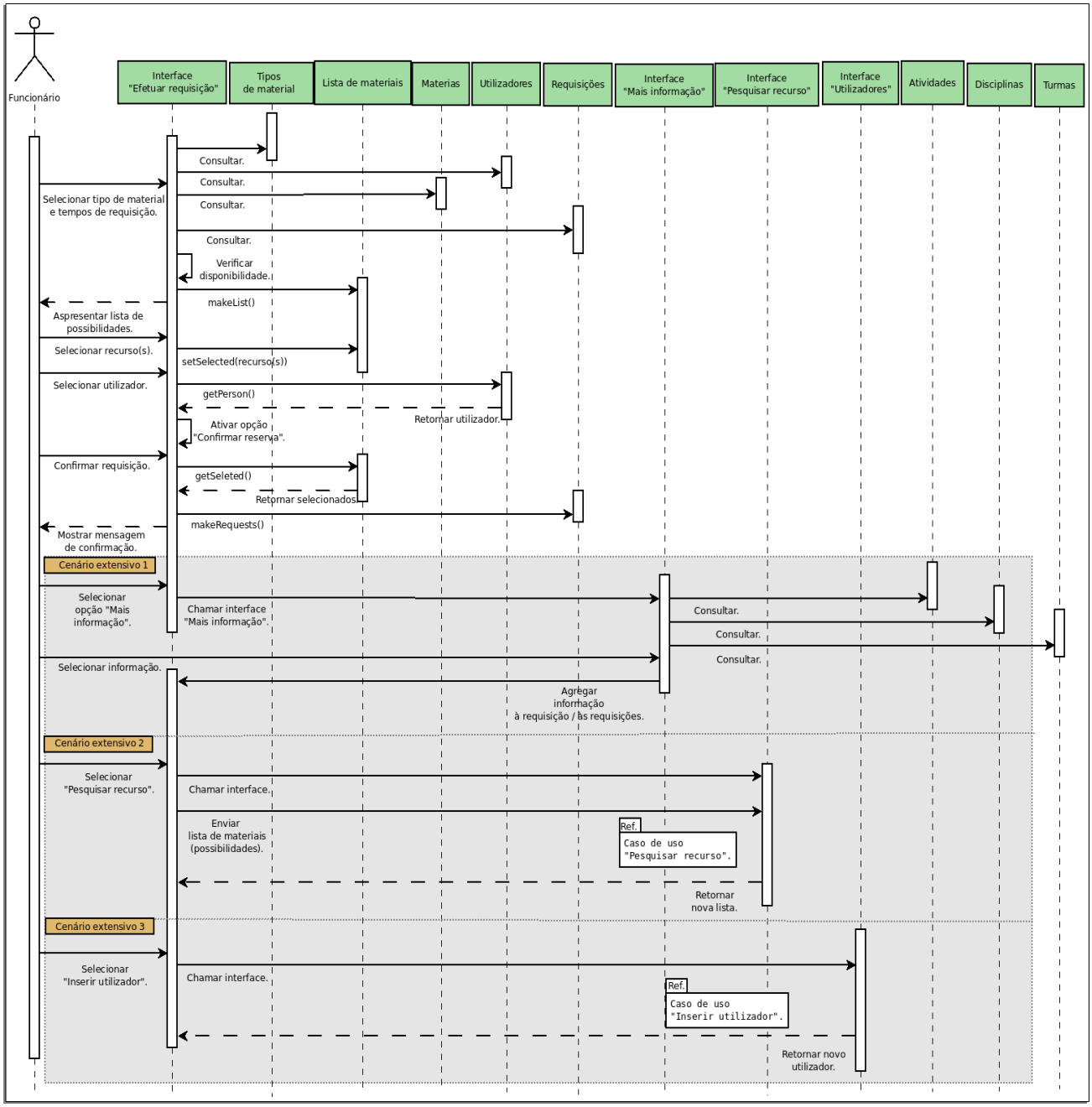


Ilustração 12: Diagrama de sequência: “Inserir reserva”.

Cada recurso reservado será uma nova requisição, com apenas uma atividade (exame, frequência, orientação tutorial, teórico-praticas, etc.) e um utilizador (poderá, no entanto, associar-se a várias turmas e disciplinas, condição que não será obrigatória). Por isso, para facilitar ou acelerar o processo de requisição, o sistema deverá permitir a seleção de múltiplos recursos, daí que, ao existir confirmação da ação / requisição, o sistema realizará uma reserva por cada recurso encontrado.

Este caso de uso pode implementar outros casos de uso, sendo esta situação descrita no diagrama de sequência atrás. O utilizador pode não estar registado, por isso, para agilizar o momento, a interface deverá possuir recursos para chamar o processo de inscrição de utilizador, e para a seleção de material conforme as caraterísticas desejadas, a interface deverá possuir recursos para invocar o caso de uso "Pesquisa por recurso" (em anexo), filtrando a lista através das escolhas da pesquisa.

Por fim, a associação da requisição com atividades, turmas e disciplinas, ocorrerá em nova interface: "Mais informação", numa associação que não será obrigatória, e por isso, é apresentado no diagrama como cenário extensivo.

3.4.3.6. Confirmar empréstimo.

A ideia original deste projeto teve como ponto orientador os dois seguintes casos de uso, pretendia-se, como disposto nos objetivos, fornecer ferramentas de auxílio aos funcionários no registo de empréstimos e na entrega do material. Estes serão os momentos para concretização daqueles objetivos e, por isso, ocorrem na janela inicial / principal da aplicação. Pretende-se, com isso, dar ênfase a estes momentos, já que serão as funcionalidades mais frequentes da aplicação.

É fundamental que sejam simples, concretizadas em poucos passos. O intuito passa por realizar estes momentos em dois simples passos (seleção e confirmação). A opção (botão) de confirmação, neste caso de uso, estará ativa, quando o tempo atual se encontrar nos períodos destinados à requisição (àquele intervalo é acrescentado tempo de tolerância: menos 30 minutos que o tempo de levantamento inscrito) e o recurso reservado se encontrar em situação de disponibilidade (estado "livre").

Quando a requisição for compreendida por tempos anteriores àqueles lidos pela aplicação, a opção (botão) de confirmação não estará ativa; acontecendo a mesma situação quando compreendida por tempos ultrapassados. Para alterar a requisição será necessário chamar o caso de uso "Substituir requisição".

Na descrição de caso de uso faz-se referência a reservas "passivas", serão as reservas tidas como não concretizadas, reservas às quais, ainda, não chegou o seu tempo. A confirmação do empréstimo implicará que a reserva passe à condição de "ativa" (nos diagramas de estados abordar-se-á esta situação com mais detalhe).

Nome	Confirmar empréstimo
Objetivo	Registar a entrega do recurso ao utilizador com marcação de data e hora da ocorrência.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Alta
Fluxo principal	**1.** O sistema apresenta uma tabela com as reservas "passivas" (ainda não

	concretizadas) e atualiza as entradas conforme a data e a hora atual. (1A) **2.** O ator seleciona a reserva que pretende confirmar. **3.** O sistema ativa a opção "Confirmar requisição" e apresenta os dados relativos à requisição num painel de informações. (3A) (3B) (3C) **4.** O ator confirma a entrega na opção. **5.** O sistema regista a entrega do recurso ao utilizador, incluindo data e hora da ocorrência e reformula a tabela de requisições passivas.
Fluxos alternativos	O ator pode cancelar / desfazer a seleção em qualquer instante.
	3A **a)** A reserva selecionada não foi concretizada no tempo (data e hora) atribuído, o sistema apenas ativa a opção "Substituir requisição". O sistema não ativa a opção "Confirmar requisição".
	3B **a)** O material requisitado está emprestado, normalmente, em situação de atraso na entrega do recurso. O sistema não ativa a opção "Confirmar requisição".
	3C **a)** O sistema não ativa a opção "Confirmar requisição", o tempo inicial (data e hora) da reserva ainda são futuros. O sistema apenas ativa a opção "Substituir requisição".
Fluxos de exceção	**1A** **a)** Não existe ligação à base de dados, o sistema apresenta mensagem do erro ocorrido.
Pós-condição	O sistema deve alterar o estado do recurso para ocupado. O sistema deve fazer nova contagem dos recursos livres e ocupados.
Casos de teste	**1.** Verificar se a requisição apresenta o estado ativo. **2.** Verificar se o recurso apresenta o estado ocupado.

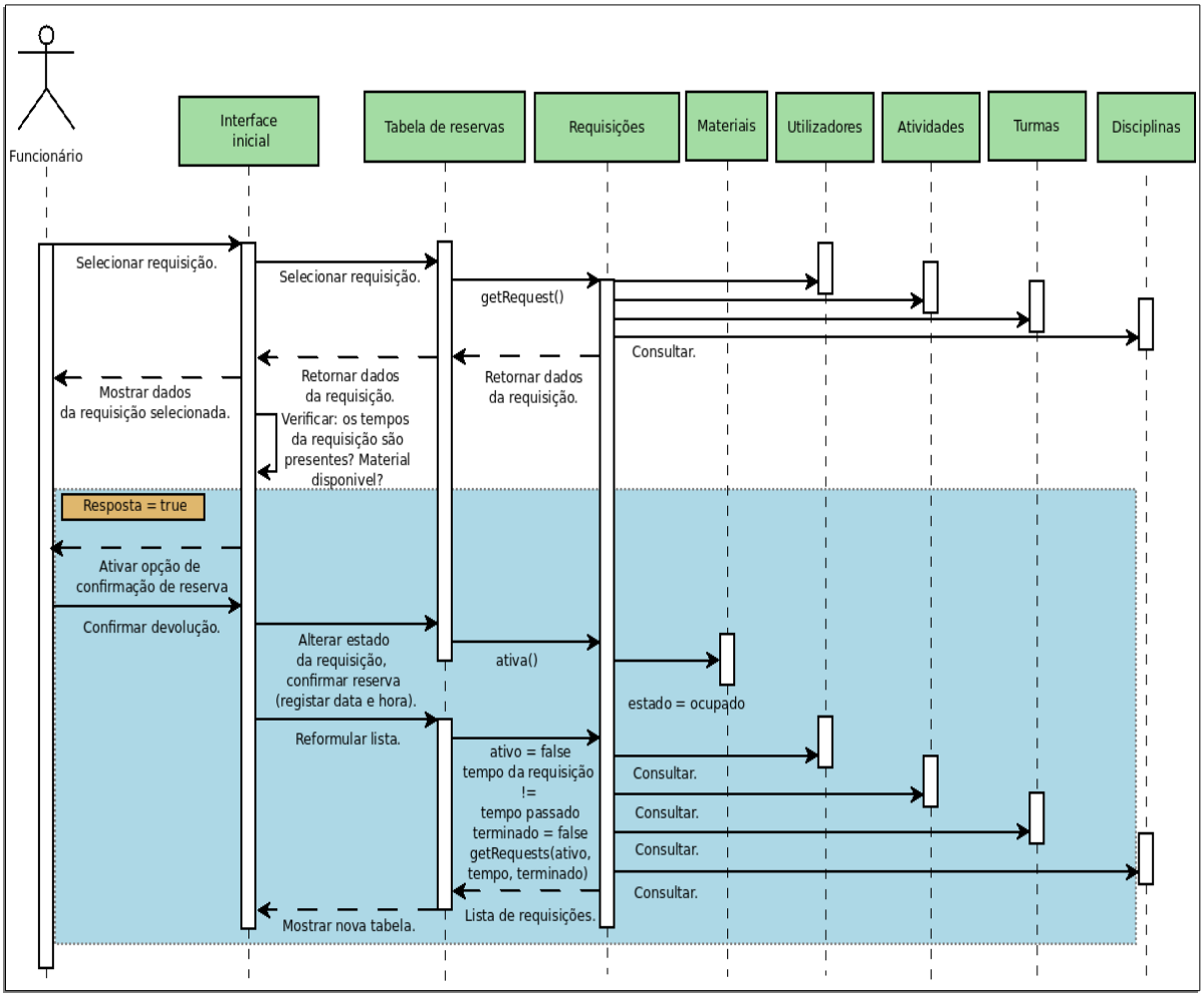


Ilustração 13: Diagrama de sequência: "Confirmar empréstimo".

Pretende-se que a interface principal apresente uma interface dividida, lista de reservas ou requisições por concretizar por um lado e lista de reservas ativas ou devoluções por um outro. As listas poderão ser visualizadas até um determinado período de tempo, procurando que o intervalo possa ser de um, dois dias, uma semana ou quinze dias. Na lista de empréstimos ou reservas por concretizar o intervalo contabilizará a data recente e datas posteriores, a sua ordenação será de forma ascendente; na lista de reservas ativas ou devoluções, o intervalo contabilizará a data presente, qualquer data posterior (se o material for levantado, a qualquer altura o requisitante pode requerer o término do empréstimo) e datas anteriores até ao máximo de quinze dias; a sua ordenação será, igualmente, ascendente.

A aplicação deverá, ainda, possuir um mecanismo de controlo que permitirá visualizar as requisições acompanhando a data e a hora do sistema. O mecanismo fará uso da barra de deslocação das tabelas, barra que se deslocará, acompanhando os tempos do sistema e posicionando-se junto a reservas ou requisições com tempos semelhantes.

3.4.3.7. Confirmar devolução.

A seleção de uma reserva, neste caso de uso, ativará de imediato a opção (botão): “confirmação da devolução”, não deverão existir restrições por intervalos de tempo: a devolução pode ocorrer antes, durante ou depois do tempo final ter concluído.

Este caso de uso conclui o processo principal da aplicação, por isso, na classe “requisição” deverão existir campos que a classifiquem como terminada ou ativa, registando, também, os tempos de levantamento e entrega, criando conjuntos para distinção aquando da procura (*queries*) na base de dados; facilitando, de igual modo, a análise estatística que pode ocorrer com mais detalhe, avaliando tempos de uso e atraso.

Nome	Confirmar devolução
Objetivo	Registar a devolução do recurso com data e hora.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Alta
Fluxo principal	**1.** Na janela principal, o ator seleciona a requisição ativa da tabela de devoluções. (1A) **2.** O sistema apresenta os dados relativos à requisição selecionada: nome do utilizador, recurso, data e hora de entrega, atividade(s) e disciplina(s) associada(s) num painel de informações. **3.** O ator confirma a devolução na opção “Confirmar devolução”. **4.** O sistema regista a devolução na base de dados, adicionando data e hora da ocorrência e reformula a tabela de requisições ativas.
Fluxos alternativos	O ator pode cancelar / desfazer a seleção em qualquer instante.
Fluxos de exceção	**1A** **a)** Não existe ligação à base de dados, o sistema apresenta mensagem do erro ocorrido: “Não existe ligação à base de dados”.
Pós-condição	O sistema deve alterar o estado do recurso para livre. O sistema deve fazer nova contagem dos recursos livres e ocupados.
Casos de teste	**1.** Verificar se a requisição apresenta o estado terminado. **2.** Verificar se o recurso apresenta o estado livre.

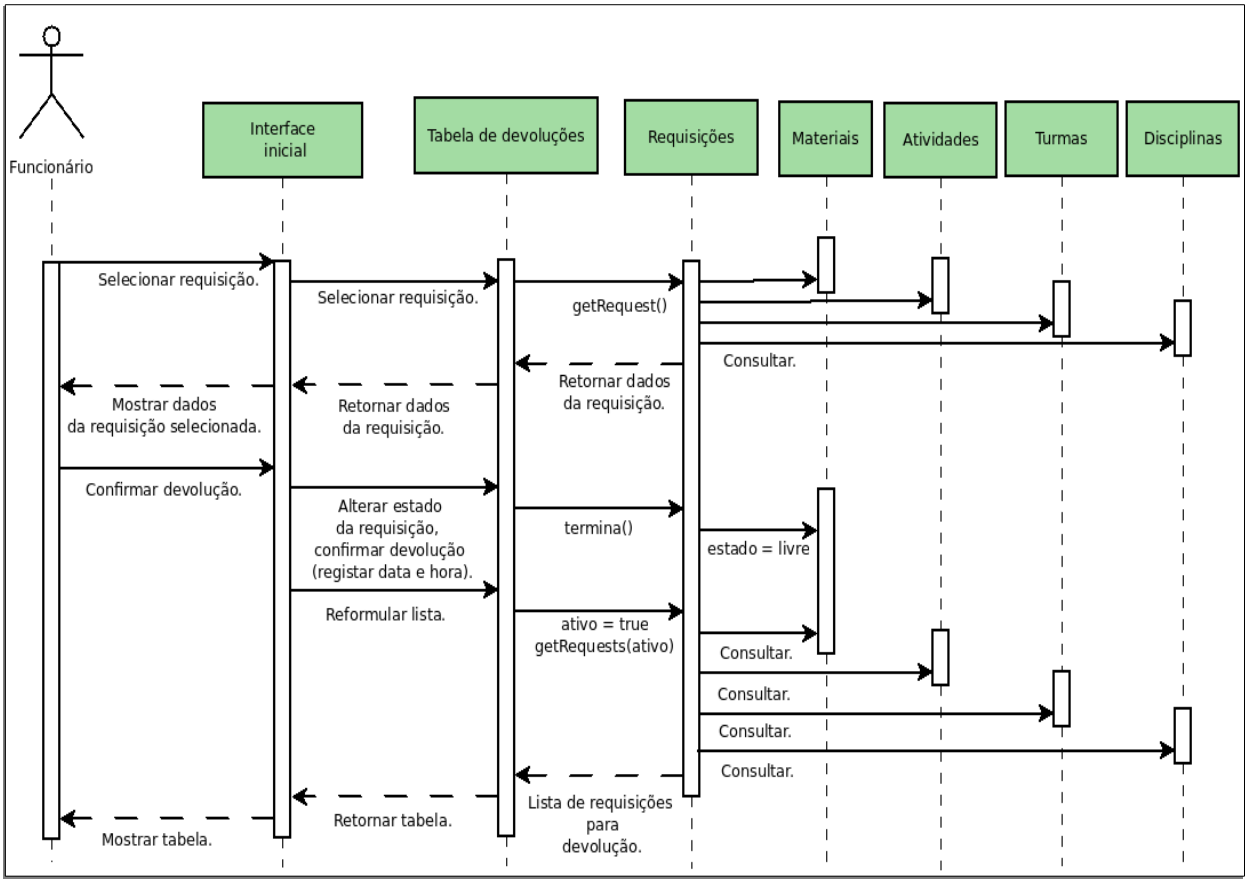


Ilustração 14: Diagrama de sequência: "Confirmar devolução".

3.4.3.8. Substituir reserva.

O caso de uso descrito ocorre com o recurso à interface inicial e à lista ou tabela de requisições por concretizar (requisições passivas), [12] identificando-se, também, uma nova interface: “Substituir requisição ou reserva”.

Para substituir uma requisição, é necessário a alteração dos tempos e / ou do recurso pretendido na requisição original. A substituição não permitirá alterar a atividade ou o utilizador da requisição, estas mudanças exigirão a criação de uma nova requisição. A requisição original não é eliminada, a classe ou objeto a construir deverá ter um campo onde possa ser registada a identificação da requisição suplente, marcando, também, o seu novo estado.

Nome	Substituir reserva
Objetivo	Substituir a reserva original por uma alternativa com diferente recurso, data e hora.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Alta
Fluxo principal	**1.** O sistema apresenta um tabela com as reservas “passivas” (ainda não concretizadas) e atualiza as entradas conforme a data e a hora atual. (1A) **2.** O ator seleciona a reserva que pretende alterar.

12 A aplicação estaria mais completa se este caso de uso pudesse ocorrer ao visualizar a atividade de cada utilizador ou de cada recurso. Neste projeto, esta situação não foi implementada.

	3. O sistema apresenta os dados relativos à requisição num painel de informações. **4.** O ator seleciona a opção “Substituir requisição”. **5.** O sistema apresenta janela com campos que permitem alterar o recurso, a data e a hora. **6.** O ator escolhe datas, horas e recurso. Confirma valores. (6A) (6B) **7.** O sistema ativa opção de confirmação da alteração e mostra resumo da alteração. **8.** O ator confirma. (8A) **9.** O sistema apresenta mensagem de realização do pedido.
Fluxos alternativos	**6A** **a)** O ator seleciona o mesmo recurso, as mesmas datas e horas. **b)** O sistema apresenta mensagem: “Os valores escolhidos são os mesmos da requisição original”.
	6B **a)** O ator seleciona uma data que colide com feriados. **b)** O sistema alerta o ator: “As datas de empréstimo e devolução não podem colidir com feriados”.
Fluxos de exceção	**1A** **a)** Não existe ligação à base de dados, o sistema apresenta mensagem do erro ocorrido.
	8A **a)** A substituição dos registos na base de dados não ocorreu, o sistema cancela a operação e mostra mensagem.
Pós-condição	O sistema altera o estado da requisição original para substituída preenchendo o campo apropriado com a identificação da nova requisição.
Casos de teste	**1.** Verificar se requisição substituída é retirada da lista de reservas “passivas”. **2.** Verificar se a escolha de uma data que represente um feriado faz apresentar mensagem de aviso. **3.** Verificar que ao escolher valores iguais aos da requisição original o sistema apresenta mensagem de aviso. **4.** Verificar validade e integridade dos dados.

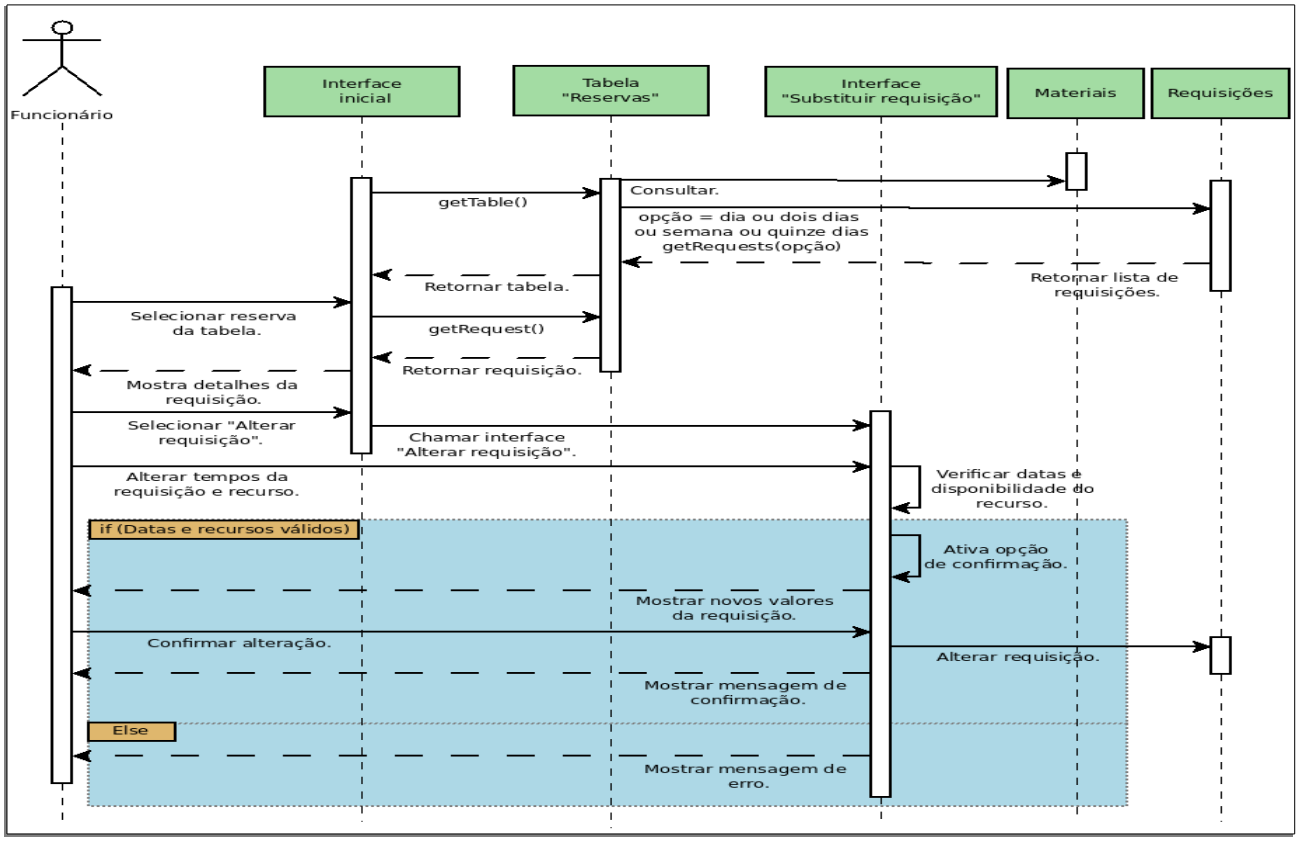


Ilustração 15: Diagrama de sequência: “Substituir reserva”.

3.5. Diagrama de classes.

O objetivo dos capítulos anteriores passava, sobretudo, pela identificação de interfaces e classes, as classes tornadas em objetos que, depois, permitirão manipular os dados e formular os comportamentos. São estas as classes identificadas ou mencionadas: requisições, utilizadores, materiais, tipos de material, funções, caraterísticas, atividades, turmas, disciplinas, software, medidas, instalações, privilégios e credenciais.

A classe "requisições" aglutinará toda a atividade do sistema, o diagrama na página seguinte mostrará uma estrutura em estrela, com as demais classes servindo a sua construção. Esta classe deverá ter referências (ligações de dependência) a outras instâncias obrigatórias: utilizadores e materiais; e instâncias facultativas como: atividades, disciplinas e turmas. Será uma classe composta por vários campos de tipo data e hora, campos necessários para estabelecer períodos, tempos de levantamento e tempos de entrega. Será, também, uma classe que possuirá campos representativos de estados: ativa, terminada, substituída, requisição conjunta.

O campo "substituída" (preenchido com a identificação de outra requisição) servirá para representar um estado, sendo que um valor distinto a 0 (valor que será definido para defeito) será manifestação de uma nova condição. O campo requisição conjunta (abordar-se-á com mais detalhe no capítulo "Documento CSV") servirá para identificar um conjunto de requisições com diferentes atividades.

A classe utilizadores será composta pelos seguintes campos: nome, identificação, correio eletrónico, telefone, data de criação do registo, privilégio associado ao registo e função relacionada com o utilizador na instituição (aluno, funcionário, professor, etc.). O sistema permitirá a inscrição de utilizadores externos à instituição.

A classe materiais terá várias tabelas associadas ou relacionadas, pretendendo-se que a pesquisa e a seleção de materiais possa ser efetuada com o recurso a critérios como caraterísticas de cada material ou software que possa estar instalado. A classe materiais terá dependência hierárquica da classe tipos de material que poderá ser tratada como classe abstrata aquando da implementação. Os campos que farão parte da classe materiais serão os seguintes: descrição, código (identificação do material na instituição), estado ou condição de empréstimo, imagem e tipo de material (identificação do tipo de material).

Para que as classes software e caraterísticas possam ter ligação com a classe materiais, na base de dados será essencial a existência das tabelas medidas e instalações que, para além de acrescentarem informação, irão permitir estabelecer o relacionamento entre elas. E, se na base de dados, a sua existência é necessária, aquando da codificação ou desenvolvimento da aplicação, essas classes tenderão a fazer parte dos objetos software e caraterísticas, e estes ligados a materiais através de listas ou conjuntos. A construção de métodos que listem esses objetos é, assim, imperativa.

O diagrama de classes apresenta-se na página seguinte.

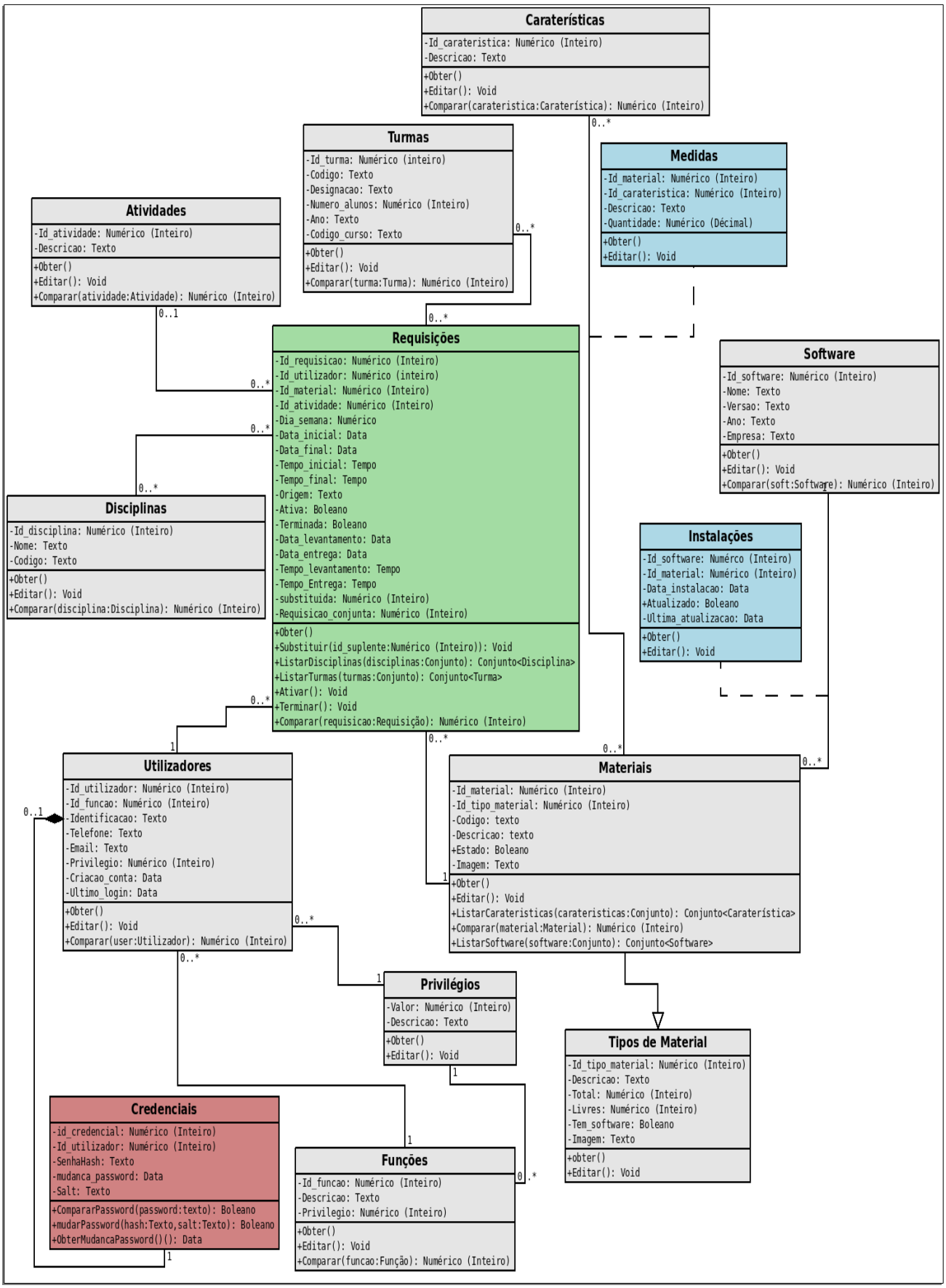


Ilustração 16: Diagrama de classes.

3.6. Modelo ER e semântica de dados.

O modelo entidade relacionamento é criado a partir do diagrama de classes. A sua existência surge com o sentido de clarificar os relacionamentos entre classes ou entidades, surge, também com o sentido de identificar chaves primárias e estrangeiras, tipos de dados e o seu tamanho. Nos capítulos seguintes, pretende-se descrever as classes ou entidades, analisando os elementos e os métodos da classe.

3.6.1. Modelo entidade relacionamento.

É o modelo necessário para antever e justificar a criação de tabelas relacionais aquando da criação do diagrama físico de base de dados. Assim, ao analisar-se relacionamentos com a condição muitos para muitos podemos perceber a necessidade de criação de mais duas tabelas relacionais, é o caso do relacionamento entre disciplinas e requisições, e entre turmas e requisições. Quando se apresentar o diagrama físico de base de dados no capítulo destinado ao desenvolvimento da aplicação, estas tabelas deverão estar representadas naquele. Eis o diagrama criado.

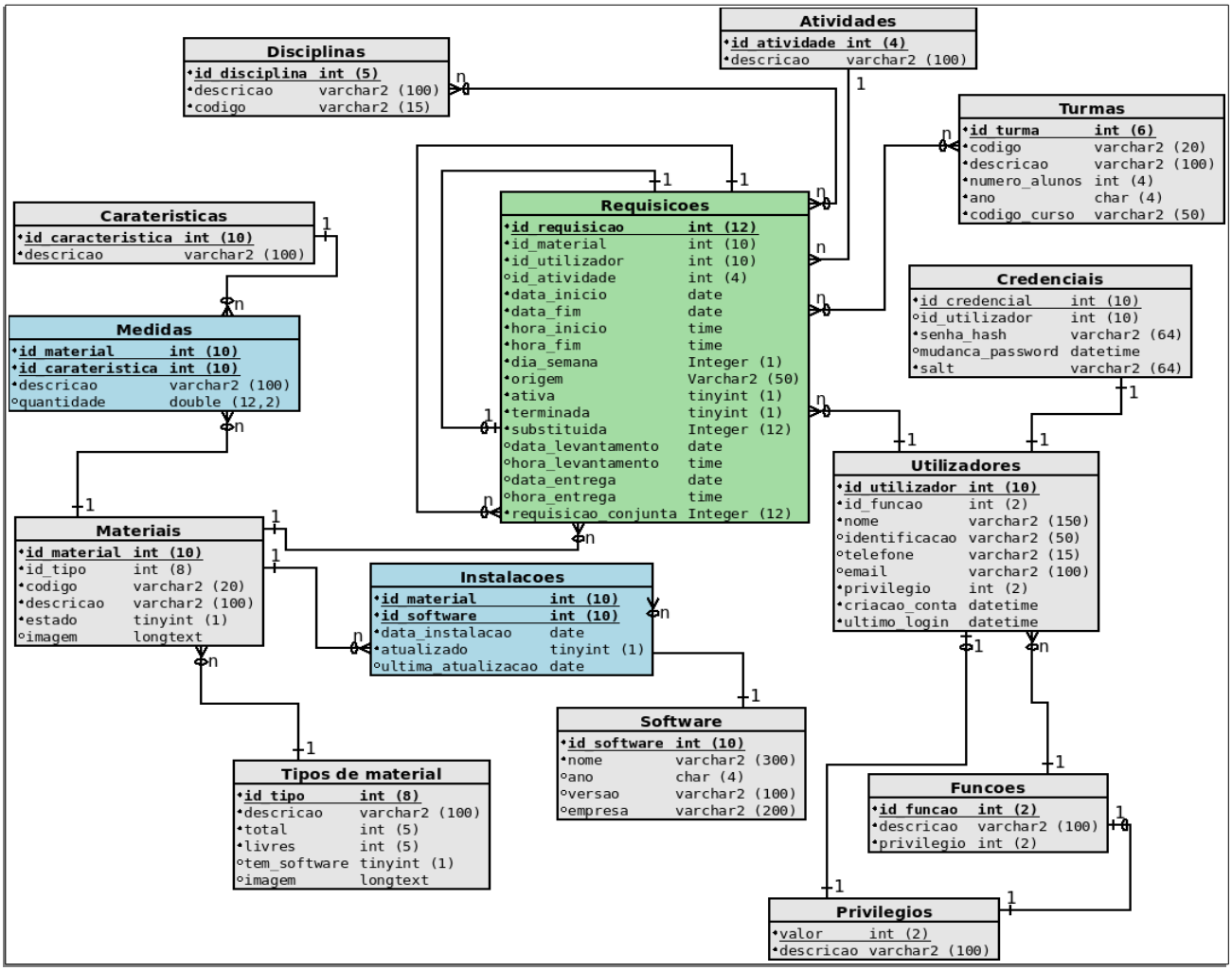


Ilustração 17: Modelo entidade relacionamento.

3.6.2. Dicionário de dados.

Com este capítulo pretende-se descrever os campos de classe, apresentar o tipo de dados, valores válidos e seus formatos. Planeia-se, também, descrever os principais métodos encontrados para cada classe, para além dos métodos de edição e obtenção de dados (GETs e SETs), algumas classes terão comportamentos e métodos próprios que é necessário explicar e exemplificar. Na maior parte das classes existirão métodos de comparação, pretendendo-se que estes objetos implementem a interface "Comparable", método que permite não só a comparação entre objetos, mas, também, a ordenação daqueles em listas ou conjuntos.

3.6.2.1. Classe "Disciplinas".

Disciplinas					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_disciplina	Integer	5	Chave primária da tabela.	Até 5 dígitos.	Maiores que 0. Não nulo. Único.
descricao	Varchar2	100	Nome da disciplina.	Até 100 carateres.	Não nulo.
codigo	Varchar2	15	Código de identificação da disciplina na Instituição.	Até 15 carateres.	Não nulo. Único.

Operações: Classe 'Disciplinas'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar (String descricao)	Conjunto de operações (set) que permitem manipular os campos da classe.
Comparar (Disciplina disciplina)	Operação que permite comparar a instância com outro objeto. 1. Um objeto da mesma classe é passado como parâmetro. 2. O sistema compara os campos 'descricao' e 'codigo' dos objetos. 3. O sistema retorna a diferença. Com valor a 0, os dois objetos são iguais.

3.6.2.2. Classe "Atividades".

Atividades					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_atividade	Integer	4	Chave primária da tabela.	Até 4 dígitos.	Maiores que 0. Não nulo. Único.

descricao	Varchar2	100	Descrição ou nome da atividade.	Até 100 carateres.	Não nulo.

Operações: Classe 'Atividades'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.
Integer Comparar (Atividade atividade)	Operação que permite comparar a instância com outro objeto. 1. Um objeto da mesma classe é passado como parâmetro. 2. O sistema compara o campo 'descricao' dos objetos. 3. O sistema retorna a diferença. Com valor a 0, os dois objetos são iguais.

3.6.2.3. Classe "Caraterísticas".

Caraterísticas					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_carateristica	Integer	10	Chave primária da tabela.	Até 10 dígitos.	Maiores que 0. Não nulo. Único.
descricao	Varchar2	100	Descrição ou referência da caraterística.	Até 100 carateres.	Não nulo.

Operações: Classe 'Caraterísticas'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.
Integer Comparar (Carateristica carateristica)	Operação que permite comparar a instância com outro objeto. 1. Um objeto da mesma classe é passado como parâmetro. 2. O sistema compara o campo 'descricao' dos objetos. 3. O sistema retorna a diferença. Com valor a 0, os dois objetos são iguais.

3.6.2.4. Classe "Turmas".

Turmas					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_turma	Integer	6	Chave primária da tabela.	Até 6 dígitos.	Maiores que 0. Não nulo. Único.
codigo	Varchar2	15	Código de identificação da turma na Instituição.	Até 15 carateres.	Não nulo. Único
descricao	Varchar2	100	Nome ou descrição da turma.	Até 100 carateres.	Não nulo.
numero_alunos	Integer	4	Número de alunos que compõem a turma.	Até 4 dígitos.	Maior ou igual que 0. Por defeito: 0.
ano	Char	4	Ano de criação da turma.	4 dígitos.	Não nulo.
codigo_curso	Varchar2	50	Código do curso a que pertence a turma.	Até 50 carateres.	Não nulo.

Operações: Classe 'Turmas'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.
Integer Comparar (Turma turma)	Operação que permite comparar a instância com outro objeto. 1. Um objeto da mesma classe é passado como parâmetro. 2. O sistema compara os campos 'descricao', 'codigo', 'ano' e 'codigo_curso' dos objetos. 3. O sistema retorna a diferença. Com valor a 0, os dois objetos são iguais.

3.6.2.5. Classe "Credenciais".

Credenciais					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_credencial	Integer	10	Chave primária da tabela.	Até 10 dígitos.	Maiores que 0. Não nulo. Único.
id_utilizador	Integer	10	Chave estrangeira que faz referência à tabela 'Utilizadores', campo 'id_utilizador'.	Até 10 dígitos.	Maiores que 0. Não nulo. Único.

senha_hash	Varchar2	64	Valor de Hash (SHA256) da senha de utilizador.	64 carateres.	Não nulo.
mudanca_password	Datetime	8	Data e hora da última mudança de senha de acesso.	‘d-m-Y H:i:s’	Pose ser nulo.
salt	Varchar2	64	Frase para composição do Hash da senha.	64 carateres.	Não nulo.

Operações: Classe ‘Credenciais’	
CompararPassword (String password)	Operação que permite validar a senha de utilizador. 1. O utilizador passa como parâmetro o texto que identifica a password. 2. O sistema concatena o texto introduzido com o texto existente no campo salt. 3. O sistema cria um hash do texto obtido através do algoritmo SHA256 . 4. O sistema compara o hash obtido com o texto existente no campo ‘senha_hash’. O sistema retorna verdadeiro se os valores comparados forem iguais e falso caso contrário.
mudarPassword(String password)	Operação que permite alterar a senha de utilizador. 1. O utilizador passa como parâmetro a nova senha. 2. O sistema cria automaticamente e aleatoriamente um novo valor para o campo ‘salt’. 3. Concatena a senha do utilizador com o texto obtido e registado no campo ‘salt’. 4. O sistema cria um hash (através do algoritmo SHA256) do texto concatenado e preenche o campo ‘senha_hash’ com o valor. 5. O sistema preenche o campo ‘mudanca_passoword’ com a data do sistema.
obterMudancaPassword()	Operação que permite obter o valor presente em ‘mudanca_password’.

3.6.2.6. Classe “Medidas”.

Medidas					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_material	Integer	10	Faz parte da chave primária (composta). Chave estrangeira que faz referência à tabela ‘Materiais’, campo ‘id_material’.	Até 10 dígitos.	Maiores que 0. Não nulo.
id_carateristica	Integer	10	Faz parte da chave primária (composta). Chave estrangeira que faz referência à tabela ‘Carateristicas’, campo ‘id_carateristica’.	Até 10 dígitos.	Maiores que 0. Não nulo.
descricao	Varchar2	100	Descrição da medida de valor na relação: caraterística / material.	Até 100 dígitos.	Não nulo.
quantidade	Double	12	Valor numérico da medida.	Até 10 dígitos,	Pode ser nulo.

				precedidos de 2 casas decimais.	

Operações: Classe 'Medidas'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.

3.6.2.7. Classe "Software".

Software					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_software	Integer	10	Chave primária da tabela.	Até 10 dígitos.	Maiores que 0. Não nulo.
nome	Varchar2	300	Descrição ou título do software.	Até 300 carateres.	Não nulo.
ano	Char	4	Ano de lançamento do software ou de nova versão.	4 dígitos.	Pode ser nulo.
versao	Varchar2	100	Versão do software.	Até 100 carateres.	Pode ser nulo.
empresa	Varchar2	200	Nome da empresa proprietária do software.	Até 200 carateres.	Pode ser nulo.

Operações: Classe 'Software'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.
Integer Comparar (Software soft)	Operação que permite comparar a instância com outro objeto. 1. Um objeto da mesma classe é passado como parâmetro. 2. O sistema compara os campos 'nome', 'ano', 'versao' e 'empresa' dos objetos. 3. O sistema retorna a diferença. Com valor a 0, os dois objetos são iguais.

3.6.2.8. Classe "Instalações".

Instalações					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_material	Integer	10	Faz parte da chave primária (composta). Chave estrangeira que faz referência à tabela 'Materiais', campo 'id_material'.	Até 10 dígitos.	Maiores que 0. Não nulo.
id_software	Integer	10	Faz parte da chave primária (composta). Chave estrangeira que faz referência à tabela 'Software', campo 'id_software'.	Até 10 dígitos.	Maiores que 0. Não nulo.
data_instalacao	Date	3	Data da instalação. É criada automaticamente com a criação do registo.	'd/m/Y'	Não nulo. Por defeito: data do sistema.
atualizado	Tinyint	1	Estado do software instalado no material. Atualizado: 1, desatualizado: 0.	Valores válidos: 0 ou 1.	Não nulo. Por defeito: 0.
ultima_atualizacao	Date	3	Data da última atualização. É alterado quando o campo 'atualizado' passa de 0 ao valor 1.	'd/m/Y'	Pode ser nulo.

Operações: Classe 'Instalações'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.

3.6.2.9. Classe "Tipos de material".

Tipos de material					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_tipo	Integer	8	Chave primária da tabela.	Até 8 dígitos.	Maiores que 0. Não nulo.
descricao	Varchar2	100	Descrição ou nome do tipo de material.	Até 100 carateres.	Não nulo.
total	Integer	5	Total de materiais do tipo que existe na instituição.	Até 5 dígitos.	Não nulo. Por defeito: 0.
livres	Integer	5	Número de materiais do tipo que apresentam o estado livre.	Até 5 dígitos.	Não nulo. Por defeito: 0.
tem_software	Tinyint	1	O tipo de material possui software: Sim = 1, Não = 0;	Valores válidos: 0 ou 1.	Não nulo. Pode defeito: 0.

imagem	Longtext	2^{32} - 1	Texto em Base64.	Carateres: [A-Z],[a-z],[0-9], '/', '+' e '='	Pode ser nulo.

Operações: Classe 'Tipos de material'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.

3.6.2.10. Classe "Funções".

Funções					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_funcao	Integer	2	Chave primária da tabela.	Até 2 dígitos.	Maiores que 0. Não nulo.
privilegio	Integer	2	Chave estrangeira, faz referência à tabela 'Privilegios', campo 'valor'.	Até 2 dígitos.	Maiores que 0. Não nulo.
descricao	Varchar2	100	Descrição ou nome da função.	Até 100 carateres.	Não nulo.

Operações: Classe 'Funcoes'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.

3.6.2.11. Classe "Privilégios".

Privilégios					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
valor	Integer	2	Chave primária da tabela.	Até 2 dígitos.	Maiores que 0. Não nulo.
descricao	Varchar2	100	Descrição do privilégio.	Até 100 carateres.	Não nulo.

Operações: Classe 'Privilégios'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.

3.6.2.12. Classe "Materiais".

Materiais					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_material	Integer	10	Chave primária da tabela.	Até 10 dígitos.	Maiores que 0. Não nulo. Único.
id_tipo	Integer	8	Chave estrangeira que faz referência à tabela 'Tipos_de_material', campo 'id_tipo'.	Até 8 dígitos.	Maiores que 0. Não nulo.
codigo	Varchar2	20	Código ou identificação do material na Instituição.	Até 20 carateres.	Não nulo.
descricao	Varchar2	100	Descrição ou nome do material.	Até 100 dígitos.	Não nulo.
estado	Tinyint	1	Estado de empréstimo.	Valores válidos: 0 ou 1.	Não nulo. Por defeito: 0.
imagem	Longtext	$2^{32} - 1$	Texto em Base64.	Carateres: [A-Z],[a-z],[0-9], '/', '+' e '='	Pode ser nulo.

Operações: Classe 'Materiais'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.
Set<Caraterística> ListarCarateristicas (Set<Carateristica> carateristicas)	Operação que permite listar as caraterísticas do material. 1. Um conjunto do tipo "Carateristica" é passado como parâmetro. 2. O sistema verifica a existência de relacionamentos, avaliando os registos na tabela 'Medidas'. 3. O sistema filtra os registos com correspondência. 4. O sistema retorna um novo conjunto.
Integer Comparar (Material material)	Operação que permite comparar a instância com outro objeto. 1. Um objeto da mesma classe é passado como parâmetro. 2. O sistema compara os campos, 'codigo', 'descricao', 'imagem' dos objetos.

	3. O sistema retorna a diferença. Com valor a 0, os dois objetos são iguais.
Set<Software> ListarSoftware (Set<Software> software)	Operação quer permite listar o software associado ao material. 1. Um conjunto do tipo "Software" é passado como parâmetro. 2. O sistema verifica a existência de relacionamentos, avaliando os registos na tabela 'Instalacoes'. 3. O sistema filtra os registos de "Software" com correspondência. 4. O sistema retorna um novo conjunto.

3.6.2.13. Classe "Utilizadores".

Utilizadores					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_utilizador	Integer	10	Chave primária da tabela.	Até 10 dígitos.	Maiores que 0. Não nulo.
id_funcao	Integer	2	Chave estrangeira, faz referência à tabela 'Funcoes', campo 'id_funcao'.	Até 2 dígitos.	Maiores que 0. Não nulo.
privilegio	integer	2	Chave estrangeira, faz referência à tabela 'Privilegios', campo 'valor'.	Até 2 dígitos.	Maiores que 0. Não nulo.
nome	Varchar2	150	Nome de utilizador.	Até 150 carateres.	Não nulo.
identificacao	Varchar2	50	Identificação do utilizador na instituição, é usado como 'username'.	Até 50 carateres.	Pode ser nulo, se email != nulo.
telefone	Varchar2	15	Telefone ou contacto do utilizador.	Formato telefónico.	Pode ser nulo.
email	Varchar2	100	Correio eletrónico do utilizador, é usado como 'username'.	Formato de correio eletrónico.	Pode ser nulo, se identificacao != nulo.
criacao_conta	Datetime	8	Data da criação da conta, é gerado automaticamente pelo sistema na criação do registo.	'd/m/Y H:i:s'	Não nulo. Por defeito: data do sistema.
ultimo_login	Datetime	8	Data do último login realizado. Gerada pelo sistema automaticamente.	'd/m/Y H:i:s'	Não nulo. Por defeito: data do sistema.

Operações: Classe 'Utilizadores'	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Editar ()	Conjunto de operações (set) que permitem manipular os campos da classe.
Integer Comparar (Utilizador user)	Operação que permite comparar a instância com outro objeto.

	1. Um objeto da mesma classe é passado como parâmetro. 2. O sistema compara os campos ‘nome’, ‘email’, ‘identificacao’, ‘telefone’ e ‘criacao_conta’ dos objetos. 3. O sistema retorna a diferença. Com valor a 0, os dois objetos são iguais.

3.6.2.14. Classe “Requisições”.

Requisições					
Nome	**Tipo de dados**	**Tamanho**	**Descrição**	**Formato**	**Restrições**
id_requisicao	Integer	12	Chave primária da tabela.	Até 12 dígitos.	Maiores que 0. Não nulo.
id_material	Integer	10	Chave estrangeira, faz referência à tabela ‘Materiais’, campo ‘id_material’.	Até 10 dígitos.	Maiores que 0. Não nulo.
id_utilizador	Integer	10	Chave estrangeira, faz referência à tabela ‘utilizadores’, campo ‘id_utilizador’.	Até 10 dígitos.	Maiores que 0. Não nulo.
id_atividade	Integer	4	Chave estrangeira, faz referência à tabela ‘Atividades’, campo ‘id_atividade’.	Até 4 dígitos.	Maiores que 0. Não nulo.
data_inicio	Date	3	Data de inicio da reserva ou requisição.	‘d/m/Y’	Não nulo.
data_fim	Date	3	Data final da reserva ou requisição.	‘d/m/Y’	Não nulo.
hora_inicio	Time	3	Hora de inicio da reserva ou requisição.	‘H:i:s’	Não nulo.
hora_fim	Time	3	Hora final da reserva ou requisição.	‘H:i:s’	Não nulo.
dia_semana	Integer	1	Dia da semana identificado com o número de 1 (domingo) até 7 (sábado).	[1-7]	Não nulo.
origem	Varchar2	50	A requisição pode ter duas origens: por documento ‘CSV’ ou por registo ‘local’.	‘csv’ ou ‘local’	Não nulo.
ativa	Tinyint	1	Estado da requisição: ainda não realizada ou terminada = 0, ativa = 1.	Valores válidos: 0 ou 1.	Não nulo. Por defeito: 0.
terminada	Tinyint	1	A requisição está concluída: ainda não realizada ou ativa = 0, realizada = 1.	Valores válidos: 0 ou 1.	Não nulo. Por defeito: 0.
substituida	Integer	12	Identifica a requisição que substitui a atual. Chave estrangeira que relaciona a mesma tabela e o campo ‘id_requisicao’.	Até 12 dígitos.	Maior ou igual a 0. Não nulo. Por defeito: 0.
			Data de levantamento do material		

data_levantamento	Date	3	junto ao funcionário.	‘d/m/Y’	Pode ser nulo.
hora_levantamento	Time	3	Hora de levantamento do material junto ao funcionário.	‘H:i:s’	Pode ser nulo.
data_entrega	Date	3	Data de entrega ou devolução do material.	‘d/m/Y’	Pode ser nulo.
hora_entrega	Time	3	Hora de entrega ou devolução do material.	‘H:i:s’	Pode ser nulo.
requisicao_conjunta	Integer	12	Identifica a requisição à qual se junta num único bloco. Chave estrangeira que relaciona a mesma tabela e o campo ‘id_requisicao’.	Até 12 dígitos.	Maior ou igual a 0. Não nulo. Por defeito: 0.

Operações: Classe ‘Requisições’	
Obter ()	Conjunto de operações (get) que permitem obter os campos da classe.
Substituir (Integer id_suplente)	Operação que permite preencher o campo ‘substituida’. 1. O parâmetro identifica a requisição que se pretende ser substituta. 2. O sistema verifica a existência da requisição na base de dados. 3. Com resposta afirmativa, o sistema preenche o campo ‘substituida’ do objeto com o valor passado em parâmetro.
Set<Disciplina> ListarDisciplinas (Set<Disciplina> disciplinas)	Operação quer permite listar as disciplinas associadas à requisição. 1. Um conjunto do tipo “Disciplina” é passado como parâmetro. 2. O sistema verifica a relação da requisição com os registos do conjunto, analisando as existências na tabela ‘Rel_Disciplinas_Requisicoes’. 3. O sistema filtra os registos com correspondência. 4. O sistema retorna um novo conjunto.
Set<Turma> ListarTurmas (Set<Turma> turmas)	Operação quer permite listar as turmas associadas à requisição. 1. Um conjunto do tipo “Turma” é passado como parâmetro. 2. O sistema verifica a relação da requisição com os registos do conjunto, analisando as existências na tabela ‘Rel_Turmas_Requisicoes’. 3. O sistema filtra os registos com correspondência. 4. O sistema retorna um novo conjunto.
Ativar()	Operação que permite tornar a requisição ativa. 1. O sistema coloca o valor 1 no campo ‘ativa’. 2. O sistema verifica qual o material associado à requisição e preenche o campo ‘estado’ daquela instância com 1 (ocupado). 3. O sistema preenche os campos ‘data_levantamento’ e ‘hora_levantamento’ com a data e hora do sistema, respetivamente.
Terminar()	Operação que permite concluir a requisição, com a devolução do material. 1. O sistema coloca o valor 0 no campo ‘ativa’ e 1 no campo ‘terminada’. 2. O sistema verifica qual o objeto associado à requisição e preenche o campo ‘estado’ daquela instância com 0 (livre). 3. O sistema preenche os campos ‘data_entrega’ e ‘hora_entrega’ com a

	data e hora do sistema, respetivamente.
Integer Comparar (Requisicao requisicao)	Operação que permite comparar a instância com outro objeto. 1. Um objeto da mesma classe é passado como parâmetro. 2. O sistema compara os campos 'id_utilizador', 'id_material', 'id_atividade', 'data_inicio', 'hora_inicio', 'data_fim', 'hora_fim', 'dia_semana' e 'origem' dos objetos. 3. O sistema retorna a diferença. Com valor a 0, os dois objetos são iguais.

3.7. Diagrama de Estados.

Este é o capítulo destinado à análise de estados. Dentro de cada classe podem existir momentos que a definem com determinado valor devido a situações que a transformam. Este diagramas permitem entender os objetos como mecanismos "vivos", estes passam de estados iniciais para estados finais, através de transições que se iniciam por determinado ocorrência ou pela sua ausência. Para este projeto, identificámos dois estados (existindo mais) que revelam maior interesse; o primeiro desses estados ajusta-se ao objeto requisição, o segundo refere-se ao objeto material.

3.7.1. Estados em "requisição".

O objeto / instância requisição resultante da classe "requisições", pode encontrar-se nos seguintes estados: "efetuada", "não realizada", "ativa", "substituída" e "terminada". O estado inicial deste objeto, ao ser criado, é "efetuada", a partir dali a requisição poderá tomar o rumo normal que será "ativa", caso o utilizador levante o material dentro do período temporal que lhe diz respeito.

Poderá acontecer que a reserva ativa ultrapasse o tempo que lhe é destinado (mais 30 minutos que o tempo final de inscrição); o utilizador não restitui o material e, em situações assim, a requisição passará à condição de "atrasada", podendo esta condição ser infinita se nunca ocorrer a entrega do material. A entrega do material, por sua vez, levará a requisição para a condição de "terminada", sendo este um estado final.

A requisição poderá não ser cumprida: quando os tempos finais de inscrição forem ultrapassados, a requisição entrará num estado de "não realizada", momento em que o utilizador poderá requerer uma

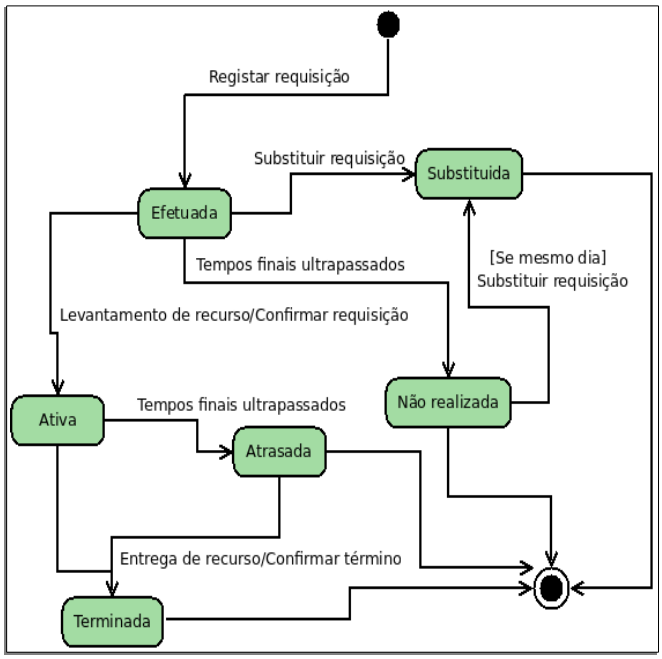


Ilustração 18: Diagrama de estados: "Requisição".

nova requisição ou a substituição daquela. O utilizador pode, mesmo, requerer a mudança de horários ou de material / recurso a qualquer altura de uma reserva passiva (desde que o dia de levantamento não seja ultrapassado), se a situação ocorrer, uma nova requisição será criada, sendo a original substituída pela última. Esta ação marcará outro estado final. “substituída” e “não realizada” são, assim, mais dois possíveis estados finais.

3.7.2. Estados em “Material”.

O diagrama que a seguir é apresentado depende das condições verificadas no diagrama anterior. São aquelas ocorrências que definem os dois estados verificados no objeto material. Esses dois estados são: livre e ocupado.

O material encontra-se, naturalmente, na condição de livre, é a requisição e o sem cumprimento que dá origem ao término deste estado. Com o levantamento do material, o estado do material passa a estar na situação de ocupado, que pode ser uma condição infinita se a requisição não for terminada (ou seja, dito de outra forma , enquanto o material ou recurso não for entregue). Uma condição infinita infere um estado final, daí que no diagrama ao lado, a transição aponte para o estado final.

Com a entrega do material, o estado do material retorna à condição inicial, livre. O estado “livre” é, também, um estado final.

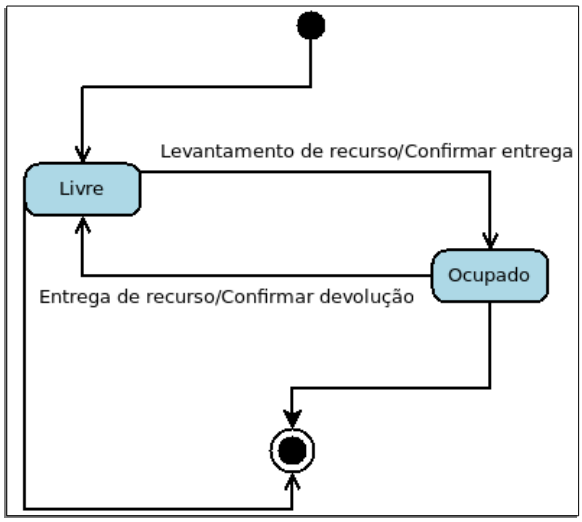


Ilustração 19: Diagrama de estados: "Material".

3.8. Documento CSV.

Um dos objetivos principais para o projeto, descritos no início deste documento, tem como foco a criação de processos de automatização no registo e criação de requisições, num processo que deverá ser conseguido através da análise e tratamento de um documento CSV. Um tipo de documento que permite a organização da informação através da separação dos seus dados com vírgula (daí o nome em inglês: *Comma Separated Value*), ponto e vírgula ou outro símbolo considerado ajustado.

Cada entrada (linha) do documento usado neste projeto possui 21 valores, sendo estes os seus nomes: data_id (numérico), CdTurma (numérico), DgTurma (Texto, representa os turnos ou grupos por cada turma), DiaSemana (numérico), HoraIni (numérico), MinutoIni (numérico), HoraFim (numérico), MinutoFim (numérico), CdRegime (numérico, código da atividade), DgRegime (descrição da atividade), NmDocente (Nome do docente), Sala (descrição da sala), CdDocente (Código do docente), CdDis (Código da disciplina), NmDis (descrição da disciplina), CdSala (Código da sala), CdPEstudo (ciclo de estudo), AbrDis (abreviatura da disciplina), CdCampus (código do campus), CdEdificio (código do edifício), CdPiso (código do piso).

Como nem todos os dados teriam referência para a aplicação pensada, aquando desta fase de estudo e planeamento, achou-se que os seguintes valores seriam suficientes na construção das requisições:

- Dia da semana, necessário para obter as datas. O programa terá que ser capaz de atribuir datas a cada dia da semana durante um determinado período de tempo.
- Hora de início e minutos de início, criar tempo de início para a requisição.
- Hora final e minutos finais para estabelecer o fim horário da requisição.
- Código e descrição da atividade (para a criação da atividade).
- Código do docente e nome do docente (para a criação do utilizador).
- Código e descrição da sala (para a criação do material).
- Código e descrição da disciplina (para a criação da disciplina).
- Código da turma e ciclos de estudo (para a construção da turma).

A informação recolhida através destes campos permitirá criar os objetos necessários para compor a requisição, composta por um utilizador, tempos iniciais e tempos finais, um dia da semana, um material, por disciplinas, turmas e uma atividade. E, como os objetos criados, a partir deste documento, se encontrarão incompletos, a aplicação deverá permitir a edição posterior.

O sistema deverá ter a capacidade de comparar objetos, evitando duplicação de dados. Este tipo de requisições será, também, classificado na base de dados com a identificação de origem, evitando-se, assim, conflitos e processos simples na atualização.

	A	B	C	D	E	F	G	H	I	J	K
1	data_id	CdTurma	DgTurma	DiaSemana	HoraIni	MinutoIni	HoraFim	MinutoFim	CdRegime	DgRegime	NmDocente
2	42338	25	T1	4	0	21	13	0	3	Teórico-Práticas	[illegible]
3	42338	25	T1	6	11	30	13	30	7	Orientação Tutorial	[illegible]
4	42338	26	T2	4	14	0	16	0	3	Teórico-Práticas	[illegible]
5	42338	17	T1	2	16	30	19	30	3	Teórico-Práticas	[illegible]
6	42338	17	T1	6	9	0	11	0	7	Orientação Tutorial	[illegible]
7	42338	18	T2	6	11	0	13	0	7	Orientação Tutorial	[illegible]
8	42338	18	T1	4	11	0	13	0	4	Prática Laboratório	[illegible]
9	42338	18	T1	5	11	0	12	0	1	Teóricas	[illegible]
10	42338	18	T1	5	12	0	13	0	7	Orientação Tutorial	[illegible]
11	42338	32	T1	5	14	0	15	0	3	Teórico-Práticas	[illegible]
12	42338	32	T1	5	15	0	16	0	7	Orientação Tutorial	[illegible]
13	42338	32	T1	5	18	0	20	0	3	Teórico-Práticas	[illegible]
14	42338	33	T2	4	16	0	18	0	3	Teórico-Práticas	[illegible]
15	42338	20	T1	3	14	0	16	0	7	Orientação Tutorial	[illegible]
16	42338	20	T1	4	14	30	16	30	4	Prática Laboratório	[illegible]
17	42338	20	T1	4	16	30	17	30	3	Teórico-Práticas	[illegible]
18	42338	14	T1	2	13	30	14	30	1	Teóricas	[illegible]
19	42338	14	T1	2	14	30	15	30	3	Teórico-Práticas	[illegible]
20	42338	14	T1	2	15	30	16	30	7	Orientação Tutorial	[illegible]
21	42338	11	T1	3	16	0	17	0	3	Teórico-Práticas	[illegible]
22	42338	11	T1	3	17	0	19	0	4	Prática Laboratório	[illegible]
23	42338	11	T1	5	11	0	12	0	7	Orientação Tutorial	[illegible]
24	42338	12	T2	5	12	0	13	0	7	Orientação Tutorial	[illegible]
25	42338	12	T1	2	14	0	17	0	3	Teórico-Práticas	[illegible]
26	42338	12	T1	3	11	0	13	0	7	Orientação Tutorial	[illegible]
27	42338	13	T2	2	17	0	20	0	3	Teórico-Práticas	[illegible]
28	42338	13	T2	4	14	30	16	30	7	Orientação Tutorial	[illegible]
29	42338	11	T1	2	14	0	17	0	3	Teórico-Práticas	[illegible]
30	42338	11	T1	6	9	30	11	30	7	Orientação Tutorial	[illegible]
31	42338	11	T1	3	10	0	11	0	1	Teóricas	[illegible]
32	42338	11	T1	3	11	0	13	0	4	Prática Laboratório	[illegible]

Ilustração 20: Documento CSV usado.

Como se estipulou que cada requisição corresponderia a um utilizador, um material e uma atividade, apercebe-mo-nos, através da análise do documento, que várias atividades (e linhas do documento) poderiam corresponder a uma aula, por exemplo, um tempo letivo poderia englobar a atividade “orientação tutorial” e atividade “teórico-prática”, não sendo necessário repetir o processo de entrega e levantamento de material. Como todos os valores são iguais, apenas diferindo os tempos iniciais e finais, foi possível estabelecer uma relação através da comparação daqueles tempos; assim, a sequencialidade dos horários (tempos finais de uma requisição são iguais que os tempos iniciais de outra) constitui um momento letivo.

Foi necessário, então, identificar o campo requisição conjunta na classe requisição. Quando aquele campo contiver o valor zero haverá significado de singularidade e quando aquele campo estiver preenchido por valores diferentes (com um número identificativo ou código de uma requisição) haverá significado de agregação. A aplicação deverá ter mecanismos de tratamento e formas de apresentação para este tipo de requisições, para que não se repita o processo de entrega e devolução.

A aplicação deverá criar uma requisição por cada data gerada, por um período de tempo estabelecido pelos utilizadores (semestres letivos), evitando feriados e períodos de paragem ou férias (será necessário criar objetos, métodos e formas de gravação para registar as escolhas do utilizador).

Como cada entrada do documento representa uma data única e distinta, não haverá preocupação com datas (data inicial e data final), no entanto, o objeto requisição deverá contemplar períodos de empréstimos maiores que um dia. O diagrama de atividade a seguir mostra o processo exigido para o tratamento deste tipo de requisições, com origem "CSV".

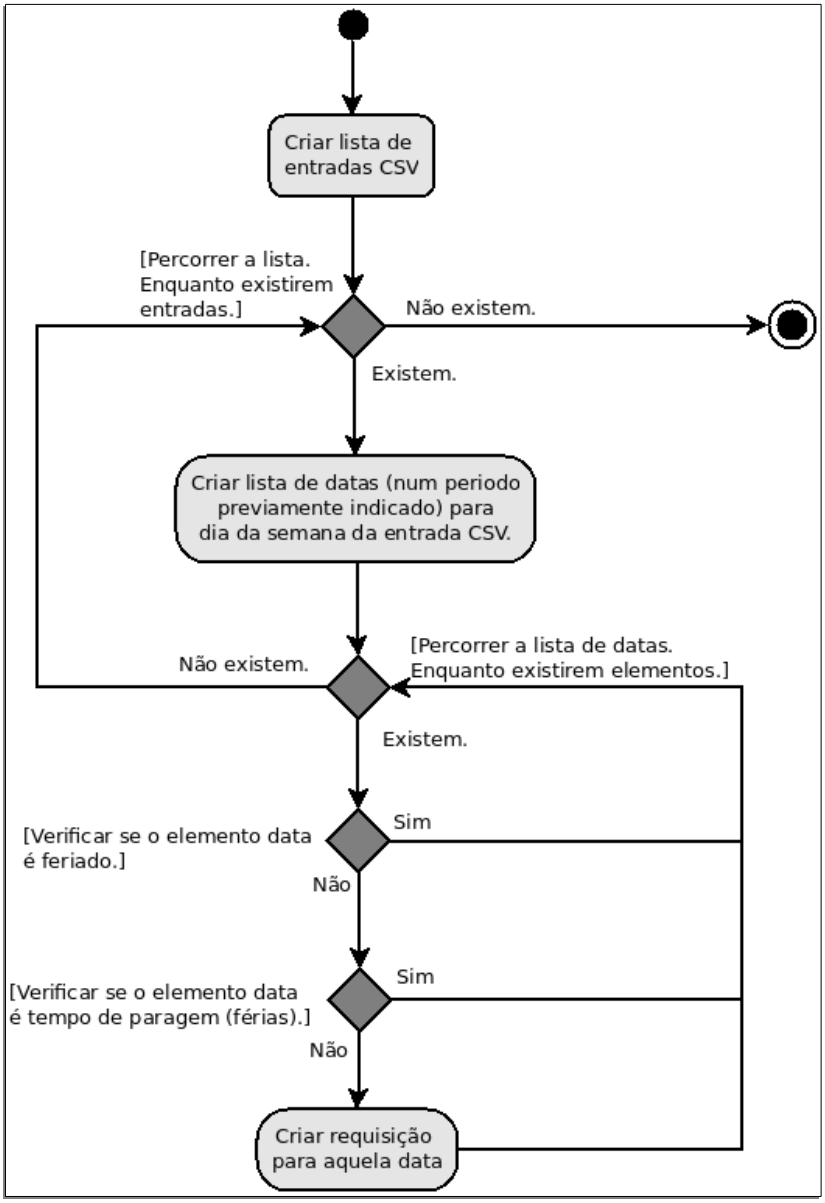


Ilustração 21: Diagrama de atividades: "Criar requisição".

3.9. Tecnologias.

Na disciplina de programação avançada a linguagem de referência é Java, na contextualização deste projeto foi referido que o início deste projeto foi ali concebido pelo que a construção da aplicação tivesse sido pensada para o seu uso. O sistema de gestão de base de dados escolhido foi o MySQL, a escolha recaiu neste sistema porque se pretendeu aprofundar conhecimentos na sua manipulação.

3.9.1. Java.

Java é uma das linguagens de programação mais usadas[13], é uma linguagem segura e os recursos que disponibiliza são vastos e variados. A comunidade de entusiastas e programadores da linguagem é enorme e a informação disponibilizada na Web permite achar rapidamente respostas e soluções. O interesse pessoal na aprendizagem e especialização desta tecnologia pesou, também, nesta escolha.

Os seguintes recursos (bibliotecas ou conjuntos daquelas escritos em Java) foram identificados como necessários para a construção da aplicação:

- *Mysql connector/J 5.1.40*, é o driver oficial para ligação à base de dados. É desenvolvido e mantido pela Oracle, empresa proprietária do sistema de gestão de base de dados (SGBD) Mysql.
- Joda-time[14], um recurso para datas com mais respostas e soluções que aquelas usadas por defeito em Java. Suporta vários tipos de calendários permitindo uma apresentação em vários formatos, possuindo várias ferramentas para cálculos sobre datas, condição necessária para este projeto.
- OpenCSV 3.8, é uma biblioteca para uso, edição, leitura e manipulação de documento CSV. Existem vários recursos para tratar aqueles tipo de documentos, no entanto, não são livres e exigem pagamento. Esta biblioteca, livre e que é publicada através da licença Apache 2.0, é uma biblioteca com recursos simples, mas que permite atingir os objetivos deste trabalho.
- SvgSalamander, é uma biblioteca que permite a renderização de imagens vetoriais. Pensa-se no uso deste tipo de imagens em partes da interface principal da aplicação, uma interface que se pensa dinâmica e que se redimensiona, este tipo de imagens permitirá acompanhar as mudanças do ecrã, sem grande preocupação com a lógica de programação. Estas imagens terão, sobretudo, ligação com a classe tipos de material.
- Swingx 1.0, é uma biblioteca que estende o normal conjunto de objetos gráficos e visuais do JDK (*Java Development Kit*), javax.Swing. Para além do acréscimo de funcionalidades aos objetos, esta biblioteca disponibiliza alguns recursos gráficos / visuais para manipulação de datas, condição que originou a escolha e seleção deste recurso.

13 Ver: http://www.tiobe.com/tiobe-index/.
14 Ver: http://www.joda.org/joda-time/.

- Apache POI, é uma API (*Application Programming Interface*) disponibilizada pela empresa que lhe dá nome. A sua existência deve-se à procura de criar soluções para a manipulação de documentos OpenOffice.[15] No entanto, a sua usabilidade vai mais longe, oferecendo ferramentas para edição e criação de documentos xls (Excel), doc (Office) e ppt (Powerpoint). Neste projeto, o seu uso será necessário quando se desenvolverem funções que permitam a criação e exportação de documentos para Excel.
- Apache PDFBox, é uma biblioteca em código aberto para trabalho com documentos do tipo PDF. Permite criar, manipular e extrair informação. Está distribuída através da licença da Apache 2.0. O seu uso na aplicação permitirá criar documentos sob aquele formato e a sua posterior impressão.
- JfreeChart 1.0.19, é uma biblioteca que permite a criação e a apresentação de vários tipos de gráficos. Tem suporte para os componentes javax.swing, podendo usar-se as classes Jdialog ou Jframe para "habitáculo" dos gráficos. Esta biblioteca permite, ainda, a exportação para imagem: *.png, *.jpeg, *.pdf, *.eps e *.svg.
- Junit 4.12, é uma framework que permite implementar ou criar teste unitários repetitivos. Será uma ferramenta útil aquando da realização de testes, validação e verificação da integridade dos dados.

3.9.2. MySQL.

O sistema de gestão de base de dados MySQL é um sistema sólido, com bastante aceitação e uso em aplicações Web. Devido à sua robustez, segurança, eficácia e integração com diferentes linguagens de programação, fruto do trabalho de uma grande comunidade de programadores, a sua aceitação e escolha é grande. O seu nome deriva da combinação My, o nome da primeira filha de um dos fundadores do sistema, Michael Widenius, e da abreviatura SQL, sigla inglesa para linguagem de pesquisa / procura estruturada.[16] O sistema é distribuído sob os termos da licença GNU, embora existam versões com funcionalidades adicionais que exijam compensação económica.

O MySQL pode ser distribuído através de aplicações auxiliares que permitem a instalação completa e a configuração automática de um servidor local (Apache, MySQL, PHP e Perl). O Xampp (Lamp em Linux) é uma dessas aplicações, bastante recorrente em atividades letivas e projetos pessoais, permitiu ao aluno uma aproximação e reconhecimento do MySQL. Esta situação motivou, em grande parte, a sua escolha.

3.9.3. Git.

É uma ferramenta para controlo de versões. O seu uso é essencial para garantir o controlo no desenvolvimento e a segurança da preservação de dados. É uma ferramenta que permite a construção de

15 Ver: https://poi.apache.org/.
16 Em, https://en.wikipedia.org/wiki/MySQL, última atualização a 03 de Novembro de 2016.

software de uma forma não linear, possibilitando um controlo completo sobre as alterações, evitando versões conflituosas que resultam em erros e perda de informação.

Todo os momentos de desenvolvimento serão lançados em repositório local, mas também remoto, através do GitHub que providencia serviços de alojamento para projetos que utilizem o Git. O repositório será público já que daquela forma não existe custo associado e dá lugar à partilha de código.

3.10. Componentes e instalação.

Com o capítulo anterior pretendeu-se realizar o reconhecimento das tecnologias, bibliotecas e recursos de apoio necessários, neste capítulo pretende-se identificar a lógica de serviços e comunicação, relacionamento entre componentes e ligação entre diferentes espaços (através do diagrama de componentes, imagem que pode ser consultado em baixo).

3.10.1. Diagrama de componentes.

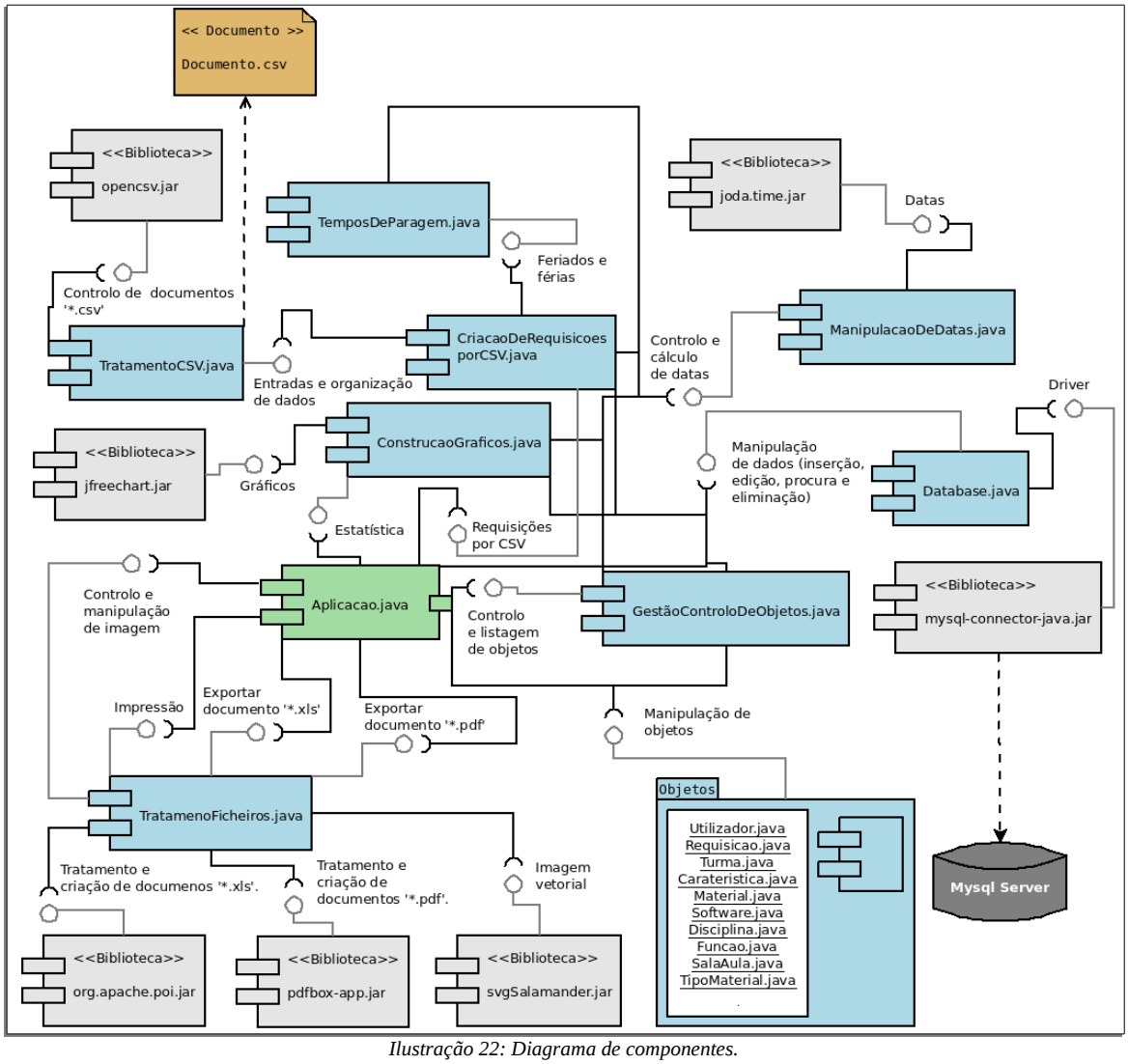


Ilustração 22: Diagrama de componentes.

Os componentes são objetos com caraterísticas e funcionalidades próprias, podem ser classes, grupos de classes, documentos ou interfaces gráficas, são pedaços da aplicação que realizam determinadas funções e serviços. Cada um deles presta funcionalidades e as recebe através de processos de comunicação e invocação.

Os componentes relacionam-se através dum processo de comunicação e dependência, e o diagrama de componentes atrás permite fazer uma análise dessa relação. Por exemplo, o componente de manipulação de base dados irá necessitar do driver MySQL que permite a comunicação com a base de dados. Por sua vez, o grupo de classes que servirão para a manipulação de objetos terão que requisitar ou invocar os métodos do componente que existirá para manipulação da base de dados, terão que requisitar os métodos do componente que servirá para a gestão de datas, e terão que requisitar os métodos dos objetos da aplicação.

Da mesma forma, o componente que irá permitir a manipulação do documento CSV requisitará os serviços da biblioteca opencsv.jar, e a classe ou componente, que realizar as requisições através daquele método, necessitará de métodos para tratamento de datas, de métodos para tratamento de feriados e períodos de paragem e de métodos para manipulação de base de dados.

O componente que se construir para o tratamento de imagens e ficheiros terá que requisitar os métodos ou serviços das bibliotecas: org.apache.poi.jar, pdfbox-app.jar e svgSalamander.jar. E, por sua vez, este concretizará as funções que a aplicação (interface gráfica) necessitará. Esta interface gráfica (conjunto de janelas da aplicação) será o componente aglutinador, componente para onde se dirigirão a maior parte dos serviços identificados.

3.10.2. Diagrama de instalação.

O diagrama de instalação que se apresenta a seguir mostra a estrutura física da aplicação, para que esta possa ser executada com sucesso será necessário localizar as partes do sistema, estabelecer os canais de ligação e configurar os ambientes ou sistemas operativos. A aplicação dependerá da instalação do Java Virtual Machine na máquina a que se destina, dependerá da existência do documento CSV e da correta localização daquele, dependerá da configuração do servidor, da instalação do sistema de gestão de base de dados MySQL e da criação do script que permitirá criar a base de dados, dependerá, por fim, da existência de um canal de ligação TCP / IP entre a máquina e o servidor.

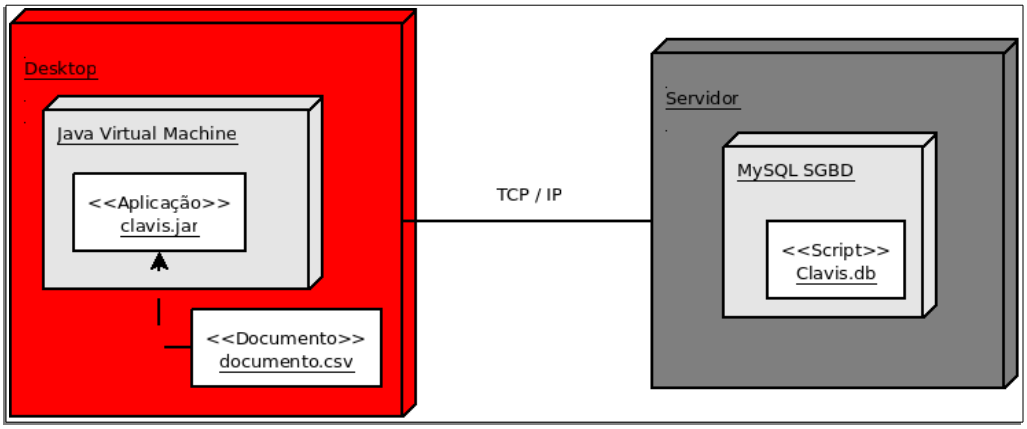


Ilustração 23: Diagrama de instalação.

4. Desenvolvimento.

Até agora usaram-se tempos verbais futuros do presente, já que os artigos anteriores procuraram ser textos de desenho e projeção da aplicação. A partir de agora procurar-se-ão descrever os passos tomados após a concretização daquele trabalho de engenharia, e, por isso, os tempos tomarão, em grande parte, a forma do pretérito perfeito.

4.1. Objetos auxiliares.

Para construir a aplicação, foi necessário a construção de alguns objetos que permitissem o controlo e a manipulação das datas, dos ficheiros e das imagens. O primeiro grupo em análise, as datas e tempos, foi um conjunto (package) gerado com os seguintes elementos:

- Date.java,
- Time.java,
- Holiday.java,
- Dinamicholiday.java,
- HolidaysList.java,
- WeekDay.java,
- Semester.java,
- SemesterList.java,
- BreakPeriod.java,
- BeakPeriodList.java.

Neste projeto deu-se especial atenção às classes “WeekDay.java”, “Time.java”, “HoliDay.java” e “Date.java” que introduziram um conjunto de métodos que adicionaram funcionalidades às classes “standard” do Java para controlo de datas (java.util.Date e java.util.Calendar) e períodos de tempo. No entanto, será necessário referir que uma boa parte destes métodos existem na classe org.joda.time.DateTime, pertencente à *framework* a que já se fez menção no capítulo “Tecnologias”.

Três métodos podem ser destacados na classe Date.java: dateAfter(Integer dias), dateBefore(int dias) e isBigger(Outra Data); o primeiro permite a obtenção de uma nova data a partir de uma data que serve de referência à instância, o segunda retorna uma data anterior, a terceira é uma classe que retribui o número de dias entre duas datas (para além disso permite fazer a comparação entre datas, já que retorna qualquer valor inteiro, permitindo conhecer se uma data é maior, menor ou igual que outra).

O algoritmo, na página seguinte, mostra a lógica por detrás do método dateAfter(Integer dias).[17]

17 Os restantes algoritmos podem ser encontrados em anexo.

```
ALGORITMO "Data após tantos dias (ndias)"
VARIÁVEIS
        ano: Inteiro, mês: Inteiro , dia : Inteiro,
        ndias: Inteiro, meses[13] : Inteiros, ndias_auxiliar Inteiro,
        ndia_do_ano: Inteiro, anoauxiliar: Inteiro
INICIO
        LEIA ndias, ano, mês, dia
        meses[0] ← 0, meses[1] ←31, meses[2] ← 28, meses[3] ← 31, meses[4] ← 30,
        meses[5] ← 31, meses[6] ← 30, meses[7] ← 31, meses[8] ←31, meses[9] ←30,
        meses[10] ← 31, meses[11] ← 30,          meses[12] ← 31
        SE  ( ndias > 0 ) ENTÃO
                SE ( ( ndias + dia ) > meses[mês] ) ENTÃO
                        LEIA ndia_do_ano
                        ndias_auxiliar = ndia_do_ano + ndias
                        SE ( Bissexto ( ano ) ) ENTÃO
                                anoauxiliar ← 366
                        SENÃO
                                anoauxiliar ← 365
                        FIM SE
                        ENQUANTO ( ndias_auxiliar > anoauxiliar ) FAÇA
                                SE ( Bissexto ( ano ) ) ENTÃO
                                        ndias_auxiliar ← ndias_auxiliar - 366
                                SENÃO
                                        ndias_auxiliar ← ndias_auxiliar - 365
                                FIM SE
                                ano++
                                SE ( Bissexto ( ano ) ) ENTÃO
                                        anoauxiliar ←366
                                SENÃO
                                        anoauxiliar ←365
                        FIM ENQUANTO
                        SE ( Bissexto ( ano ) ) ENTÃO
                                meses[2] ←29
                        FIM SE
                        PARA incremento DE 1 ATÉ 12 FAÇA
                                mês ←incremento
                                SE ( ndias_auxiliar > meses[incremento] ) ENTÃO
                                        ndias_auxiliar ←ndias_auxiliar - meses[incremento]
                                SENÃO
                                        dia ← ndias_auxiliar
                                        RETORNA dia, mês, ano
                        FIM PARA
                SENÃO
                        dia ←dia + ndias
                        RETORNA dia, mês, ano
                FIM SE
        FIM SE
RETORNA dia, mês, ano
FIM ALGORITMO
```

Na classe Time.java destaca-se o método addSeconds(Integer segundos) que permite obter uma nova hora a partir da hora ou tempo de referência para a instância. Da classe WeekDay.java sublinha-se o método que obtém o dia da semana (valor inteiro entre 1 e 7) a partir de uma data[18].

A classe Holiday.java possui os seguintes métodos: getMobileHolidays(Integer ano) e setExpanded(). O primeiro retorna uma lista (array) de datas com os feriados móveis do ano indicado em parâmetro; na sua construção houve recurso ao algoritmo de Gauss[19] para o cálculo da Páscoa, os demais feriados móveis

18 Usou-se o algoritmo Doomsday. Ver: https://pt.wikipedia.org/wiki/Algoritmo_Doomsday.

19 Informação obtida em : https://pt.wikipedia.org/wiki/Cálculo_da_Páscoa. Data da última atualização em 15 de Agosto de 2016.

dependem da adição de mais ou menos dias[20]. O segundo método permite alargar o feriado, tendo em atenção as possíveis "pontes", o método verifica a ocorrência de feriados durante a terça-feira ou a quinta-feira e associa nova data ao feriado.[21]

O segundo grupo de recursos auxiliares reúne um conjunto de métodos que permitiu tratar e recolher imagens, criar documentos, exportar documentos e imprimir. As duas classes: ImageAux.java e PrintAux.java agrupam uma grande parte daquelas incumbências.

A primeira possui métodos estáticos que recorrem a objetos Java (JFileChooser e FileDialog) para selecionar imagens, métodos para transformar ou redimensionar as imagens e métodos para obter imagens a partir de texto em base64, ou o ato inverso (as imagens são guardadas na base dados em formato de texto). A segunda classe possui métodos para chamar a janela (javax.swing.JDialog) de impressão (janela que é desenvolvida e criada na classe). Possui, também, métodos para transformar ficheiros PDF em imagem.

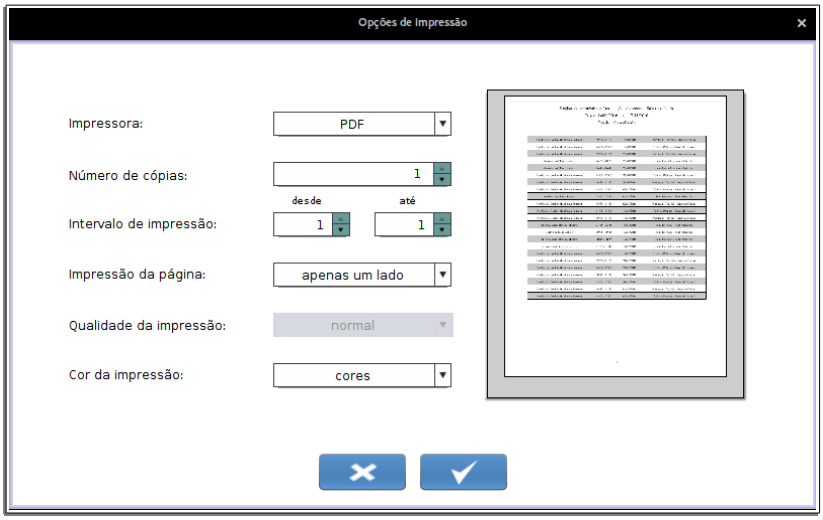


Ilustração 24: Ecrã de impressão.

Foram desenvolvidas, por fim, outras classes para criação de dados estatísticos a apresentação gráfica. O conjunto (*pacakge*) de classes: *Statistic*, permite a obtenção dos valores que depois são apresentados em soluções gráficas desenvolvidas pelo bloco de classes (package): *Graphics* (composto por: Barchart.java, Linechart.java e piechart.java), fazendo uso da biblioteca *jfreechart.jar.*

4.2. Objetos "CSV".

Foram criadas três classes para o tratamento do documento CSV, classes que permitiram transformar os dados do documento em objetos que a aplicação usa, são elas:

- *HandlingCSV.java*,
- *ElementsCSV.java*,
- *ObjectCSv.java*.

A primeira delas permitiu receber (encontrar) o ficheiro, a ligação pode ser efetuada através de ficheiro local, mas, também, através de URL sendo possível a indicação de um endereço de rede. Esta classe

20 Mais 60 dias para o corpo de deus, menos 2 dias para sexta feira santa e menos 47 dias para o Carnaval.
21 O código pode ser consultado em anexo.

cria uma cópia do documento em pasta local e transfere o conteúdo para esse documento. O documento, cópia é criado, ou recriado, quando não exista ou quando o documento original ser distinto à cópia existente.

A função desenvolvida que permite verificar documentos desiguais pode ser usada para automatizar o processo de criação das requisições "CSV", sendo uma mais valia em futuras atualizações do sistema.

O código desse método é o seguinte.

```
public static boolean isNew(String url, String nome) throws MalformedURLException, IOException {
	File file = new File(new File("").getAbsolutePath()+ System.getProperty("file.separator")
		+ "Resources" + System.getProperty("file.separator")
		+ "Download" + System.getProperty("file.separator") + nome);
	if (!file.exists()) {
		File diretoria = new File(new File("").getAbsolutePath()
			+ System.getProperty("file.separator")
			+ "Resources" + System.getProperty("file.separator")
			+ "Download");
		if (!diretoria.exists()) {
			diretoria.mkdirs();
		}
		file.createNewFile();
		return true;
	} else {
		File file2 = new File(new File("").getAbsolutePath()
			+ System.getProperty("file.separator")
			+ "Resources" + System.getProperty("file.separator")
			+ "Download" + System.getProperty("file.separator")
			+ nome + "_temp.csv");
		InputStream input = new URL(url).openStream();
		CSVReader reader;
		List<String[]> entradas;
		reader = new CSVReader(new InputStreamReader(input), ';');
		if (reader.verifyReader()) {
			entradas = reader.readAll();
			try (CSVWriter scv = new CSVWriter(new FileWriter(file2), ';')) {
				scv.writeAll(entradas);
				scv.flush();
			}
		}
		if (!FileUtils.contentEquals(file, file2)) {
			file2.delete();
			return true;
		} else {
			file2.delete();
			return false;
		}
	}
}
```

Através da análise do documento, os dados são utilizados para criar um objeto do tipo *ElementsCSV*, um objeto que possui as seguintes propriedades: dia da semana (inteiro), hora inicial (inteiro), minuto inicial (inteiro), hora inicial (inteiro), minuto final (inteiro), nome da pessoa (texto), código da pessoa (texto), nome da disciplina (texto), (código da disciplina), atividade (texto), código da atividade (texto), descrição do material (texto), código do material (texto) e código da turma (texto).

Aqueles são depois tratados pela classe *ObjectCSV.java* que constrói os seguintes objetos (propriedades de classe): pessoa, material, disciplina, dia da semana, hora inicial, hora final, atividade e turma. Esta última classe é, também, responsável pela criação da requisição.

Uma lista de requisições é, depois, processada e enviada para a classe *UpdateCSVonDB.java*, esta é uma classe composta pelas datas que compõem o intervalo de cálculo, pela lista de feriados e por uma lista de tempos de paragem. O método principal da classe – *update()* – implementa o seguinte diagrama de atividades.

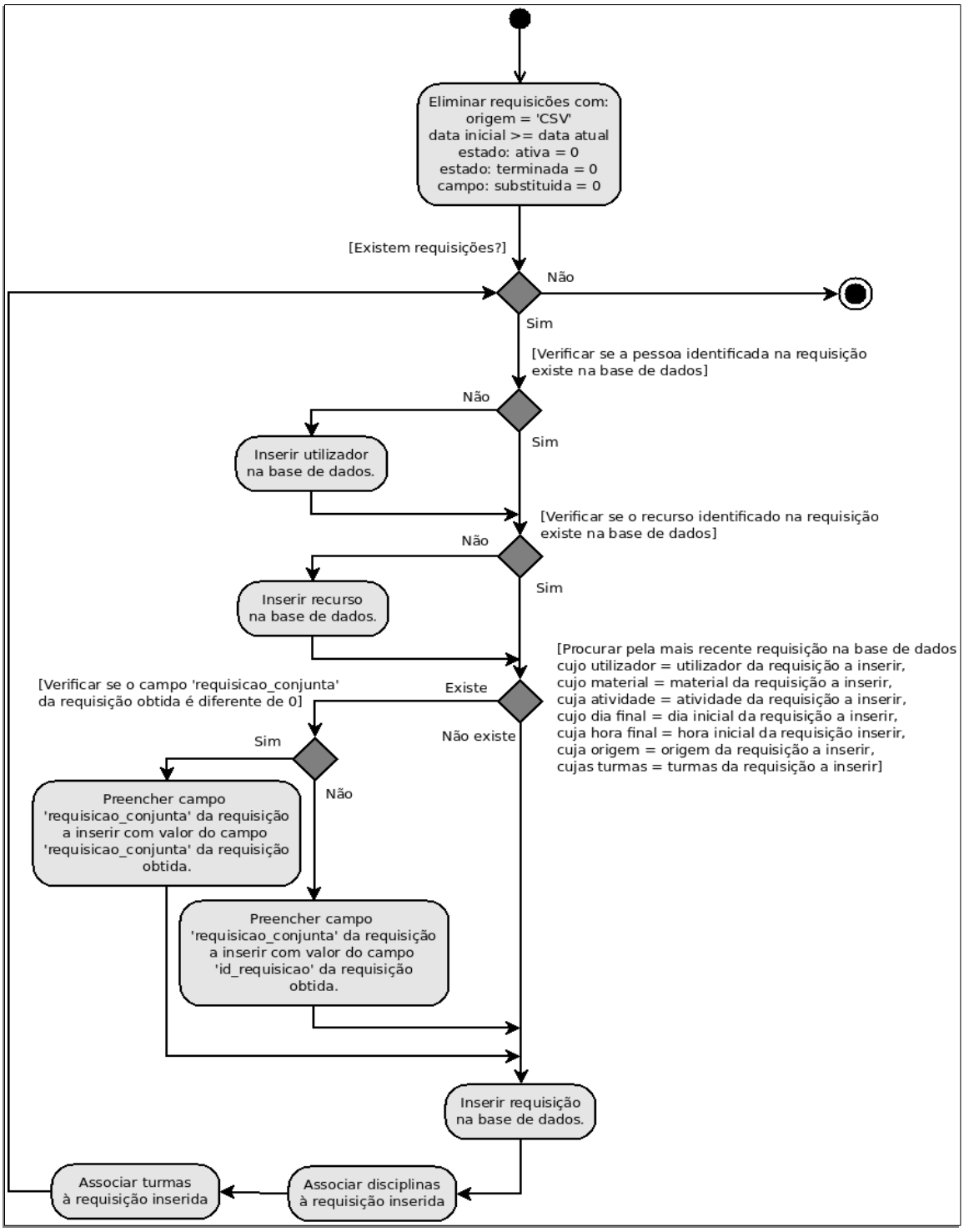


Ilustração 25: Diagrama de atividades "Registar requisições por CSV".

4.3. Base de dados.

O modelo entidade relacionamento, descrito em capítulos anteriores, foi utilizado na criação do *script* de base de dados através da aplicação *phpMyAdmin*. A base de dados criada está composta como indica o seguinte diagrama físico.

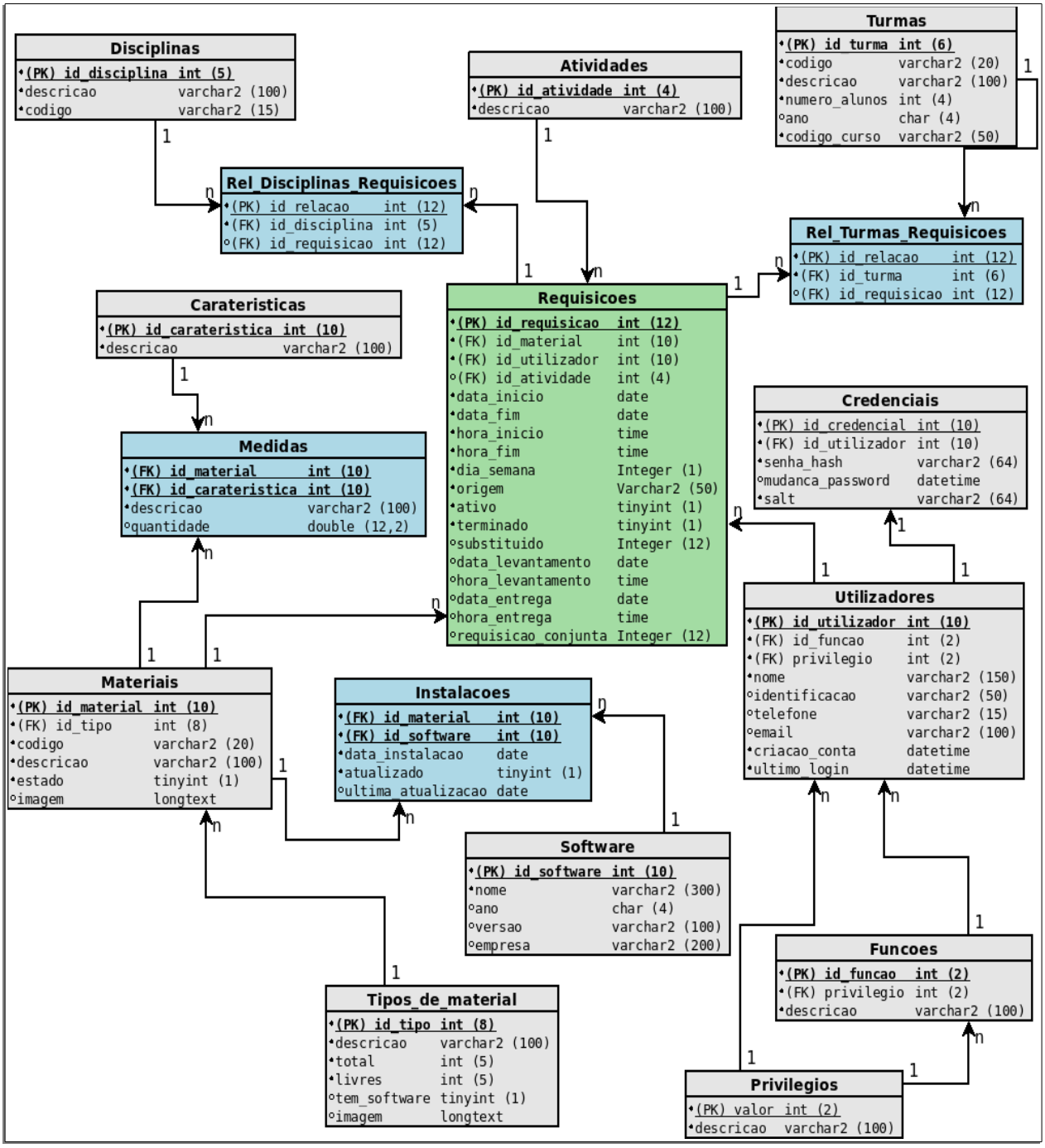


Ilustração 26: Diagrama físico de Base de Dados.

A base de dados está, ainda, composta por *triggers*, sendo o gatilho eficiente de atualização e controlo da integridade ou validade dos dados, libertando processamento da máquina local e lógica da aplicação. Neste

caso, o seu uso prendeu-se com a contagem sobre materiais inseridos e eliminados, atualizando o campo total na tabela Tipos_de_material, e a contagem sobre materiais emprestados, atualizando o campo livres da tabela referida.[22]

Todo processamento ou ações de registo, edição, eliminação e busca foram implementas numa classe criada para o fim, essa classe foi apelidada de *DataBase.java*. Foi construída com a intenção de separar tarefas, facilitar o processo de ligação à base de dados e o processo de recriação ou obtenção de objetos. Por cada objeto (requisição, utilizador, material, tipos de material, turma, caraterística, disciplina, software, atividade) esta classe implementa métodos de obtenção individual e coletiva (listas ou conjuntos), métodos de registo e atualização, métodos de transformação e obtenção de estados.

4.4. Interface.

Nos capítulos anteriores foram identificadas as interfaces que deveriam permitir a concretização dos casos de uso, neste capítulo abordaremos a sua implementação. O diagrama, em baixo, apresenta um conjunto de janelas, um conjunto que demonstra os movimentos sequências que levam os utilizador pelas diversas funcionalidades. A interface principal ou inicial é a janela base da aplicação, a partir dela o utilizador seleciona as demais, através de menu ou através de recursos colocados em pontos chave.

O conjunto de interfaces ligada a definições permite configurar alguns aspetos da aplicação: cores do sistema, linguagem do sistema, endereços de base de dados e do documento CSV, definir períodos de férias e feriados anuais. O conjunto ligado a movimentos compreende o grosso das atividades: registo e edição de utilizadores; registo, consulta, marcação e substituição de requisições; registo, edição e pesquisa de material.

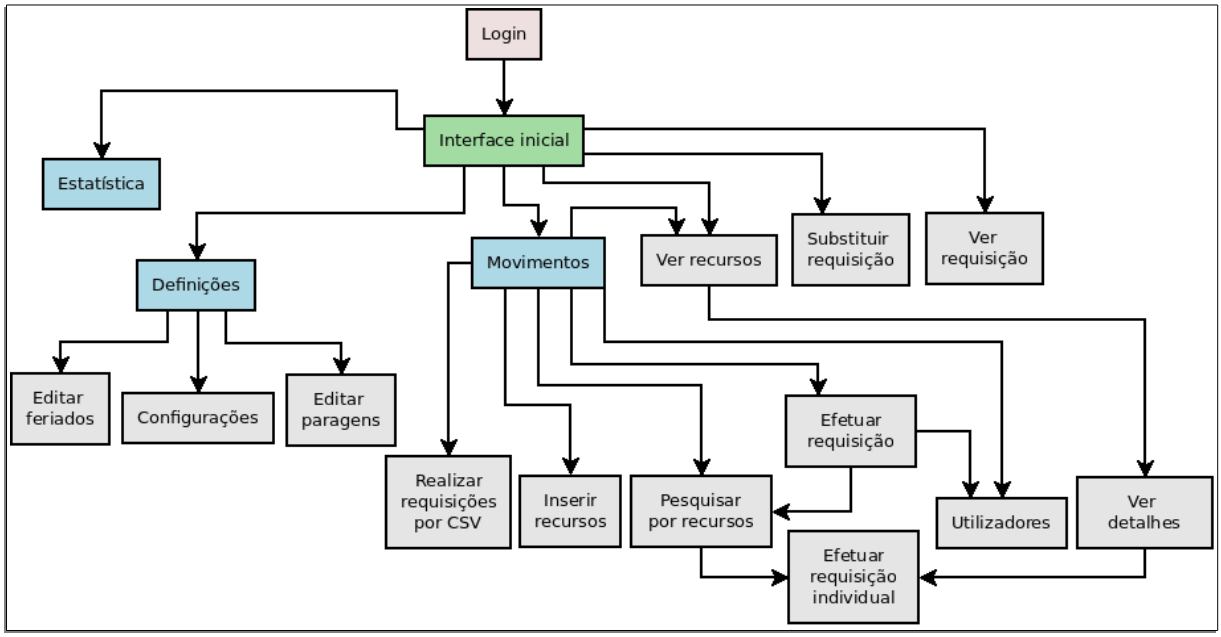


Ilustração 27: Diagrama de hierarquia.

22 O código desses *triggers* pode ser consultado em anexo.

4.4.1. Interface inicial (principal).

Desde a fase de planeamento houve a ideia de dividir a interface principal em tabela de reservas e tabela de devoluções, para isso, pensou-se no uso do instrumento *javax.swing.jSplitPane* como ponto central daquela separação. Esta interface (pode ser visualizada na ilustração 28) apresenta, por aquela razão, dois painéis sobrepostos, a divisão é efetuada por uma barra que pode ser manipulada através de ação direta ou através de botões colocados na zona superior da interface.

Às tabelas com as listas de requisições ou devoluções[23], juntaram-se painéis de informação. A ideia passou por consolidar a seleção e garantir uma informação completa ao funcionário. Com os painéis de informações colaram-se os botões de confirmação de levantamento / devolução de material e os botões de substituição e visualização de estado.

Acima dos painéis referidos, o utilizador da aplicação pode fazer uso de ferramentas de pesquisa que podem ser manipuladas pelo nome de utilizador, pela descrição do recurso ou pela hora de levantamento / devolução. Na barra superior aos painéis de separação, o utilizador pode fazer uso da seleção do tipo de pesquisa, pode fazer uso da escolha do tipo de material que deve ser listado nas tabelas abaixo, pode selecionar o intervalo de visualização das requisições e pode optar ou não pelo movimento automático das áreas de visualização da tabela, cujo cálculo de deslocação acompanha o tempo do sistema e a sua relação com os tempos de levantamento ou devolução[24].

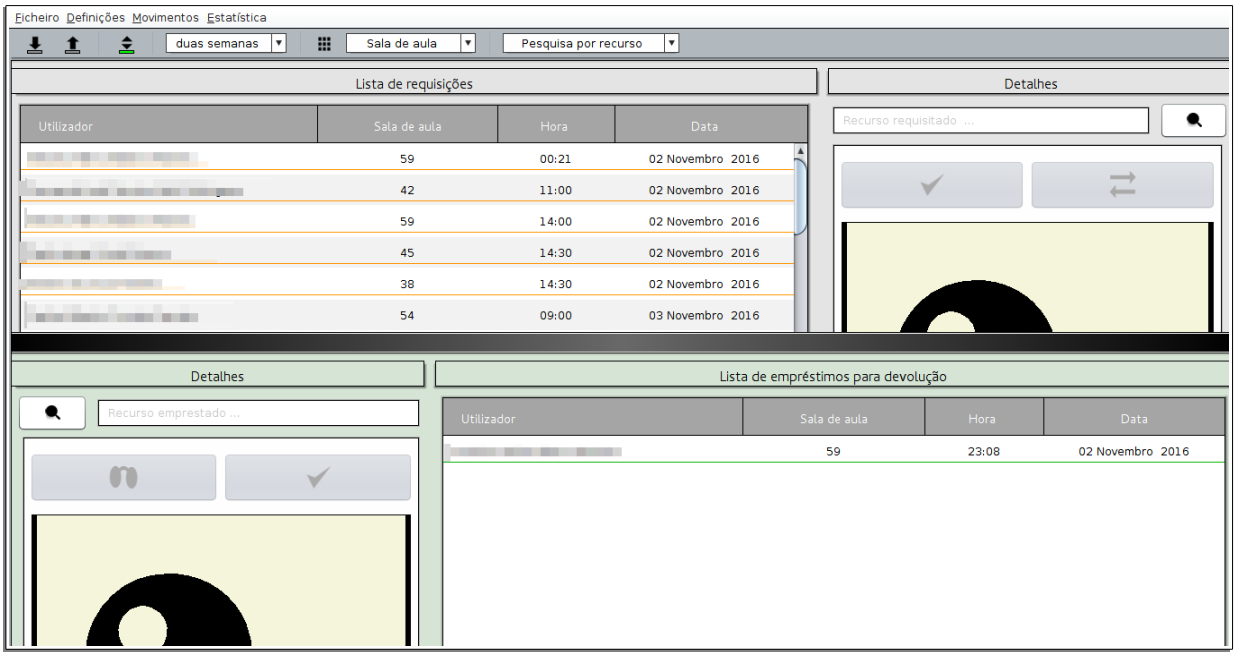


Ilustração 28: Interface principal.

23 Uma linha da tabela de requisições possui a seguinte informação: utilizador, material, hora de levantamento e data de levantamento; a linha da tabela de devoluções possui a seguinte: Utilizador, material, hora de devolução e data de devolução.

24 O deslocamento automático das tabelas é realizado por *threads* que verificam os tempos de requisição, o *scroll* é ajustado à primeira requisição cujo tempo (tempo de levantamento na tabela de reservas e tempo de devolução na tabela respetiva) se assemelhe ao tempo do sistema. A verificação é feita de segundo a segundo. De cada vez que o utilizador arraste a barra de *scroll*, os *threads* são interrompidos por 30 segundos.

As listas são ordenadas por data, hora e por valor alfabético do campo utilizador, as entradas dessas listas são pintadas de acordo com o estado da requisição: na tabela de reservas, requisições com tempo final ultrapassado são pintadas com uma linha vermelha (borda), requisições cujo tempo de levantamento é atual (concede-se menos 30 minutos ao tempo de levantamento) são pintadas com linha verde (só nesta situação é permitido confirmar a reserva), e requisições na mesma situação, mas com material não disponível, são pintadas com linha de cor preta; na tabela de devoluções utiliza-se o mesmo princípio, mas para tempos de devolução; assim, tempos de devolução (mais 30 minutos) ultrapassados originam o aparecimento de linha vermelha, tempos anteriores ou iguais aos tempos de devolução (no mesmo dia) originam o aparecimento da linha verde.

O menu foi criado nesta janela, está divido em quatro sub-menus: Ficheiro, Definições, Movimentos e Estatística. Este menu, como na maior parte das aplicações, permite ser o trampolim para outras interfaces, embora existam outros caminhos para aceder às interfaces “Alterar / substituir requisição”, “Ver requisição” e “Ver recursos”.

4.4.2. Movimentos.

É o grupo de interfaces identificadas no momento destinado à engenharia, às quais se acrescentou, ou não foram mencionadas, a interface “Ver requisição” e a interface “Realizar requisições por CSV”. A seguir descreve-se o funcionamento desenvolvido para cada uma delas.

- Interface “Utilizadores”, é uma interface que permite o registo de novos utilizadores e a sua edição. A interface usa uma tabela (*javax.swing.Jtable*) onde se listam todos os utilizadores. A ordenação está feita por ordem alfabética. Existe uma ferramenta de pesquisa que permite encontrar a pessoa pelo seu nome ou pela sua identificação (código ou correio eletrónico). A tabela permite, apenas, a seleção de um utilizador; quando tem valores selecionados, a interface funciona como edição, quando não existe seleção, a interface funciona como registo de novo utilizador.
- Interface “Realizar requisições por CSV”, é a interface que permite atualizar as requisições que existam através do documento daquele tipo. A interface possui os campos para registo de datas que identificam os semestres letivos que, depois, são listados num painel lateral. A aplicação guarda esses valores em ficheiro local. Quando selecionado o botão “Efetuar requisições”, o sistema atualiza as inscrições na base de dados.
- Interface “Inserir recursos”, interface criada para registar novos recursos. Possui opções para seleção do tipo de material e campos para inserção da descrição e código do material. O objeto Java (Jlabel) que é preenchido com a imagem do recurso é interativo e permite chamar o gestor de ficheiros para atualização da fotografia daquele.

  A lista de caraterísticas é atualizada de acordo com a mudança do tipo de material, existindo opções para o registo de novas ou caraterística distintas. Os itens daquela lista podem arrastar-se ou ser selecionados a partir dum quadro à parte, ação que permite associar as caraterísticas ao novo material.

- Interface “Pesquisar por recursos”, é uma interface que permite pesquisar recursos através das suas caraterísticas. Ela apresenta uma lista de recursos, normalmente, listando todos aqueles identificados com um determinado tipo de material; através dos quadros de caraterísticas e software (tipos de recursos que possuam software), a seleção ou arrasto dos itens para um quadro de relação permite filtrar os recursos, de acordo com a existência da associação ou da relação medida. Esta interface permite, ainda, chamar outra interface: “Efetuar requisição”.
- Interface “Efetuar requisição”, é a interface usada para registar reservas (a imagem 29 apresenta um *printscreen* daquele ecrã). Possui ferramentas para seleção de datas e tempos, campos para seleção de utilizadores e campos para seleção de materiais. Os utilizadores podem ser selecionados a partir de uma lista, essa lista é apresentada a partir de uma *combobox* que implementa um sistema de busca automática[25]. Os utilizadores podem, ainda, ser escolhidos através da introdução do correio eletrónico ou através da sua identificação.

  Depois de selecionado o tipo de material, a lista de materiais disponíveis é atualizada conforme a seleção dos períodos de requisição (o algoritmo que implementa o mecanismo de verificação dessa disponibilidade pode ser consultado no fim deste documento). A escolha do material pode ser múltipla ou particular (um recurso específico), caso a seleção for impessoal e não específica o sistema apresenta propostas aleatoriamente[26].

  Por cada recurso escolhido, o sistema vai apresentando objetos gráficos, com imagens ligadas ao material, tornando mais intuitiva a aplicação (o utilizador pode interatuar com esses objetos, sendo possível a alteração do recurso através de um menu (*popup*) sobre objeto que permite invocar nova interface para substituição do material). Quando os materiais se encontram

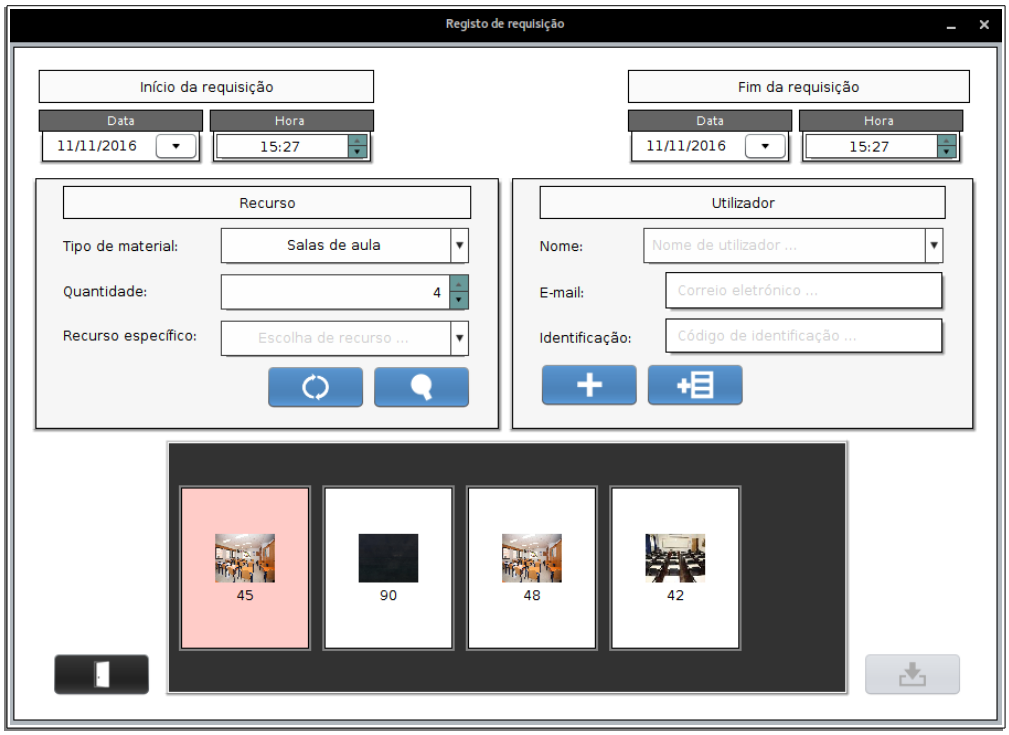


Ilustração 29: Interface "Efetuar requisição".

25 Em anexo é acrescentado o código que porta esta funcionalidade.
26 No futuro, a aplicação poderá analisar o histórico de empréstimos de cada utilizador e apresentar propostas ligadas a esses momentos.

emprestados, em situação de atraso, a aplicação possibilita a reserva, mas informa os utilizadores através de mensagem e utilizando a cor vermelha como fundo dos objetos.

- Interface “Efetuar requisição individual”, é uma interface que permite realizar reserva sobre um recurso específico. Este ecrã é chamado após seleção do recurso com a interface “Ver detalhes”, ou após a realização de pesquisa com a interface “Pesquisar por recurso”.

  Esta parte do programa está organizada com campos de seleção de datas, campos para seleção de utilizador e uma lista de informação com as requisições pendentes para aquele recurso. Esta interface, assim como a interface “Efetuar requisição,” permite o registo da requisição quando os tempos introduzidos não sobreponham os tempos de requisições anteriores.

- Interface “Ver recursos”, ecrã destinado à listagem dos recursos por tipo de material. Os materiais, aqui, são apresentados conforme o seu estado: todos, livres e ocupados (emprestados). Para isso, fez-se uso de abas separadores (javax.swing.JtabbedPane).

  O ecrã (ver figura em baixo) permite a consulta ou pesquisa de recursos pela sua designação ou por caraterísticas / software relacionada(o)s. Os recursos são visualizados sob a forma de botões (javax.swing.Jbutton); estes implementam ações que permitem, por exemplo, listar futuras requisições e invocar a interface “Ver detalhes”. Esta última é a interface central do recurso, ali se encontra a informação detalhada do recurso, ali se invoca a interface que permite realizar uma reserva, ali se editam os dados e se exportam os movimentos sobre o recurso para documentos Excel e impressão em papel.

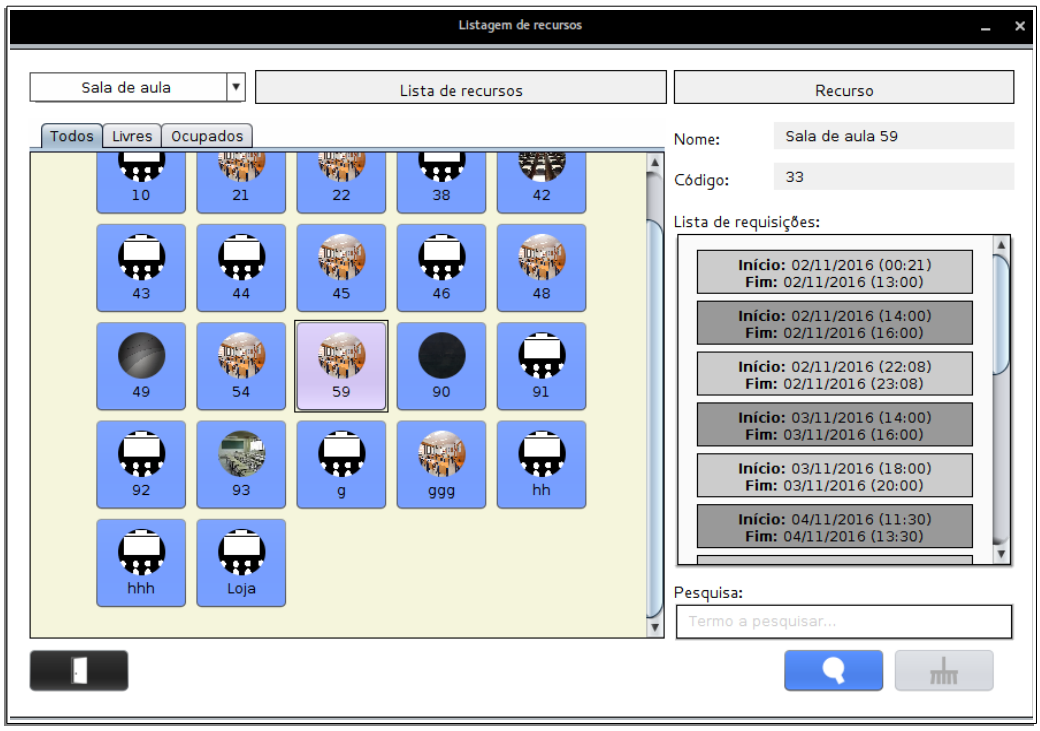


Ilustração 30: Interface "Ver recursos".

- Interfaces “Substituir requisição” (ilustração 31) e “Ver requisição”. Estas interfaces, na aplicação, não incorporam o menu que dá nome a este subcapítulo, no entanto, como representam ações sobre

objetos, achou-se que poderiam ser aqui incluídos. A primeira interface é usada após a seleção de uma reserva na tabela de requisições passivas, esta ação liberta o botão que a permite chamar. Este ecrã dispõe de campos para seleção de datas, tempos e recursos (javax.swing.JcomboBox), sendo esse último campo portador de métodos de pesquisa sobre a lista que o compõe.

A interface "Ver requisição" não tem função específica, é sobretudo uma janela informativa, com cronómetros sobre os tempos da requisição e tempos de atraso. A interface é chamada, ou permitida a sua chamada, quando selecionada uma reserva da tabela de devoluções. Esta interface permite também terminar uma requisição, após a confirmação da entrega do recurso.

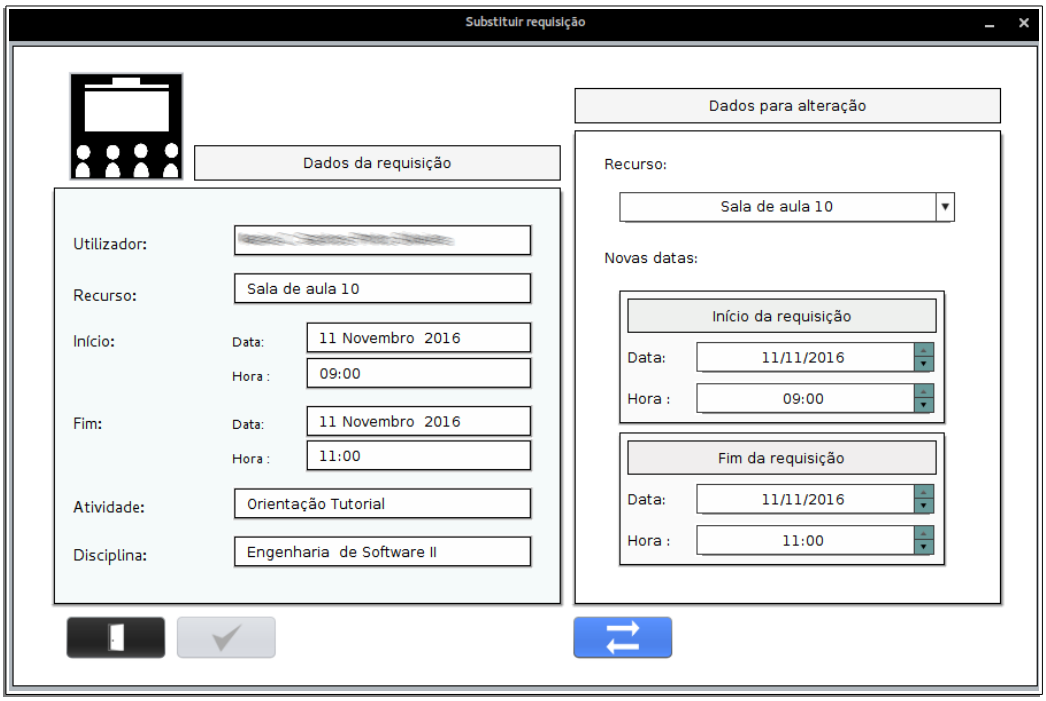


Ilustração 31: Interface: "Substituir requisição".

4.4.3. Definições.

As interfaces deste capítulo foram criadas para a configuração da aplicação, permitindo, também, a edição de tempos de paragem (férias) e feriados. Os valores escolhidos para férias e feriados são guardados em ficheiros locais e, para isso, foram criadas classes que permitiram a manipulação daqueles dados. Essas classes são (podem ser consultadas em anexo):

- FileHolidays.java.
- FileBreakPeriods.java

A janela, que permite a edição de feriados, lista um conjunto de feriados fixos portugueses que pode ser alterado através da edição de ficheiro (*Clavis/Recursos/feriados_lista.dat*) empacotado no ficheiro.jar; já os feriados selecionados (oficiais de cada ano), porque necessitam de ser editados, são guardados em pasta e ficheiro externo àquele pacote. Os feriados móveis são calculados pela aplicação e a passagem de ano implica

a atualização imediata das datas e do documento local. Esta operação é realizada na classe *HolidaysList.java*, o seu código é o seguinte:

```
public static void updateDynamicHolidays() {
	TimeDate.Date da = new TimeDate.Date();
	if ((da.getMonth() == 1) && (da.getDay() == 1)) {
		FileHolidays file = new FileHolidays();
		HolidaysList list = file.getHolidays();
		java.util.Map<Integer, TimeDate.Holiday> lista = new ConcurrentHashMap<>();
		int i = 0;
		for (TimeDate.Holiday day : list.getHolidays()) {
			if (day instanceof TimeDate.DinamicHoliday) {
				DinamicHoliday d = (DinamicHoliday) day;
				switch (d.getName()) {
					case "pascoa":
						d = new TimeDate.DinamicHoliday(new
						HolidaysList(da.getYear()).getEaster(), "pascoa");
						break;
					case "sexta_feira":
						d = new TimeDate.DinamicHoliday(new
						HolidaysList(da.getYear()).getGoodFriday(),
						"sexta_feira");
						break;
					case "corpo_cristo":
						d = new TimeDate.DinamicHoliday(new
						HolidaysList(da.getYear()).getCorpusChristi(),
						"corpo_cristo");
						break;
					case "carnaval":
						d = new TimeDate.DinamicHoliday(new
						HolidaysList(da.getYear()).getCarnival(), "carnaval");
						break;
					default:
						break;
				}
				lista.put(i, d);
			} else {
				lista.put(i, day);
			}
			i++;
		}
		list.setHolidays(new java.util.HashSet<>(new java.util.HashSet<>(lista.values())));
		file.saveHolidays(list);
	}
}
```

A interface “Definições” permite a mudança de alguns aspetos gráficos, como controlar a dimensão da barra separadora, alterar a cor predominante no sistema e alternar entre temas de cores da janela principal. O sistema está construído, também, de forma a permitir a alteração da linguagem de apresentação. As línguas preparadas para a aplicação foram: português, inglês, francês e espanhol.

Java permite o uso de simples ficheiros de texto, com a terminação *.properties:

texto_en.properties

texto_pt_PT.properties

texto_es_ES.properties

texto_fr_FR_properties

e que usam a sintaxe chave = valor, que, relacionados com o recurso Locale (identificação de língua e país), permitem à classe ResourceBundle realizar traduções.

```
ResourceBundle lingua = resourceBundle.getBundle(texto, locale);
String traducao = lingua.getString(chave);
```

As definições escolhidas são perenes (juntamente com a posição da aplicação no ecrã, o tamanho terminal da interface inicial, a localização do separador, endereços de ligação à base de dados e ao documento CSV, nomes de utilizador e senhas encriptadas, etc.) devido ao uso da biblioteca: java.util.prefs.Preferences. Este recurso permite guardar valores no registo, em sistemas Windows ou em ficheiros de configuração local na pasta de utilizador, em sistemas Linux / Unix.

4.5. Componentes gráficos.

Para auxiliar a construção gráfica das interfaces e aumentar a funcionalidade de alguns recursos Java, foi necessário, ao longo dos períodos de desenvolvimento, a criação de componentes personalizados, classes que são extensão de vários recursos gráficos da API *javax.swing, foram elas:*

- PersonalCombo.java,
- PersonalTextField.java,
- PersonalToggleButton.java,
- PersonalButton.java,
- PopUpMenu.java,
- MessagePane.java.

A primeira destas classe permitiu implementar o sistema de pesquisa e preenchimento automático do texto. A classe *PersonalCombo.java* permite a construção de um objeto do tipo *javax.swing.JComboBox*, que absorve um conjunto de eventos ligados ao teclado, rato e foco da aplicação, transformando-se num instrumento distinto e mais interativo. Esta classe acrescenta um texto de autoajuda ou informação; usando para tal o índice 0, a classe reescreve os métodos *gets* e *sets*, de forma a evitar a contagem ou uso daquele índice.

A classe *PersonalTextField.java* permitiu introduzir, da mesma forma, textos de informação (*placeholder*) em objetos do tipo *JTextField*, embora, por descuido e menos análise, a biblioteca *swingx*, já mencionada, permita recriar esta funcionalidade de uma forma bem mais simples[27].

```
String placeholder = "Introduza o seu nome";
javax.swing.JTextField texto = new javax.swing.JTextField();
PromptSupport.setPrompt(placeholder, texto);
```

A classe PopUpMenu.java utiliza o objeto javax.swing.JPopupMenu como referência, foi usada com frequência para criar menus locais em objetos, a classe foi formada por vários construtores que permitiram

27 Em: http://stackoverflow.com/questions/16213836/java-swing-jtextfield-set-placeholder.

criar vários tipos de mensagens e objetos / itens com eventos associados. Nesta classe foi implementado um método estático que permite a criação de um menu simples de cópia e colagem, este é passado para objetos com entrada de dados através do evento *java.awt.event.MouseListener*. O código que introduz a função pode ser consultado em anexo.

A classe MessagePane.java teve como principal função a criação de diálogos que pudessem incluir outros objetos / componentes, acrescentando funcionalidades. É, sobretudo, uma classe que permitiu a apresentação de mensagens (aviso, informação ou ação), funcionando, na maior parte das vezes, como a classe que desenvolve a comunicação do sistema com o utilizador.

Ilustração 32: Exemplo de uso da classe MessagePane.java.

4.6. Testes.

Existem quatro tipos essenciais de testes de software: testes unitários, testes de integração, testes de sistema e testes de aceitação. Os testes unitários permitem testar as condições de entrada e saída de dados de um sistema e, como cada teste unitário é independente dos demais, o programador pode testar cada módulo ou método sem afetar os demais. São testes escritos e realizados por programadores que conhecem a estrutura interna dos módulos.[28] Os testes de integração combinam as unidades ou módulos em componentes, resultando num sistema integrado que respondem aos requisitos do sistema, desde funcionalidades, desempenho e confiança na validade dos modelos.[29] Os testes de sistema são realizadas aquando da integração completa do sistema (software e hardware), são executados para identificar falhas ou repostas inesperadas do sistema (testes de caixa preta)[30], identificando e validando requisitos funcionais e não funcionais do sistema (desempenho, usabilidade, segurança, manutenção, disponibilidade e confiança). Os

28 Ver: https://pt.wikipedia.org/wiki/Teste_de_unidade, última atualização a 2 de Setembro de 2016.
29 Ver: https://pt.wikipedia.org/wiki/Teste_de_integração, última atualização a 3 de Março de 2013.
30 Ver: https://pt.wikipedia.org/wiki/Teste_de_caixa-preta, última atualização a 16 de Junho de 2014. São testes em que a estrutura interna do sistema é desconhecida, apenas são testadas as interfaces do sistema e as funcionalidades que prestam ao todo.

testes de aceitação têm por objetivo verificar a relação dos requisitos originais com as funcionalidades criadas , procurando responder às reais necessidades dos utilizadores do sistema. Neste tipo de testes, a participação dos utilizadores no planeamento e na realização é uma condição obrigatória.

Neste projeto, procurou-se ter alguma atenção com a escrita de testes unitários e testes de integração e, assim, à medida que o trabalho de desenvolvimento progredia, os testes unitários e alguns testes de integração eram escritos sobre classes e componentes. A biblioteca Junit foi a ferramenta utilizada para a sua construção, ela fornece um conjunto de métodos que permitem: comparar dados esperados com dados obtidos (*assertion methods*), comprovar comportamentos através do lançamento de exceções e controlo de tempos com testes de desempenho (a erro é definido pelo limite de tempo). Existe a possibilidade de criação de suites (blocos) de testes que podem ser automatizados, podendo acompanhar todas as etapas de desenvolvimento, corrigindo falhas indesejadas e não pensadas pelo programador.

Para este relatório, apresenta-se e descreve-se um exemplo de teste unitário e um exemplo de teste de integração. O teste unitário, apresentado no código em baixo, é usado para validar o método *dateBefore*() da classe *TimeDate.Date*. Este teste utiliza parâmetros de entrada e saída, a etiqueta "@Parameters" identifica os valores e a etiqueta "@RunWith(Parameterized.class)" informa o compilador que o teste deve ter em conta a situação. O primeiro dos argumentos, passado para a variável "input" através da referência à etiqueta "@Parameter", vai servir como valor de entrada na função *dateBefore*(input); o valor esperado passa a ser identificado pela variável seguinte (String). O método a testar é identificado pela etiqueta "@Test".

Ao correr o teste, caso existam falhas elas são listadas dentro do ciclo *for*, o resultado é apresentado com o recurso ao método *wasSuccessful*() que retorna um valor boleano.[31]

```
@RunWith(Parameterized.class)
public class TestDateBefore {
	TimeDate.Date dat = new TimeDate.Date(1, 10, 2016);
	@Parameters
	public static Collection<Object[]> data() {
		return Arrays.asList(new Object[][] {
			{366, "01/10/2015"},
			{731, "01/10/2014"},
			{16802, "01/10/1970"}
		});
	}
	@Parameter // por defeito 0
	public int input; // Não pode ser privado
	@Parameter(value = 1)
	public String esperado; // Não pode ser privado
	@Test
	public void testDateBefore() {
		TimeDate.Date dat2 = new TimeDate.Date(1, 10, 2015);
		assertEquals(esperado, this.dat.dateBefore(input).toString());
	}
	public static void main(String[] args) {
		Result result = JUnitCore.runClasses(TestDateBefore.class);
		result.getFailures().stream().forEach((f) -> {
			System.out.println(f.toString());
		});
		System.out.println(result.wasSuccessful());
```

31 A resposta foi verdadeira ao correr este teste. Os valores de teste foram obtidos a partir da plataforma Web: http://pt.calcuworld.com/calendarios/calculadora-de-tempo-entre-duas-datas/.

```
        }
    }
```

O segundo exemplo é um teste de integração, serve sobretudo para testar o sistema quando faz uso da base de dados. O código cria uma requisição, um utilizador e um recurso, os três objetos são registados na base de dados, o registo da requisição ocorre em último lugar pois só pode ser realizada se o utilizador e o recurso existirem. O método que insere a requisição na base de dados retorna 1 se for bem sucedido, este é um dos aspetos a testar, os outros aspetos testados relacionam-se com os métodos que modificam o estado do empréstimo. O método *assertTrue* da FrameWork Junit permite controlar as respostas.

```
    public class TestRequests {
        String url = "endereço";
        DataBase.DataBase db;
        @Test
        public void testActionRequests() throws ParseException{
            TimeDate.Date date = new TimeDate.Date();
            TimeDate.Date date2 = new TimeDate.Date();
            TimeDate.Time tinicio = new TimeDate.Time();
            TimeDate.Time tfim = new TimeDate.Time().addSeconds(60*10);
            db = new DataBase.DataBase(url);
            java.util.List<Keys.Function> f = db.getFunctions();
            Keys.Person p = new Keys.Person("José Mário Pereirinha", "1010234", f.get(2), 3);
            db.insertPerson(p);
            Keys.TypeOfMaterial tipo = db.getTypeOfMaterial(2);
            Keys.Material m = new Keys.Material(tipo, "Len71000", "Lenovo Yoga 770", false);
            db.insertMaterial(m);
             m.setId(db.getMaterialID(m));
            Keys.Request re = new Keys.Request(date, date2, new TimeDate.WeekDay(date), tinicio,
            tfim, p, m, "local");
            int val = db.insertRequest(m, p, "", new java.util.ArrayList<>(), new java.util.ArrayList<>(),
            date, date2, tinicio, tfim);
            Assert.assertTrue(val > 0); // teste criação val = 1, sucesso.
            Assert.assertTrue(db.changeRequestActiveState(re));
            Assert.assertTrue(db.getMaterial(m.getCodeOfMaterial()).isLoaned()); // material emprestado
            Assert.assertTrue(db.changeRequestTerminateState(re));
            Assert.assertTrue(!db.getMaterial(m.getCodeOfMaterial()).isLoaned());
            db.deleteRequest(re, p, m);
            db.deletePerson(p);
            db.deleteMaterial(m);
            db.close();
        }
        public static void main(String[] args) {
            Result result = JUnitCore.runClasses(TestDateBefore.class);
            result.getFailures().stream().forEach((f) -> {
                System.out.println(f.toString());
            });
            System.out.println(result.wasSuccessful());
        }
}
```

5. Conclusão.

Como aprendido durante o percurso escolar, a conclusão de um relatório deve sempre ter em conta a avaliação dos objetivos iniciais. Uma parte daqueles objetivos foi alcançada, uma outra, não. A aplicação, por exemplo, permite o registo e a edição de utilizadores, permite, também, a inserção e a edição de materiais. A aplicação possui áreas que dispõem de funcionalidades para a realização de requisições, substituição ou alteração daquelas e a mudança de estados (levantamento e devolução de materiais). A aplicação implementa objetos de pesquisa sobre recursos, contendo filtros sobre nomes ou expressões, aos quais acrescenta filtros que identificam relações, ou associações, com caraterísticas e software. A aplicação porta, também, objetos de configuração que viabilizam a edição de tempos letivos, períodos de paragem e feriados. A aplicação propicia componentes estatísticas, com elaboração de análises gráficas (é existente, mas pouco trabalhada); faculta, ainda, métodos de exportação e impressão (sobre a atividade registada sobre cada recurso).

De entre os objetivos não concretizados destaca-se o desenvolvimento de zonas administrativas; uma parte do código proporciona alguns objetos que respondem à distinção de utilizadores e seus privilégios, no entanto, não existem ferramentas para validação de utilizadores; o *login* não é, ainda, exigido e a presença de utilizadores com diferentes níveis de privilégios não é assinalada. À aplicação faltou a atribuição de espaço e funcionalidades ligadas aos utilizadores, faltou a existência de um histórico e listagem de atividades daqueles, faltaram formas e fórmulas de pesquisa sobre momentos de reserva ligadas a utilizadores e, por fim, faltaram funções de registo e alteração de reservas através da identificação dos mesmos.

Por alguma inexperiência e falta de planeamento, a componente visual ou de design da aplicação é algo confusa, a divisão da interface inicial não resulta como o esperado e uma representação mais estática das áreas, talvez, resultasse melhor. É que, a simplicidade é, quase naturalmente, a melhor das opções quando se idealiza uma aplicação intuitiva e de fácil aprendizagem. A facilidade de uso foi sempre uma das intenções na construção das interfaces gráficas e, para isso, tentou-se a criação de objetos gráficos interativos e móveis (arrasto) com o objetivo de facilitar o uso da aplicação, talvez, negligenciando a consolidação e robustez das funcionalidades a que se destinavam.

Faltou, também, a intervenção de terceiros, dos utilizadores finais da aplicação, das suas análises e críticas. Os testes à aplicação deveriam ter sido escritos para a presença daqueles, se não foi possível a criação de testes de sistema e aceitação, deveria, pelo menos, ter sido tomada em conta a opinião de pessoas interessadas ou de agentes do sistema (*stakeholders*).

Apesar dos contratempos, este projeto é viável, no futuro poder-se-ia pensar em acrescentar processos que façam uso do cartão de identificação na instituição para automatizar o processo de validação e recolha de recursos, sem a intervenção ou uso da interface. A aplicação poderia ter processos autónomos de configuração aos tempos de paragem, bastando associar fontes ou documentos de análise existentes na plataforma escolar e que pudessem ser obtidos pelo sistema. A aplicação poderia ter processos de busca e procura associados ao utilizador, a análise da componente histórica daquele, desde recursos a caraterísticas, até tempos de utilização, numa relação de fatores que poderia trazer um contributo à pesquisa, e gerar, até, momentos de estudo.

A atualização da base de dados da aplicação com fontes provenientes da plataforma escolar seria importante para garantir a integridade da informação, a unidade dos dados e a completude da informação, através da inserção / validação de utilizadores (professores, alunos e funcionários), permitindo a associação de turmas, cursos, disciplinas e atividades. A aplicação poderia, mesmo, servir como motor de informações, alertando (por correio eletrónico ou mensagem de telemóvel) para a alteração das atividades (por exemplo, a substituição de qualquer atividade letiva) ou a realização de outras não previstas (caso de frequências, exames, reuniões e outros eventos). A própria aplicação poderia ser estendida através de uma plataforma ou aplicação Web, tornando o processo de requisição mais pessoal, mais útil e menos centralizado.

Com o fim à vista, este projeto foi um ponto consolidante da minha aprendizagem, permitiu-me obter e aprofundar conhecimentos sobre as tecnologias usadas, mas, sobretudo, serviu como fator de alerta... o planeamento é o momento crucial de qualquer obra de engenharia; em software, a análise de requisitos, a definição, o relacionamento e a escolha da estrutura de dados, a seleção e a expansão do conhecimento pelas tecnologias selecionadas são imensamente importantes. Ao não ser assim, os remendos são uma constante.

6. Bibliografia.

Apache, Apache Poi, visto a 19 de Setembro de 2016, disponível em: https://poi.apache.org/spreadsheet/quick-guide.html.

Calcuword.com, *Calculadoras*, visto a 02 de Novembro de 2016, disponível em: http://pt.calcuworld.com/calendarios/calculadora-de-tempo-entre-duas-datas/.

Deitel, Paul, Deitel, Harvey (2015) – *Java, How to program*, 10ª Edição, Deitel & Associates, Inc, Pearson Education,Inc, Estados Unidos da América, New Jersey.

DesenvolvimentoAgil.com.br, *Extreme Programming*, visto a 01 de Novembro de 2016, disponível em: http://www.desenvolvimentoagil.com.br/xp/

Larman, Craig (2004) – Applying UML and Patterns:An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3ª edição, Addison Wesley Professional.

Loeliger, J., McCullough M. (2012) – *Version Control With Git*, 2ª edição, O' Reiley Media, Inc, Estados Unidos da América, Sebastopol.

Martin, Robert C. (2008) – *Clean Code, A handbook of Agile Software Craftsmanship*, Pearson Education, Inc, Estado Unidos da América, Stoughton, Massachusetts.

Mysql, Data Type Storage Requirements, visto a 05 de Novembro de 2016, disponível em: http://dev.mysql.com/doc/refman/5.7/en/storage-requirements.html.

StackOverflow, Apache PDFBox convert pdf to images, visto em Setembro de 2016, disponível em: http://stackoverflow.com/questions/23326562/apache-pdfbox-convert-pdf-to-images.

Stackoverflow, *Copying to Clipboard in Java*, visto a 12 de agosto de 2016, disponível em: http://stackoverflow.com/questions/3591945/copying-to-clipboard-in-java.

StackOverflow, *Print a PDF file using PrinterJob in Java*, visto em Setembro de 2016, disponível em: http://stackoverflow.com/questions/16293859/print-a-pdf-file-using-printerjob-in-java.

Stackoverflow, *Simple java AES encrypt/decrypt example*, visto em Setembro de 2016, disponível em: http://stackoverflow.com/questions/15554296/simple-java-aes-encrypt-decrypt-example.

Teles, Vinícius Manhães (2005) – Um estudo de caso da adoção das práticas e valores Extreme Programming, dissertação de mestrado da Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Tomás, Mário Rui Sampaio (2009) – *Métodos ágeis: características, pontos fortes e fracos e possibilidades de aplicação*, Faculdade de ciências e Tecnologia da Universidade Nova de Lisboa, Lisboa.

Tutorialspoint, Jfree Chart *Tutorial*, visto em Setembro de 2016, disponível em: http://www.tutorialspoint.com/jfreechart/.

Tutorialspoint, JUnit Tutorial, visto em Outubro e Novembro de 2016, disponível em: https://www.tutorialspoint.com/junit/index.htm.

Tutorialspoint, *MySQL Tutorial*, visto entre Agosto e Setembro de 2016, disponível em: http://www.tutorialspoint.com/mysql/mysql_tutorial.pdf.

Oracle, *TransferHandler Class*, visto em Agosto de 2016, disponível em: https://docs.oracle.com/javase/tutorial/uiswing/dnd/transferhandler.html.

Wikipédia, *Cálculo da Páscoa*, última atualização a 15 de Agosto de 2016, disponível em: https://pt.wikipedia.org/wiki/Cálculo_da_Páscoa.

Wikipédia, *Diagrama de contexto de sistema*, última atualização a 21 de Fevereiro de 2016, disponível em: https://es.wikipedia.org/wiki/Diagrama_de_contexto_de_sistema.

Wikipédia, Doomsday, cálculo do dia da semana, última atualização a 17 de Dezembro de 2013, em: https://es.wikibooks.org/wiki/Algoritmia/Algoritmo_para_calcular_el_día_de_la_semana.

Wikipédia, MySQL, última atualização a 03 de Novembro de 2016, disponível em: https://en.wikipedia.org/wiki/MySQL.

Wikipédia, *Teste de caixa-preta*, última atualização a 16 de Junho de 2014, disponível em: https://pt.wikipedia.org/wiki/Teste_de_caixa-preta.

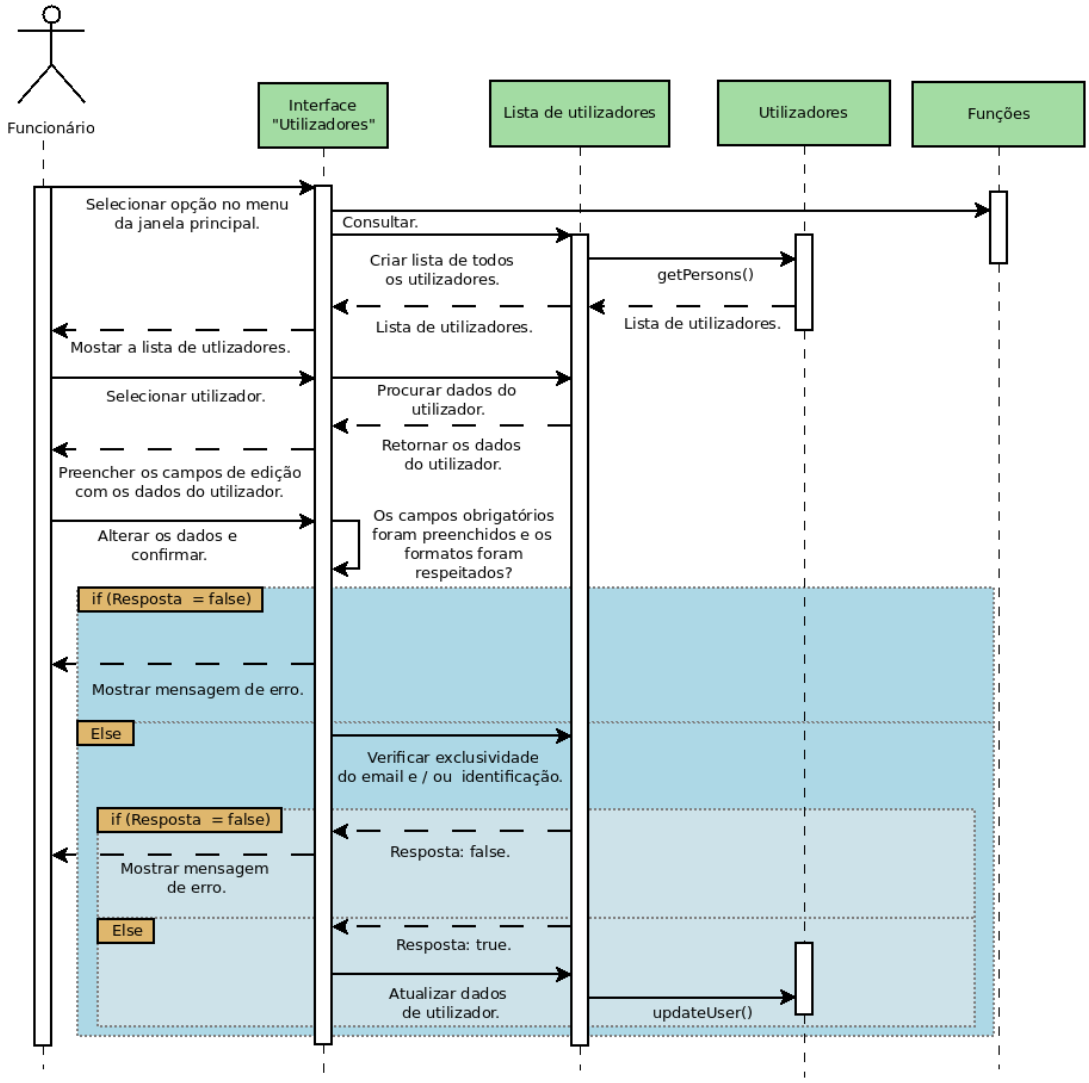
Wikipédia, *Teste de integração*, última atualização a 03 de Março de 2013, disponível em: https://pt.wikipedia.org/wiki/Teste_de_integração.

Wikipédia, *Teste de unidade*, última atualização a 02 de Setembro de 2016, disponível em: https://pt.wikipedia.org/wiki/Teste_de_unidade.

7. Anexos.

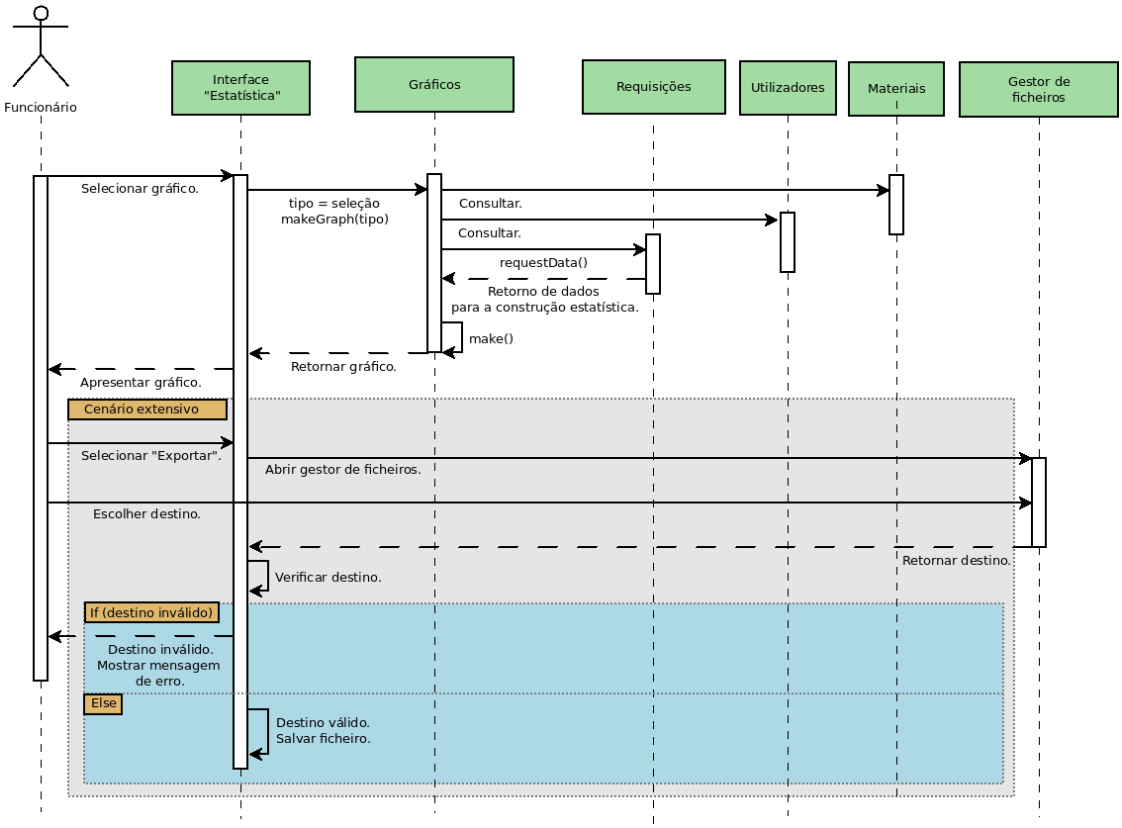
7.1. Casos de uso.

Nome	Editar utilizador “requisitante”
Objetivo	Editar / alterar dados da pessoa / requisitante no sistema.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Média
Fluxo principal	**1.** O ator seleciona a opção “Utilizadores” no menu da aplicação. **2.** O sistema apresenta uma lista de utilizadores. (2A) **3.** O ator seleciona o indivíduo da lista de utilizadores. **4.** O sistema preenche os seguintes campos de um formulário: nome, correio eletrónico, telefone, função, identificação na instituição, com os dados do indivíduo selecionado. **5.** O ator faz as alterações e confirma os novos dados. (5A) (5B) (5C) **6.** O sistema regista os novos valores no sistema.
Fluxos alternativos	**5A** **a)** O ator não preenche os campos ‘nome’, ‘correio eletrónico’ e / ou ‘identificação na instituição’. **b)** O sistema mostra mensagem de erro: “Preencha os campos ‘nome’ e, pelo menos, um destes campos: ‘correio eletrónico’ ou ‘identificação’”.
	5B **a)** O ator preenche erradamente o campo “correio eletrónico” e “telefone”. O campo correio eletrónico deve corresponder à seguinte forma: “[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}$”, o campo telefone à seguinte: “^(([+]?(\\d{3}))|(([+]?[(](\\d{3})[)])))?[- .]?((2\\d{2}[. -]?\\d{6})|(2\\d{2}[. -]?\\d{3}[. -]?\\d{3})|(9[1236]\\d{7})|(9[1236][. -]?\\d{3}[. -]?\\d{4})|(9[1236][. -]?\\d{4}[. -]?\\d{3})|(9[1236][. -]?\\d{3}[. -]?\\d{2}[. -]?\\d{2}))”. O ator confirma os dados. **b)** O sistema mostra mensagem de erro: “Os campos ‘correio eletrónico’ e / ou ‘telefone’ estão num formato incorreto”.
	3C **a)** O ator preenche os campo com um endereço de correio eletrónico ou uma identificação já existente na base de dados. **b)** O sistema mostra a mensagem de erro: “Verifique validade e exclusividade dos dados de identificação”.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema apresenta a mensagem do erro ocorrido: “Não existe ligação à base de dados”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar a alteração dos dados na lista de utilizadores. **2.** Verificar se a omissão de campos obrigatórios e formatos errados impedem o registo de utilizadores.



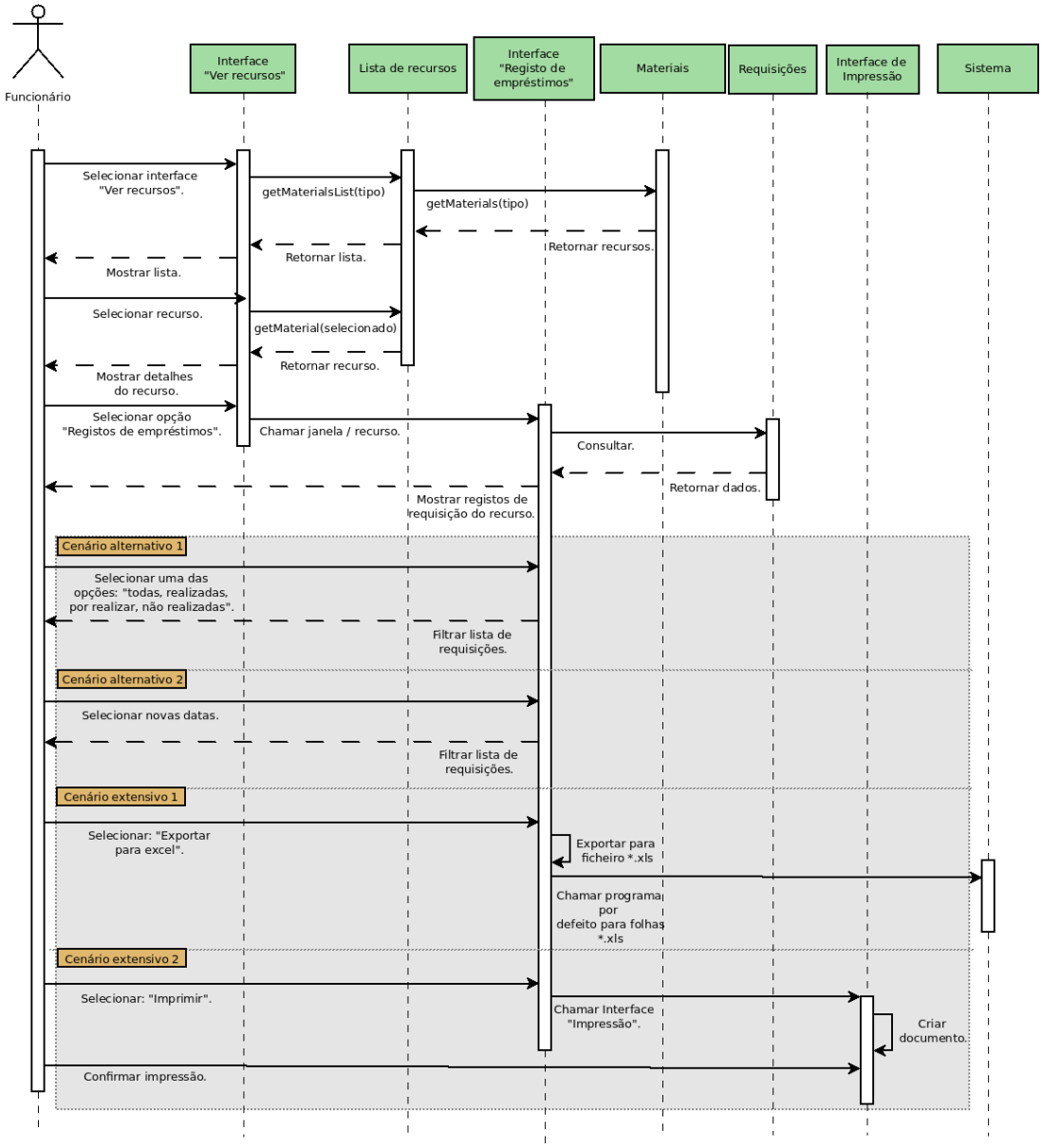
Nome	Consultar / exportar dados estatísticos
Objetivo	Fornecer dados estatísticos que podem ser consultados através de apresentação gráfica, estes podem ser exportados para ficheiro de imagem (*.png, *.jpeg).
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Baixa
Fluxo principal	**1.** O ator dirige-se à opção “Estatística” do menu da janela principal. **2.** O sistema apresenta as várias opções de consulta. (2A) **3.** O ator seleciona a opção pretendida. **4.** O sistema apresenta nova janela com o gráfico desejado. (4A)
Fluxos alternativos	**4A** **a)** Na janela aberta,o ator seleciona a opção “Exportar para imagem”. **b)** O sistema abre o recurso “Gestor de ficheiros”. **c)** O ator escolhe a localização pretendida para o ficheiro. (7A) **d)** O sistema exporta o gráfico para imagem e guarda o ficheiro no destino escolhido.
	7A **a)** O ator introduz um destino / pasta inexistente. **b)** O sistema alerta o ator para a invalidade do destino.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema não apresenta opções de escolha.

Pós-condição	Não existe.
Casos de teste	**1.** Verificar se os dados apresentados no gráfico correspondem à opção desejada. **2.** Verificar se a imagem é guardada na localização pretendida.

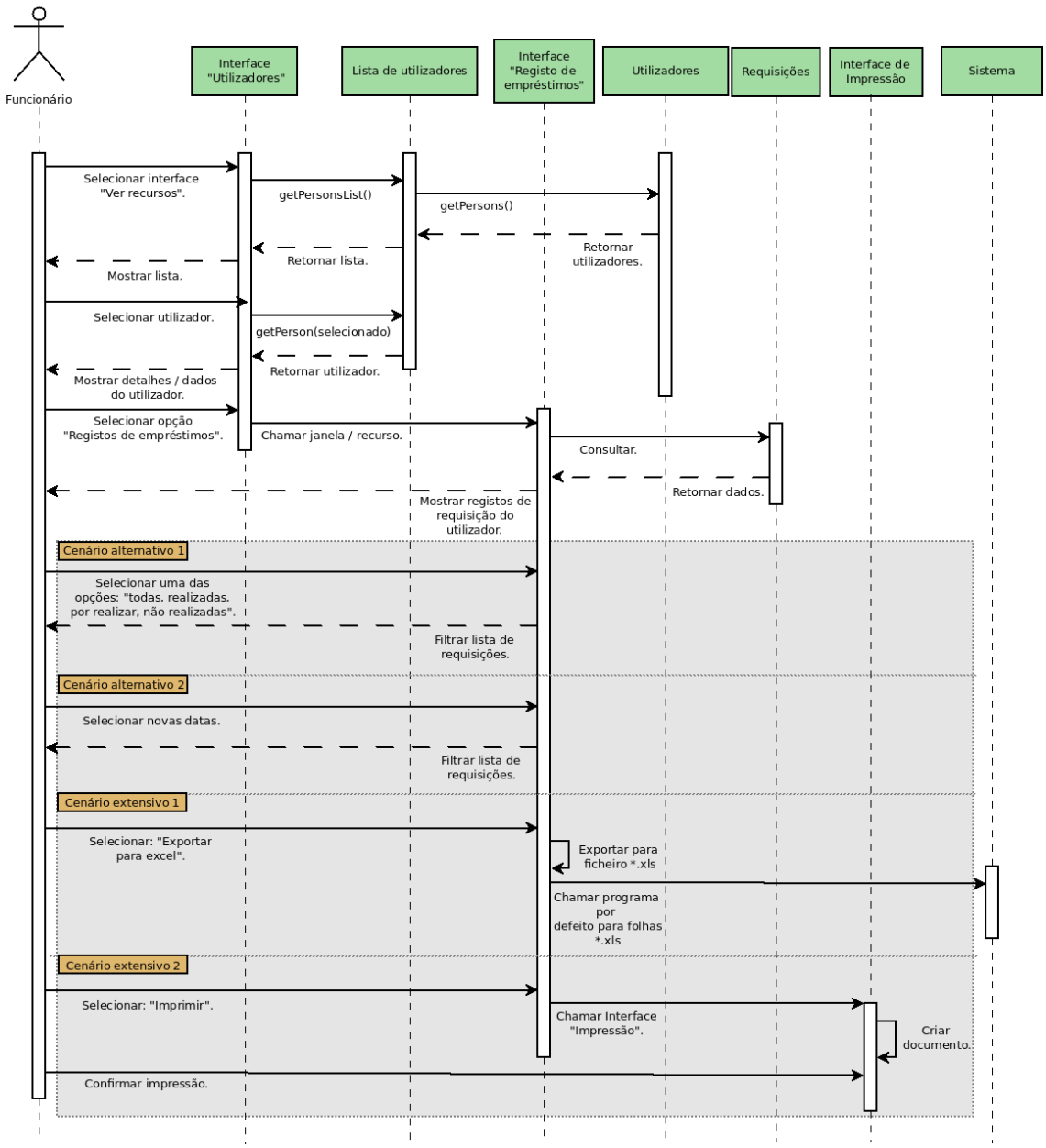


Nome	Verificar atividades por recurso
Objetivo	Consultar e exportar /imprimir registos de atividade por cada recurso.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Baixa
Fluxo principal	**1.** O ator dirige-se à opção “Ver recursos” do menu da janela principal. **2.** O sistema apresenta a lista de recursos por cada tipo de material. (2A) **3.** O ator seleciona o recurso pretendido. **4.** O sistema apresenta nova janela com detalhes do recurso e opções para outras operações. **5.** O ator seleciona a opção “Registos de empréstimo”. **6.** O sistema mostra todos os registos de requisições (realizados, não realizados e por realizar) associados ao recurso. (6A) (6B) (6C) (6D)
Fluxos alternativos	**6A** **a)** O ator seleciona uma das seguintes opções (“todas”, “realizadas”, “não realizadas”, “por realizar”). **b)** O sistema filtra a lista de requisições conforme o estado daquelas e o intervalo temporal.
	6B **a)** O ator seleciona uma nova data inicial e / ou uma nova data final. **b)** O sistema filtra a lista de requisições conforme o estado e o intervalo temporal.
	6C **a)** O ator seleciona a opção “Exportar para Excel”. **b)** O sistema exporta as requisições apresentadas para um ficheiro *.xls e abre o

	programa do sistema operativo por defeito.
	6D **a)** O ator seleciona a opção “Imprimir”. **b)** O sistema apresenta janela de impressão com opções de impressão e escolha de impressora. **c)** O ator confirma impressão. (6DcA)
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema não permite a abertura da janela com a lista de recursos.
	6DcA **a)** Não existe nenhuma impressora instalada. O sistema apresenta a mensagem de erro “Não existem impressoras instaladas”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar se as requisições apresentadas correspondem ao estado definido e ao intervalo temporal. **2.** Verificar se a exportação é bem sucedida. **3.** Verificar se a aplicação reconhece as impressoras instaladas no sistema operativo e respetivas propriedades. **4.** Verificar se a impressão é bem sucedida.

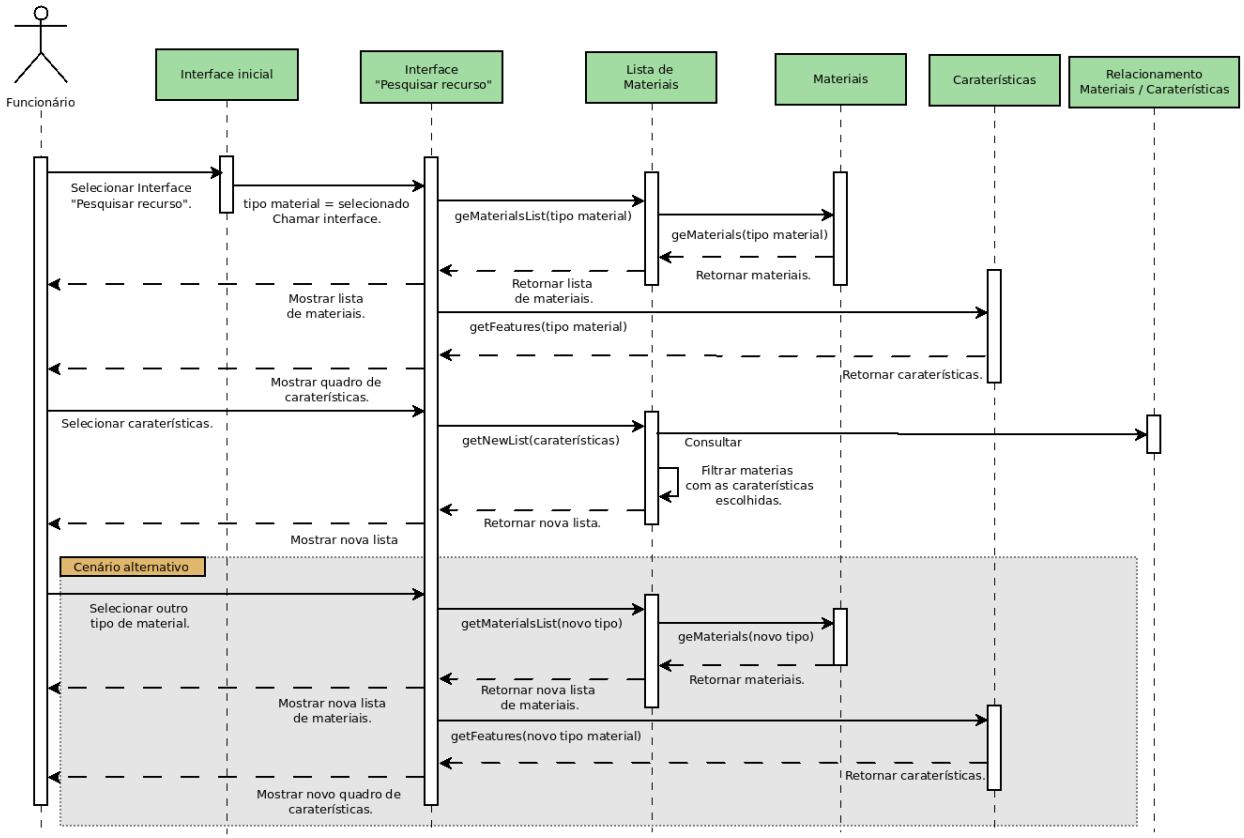


Nome	Verificar atividades por utilizador
Objetivo	Consulta e exportar /imprimir registos de atividade por cada utilizador.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Baixa
Fluxo principal	**1.** O ator dirige-se à opção “Utilizadores” do menu da janela principal. **2.** O sistema apresenta a janela com a lista de utilizadores. **3.** O ator seleciona o utilizador. **4.** O sistema apresenta os dados principais do utilizador e ativa a opção “Registos e relatórios”. **5.** O ator seleciona a opção “Registos e relatórios”. **6.** O sistema mostra todos os registos de requisições (realizados, não realizados e por realizar) associados ao utilizador. (6A) (6B) (6C) (6D)
Fluxos alternativos	**6A** **a)** O ator seleciona uma das seguintes opções (“todas”, “realizadas”, “não realizadas”, “por realizar”). **b)** O sistema filtra a lista de requisições conforme o estado daquelas e o intervalo temporal.
	6B **a)** O ator seleciona uma nova data inicial e / ou uma nova data final. **b)** O sistema filtra a lista de requisições conforme o estado e o intervalo temporal.
	6C **a)** O ator seleciona a opção “Exportar para Excel”. **b)** O sistema exporta as requisições apresentadas para um ficheiro *.xls e abre o programa do sistema operativo por defeito.
	6D **a)** O ator seleciona a opção “Imprimir”. **b)** O sistema apresenta janela de impressão com opções de impressão e escolha de impressora. **c)** O ator confirma impressão. (6DcA)
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema a mostra mensagem: “ Não existe ligação à base de dados”.
	6DcA **a)** Não existe nenhuma impressora instalada. O sistema apresenta a mensagem de erro “Não existem impressoras instaladas”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar se as requisições apresentadas correspondem ao estado definido e ao intervalo temporal. **2.** Verificar se a exportação é bem sucedida. **3.** Verificar se a aplicação reconhece as impressoras instaladas no sistema operativo e respetivas propriedades. **4.** Verificar se a impressão é bem sucedida.



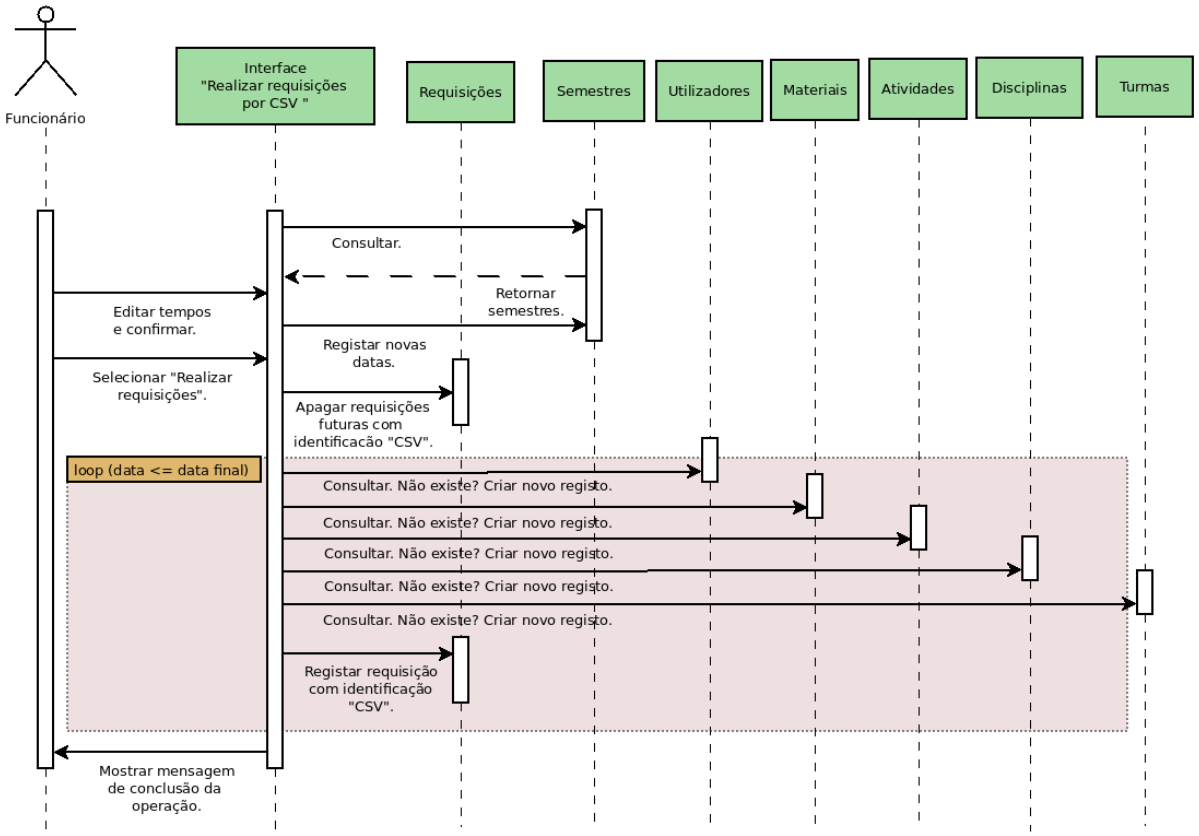
Nome	Pesquisar por recurso
Objetivo	Pesquisar recursos através das suas caraterísticas.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Alta
Fluxo principal	**1.** O ator dirige-se à opção “Pesquisar recurso” do menu da janela principal. **2.** O sistema apresenta uma janela com a lista de todos os materiais do tipo definido na janela principal, ao mesmo tempo que apresenta opções de pesquisa (caraterísticas). (2A) **3.** O ator seleciona ou arrasta caraterísticas para o quadro de pesquisa. (3A) **4.** O sistema filtra os materiais de acordo com a associação daquele às caraterísticas escolhidas. (4A)
Fluxos alternativos	**3A** a) O ator seleciona outro tipo de material na opção apropriada. b) O sistema atualiza a lista e as opções de pesquisa associadas ao novo tipo de material.

	4A **a)** O ator retira as opções do quadro de pesquisa. **b)** O sistema realiza nova atualização. Com o quadro de pesquisa vazio, o sistema apresenta todos os recursos daquele tipo.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema mostra a mensagem: “Não existe ligação à base de dados”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar se a pesquisa corresponde aos parâmetros desejados. **2.** Verificar se o campo vazio corresponde à listagem total dos materiais.



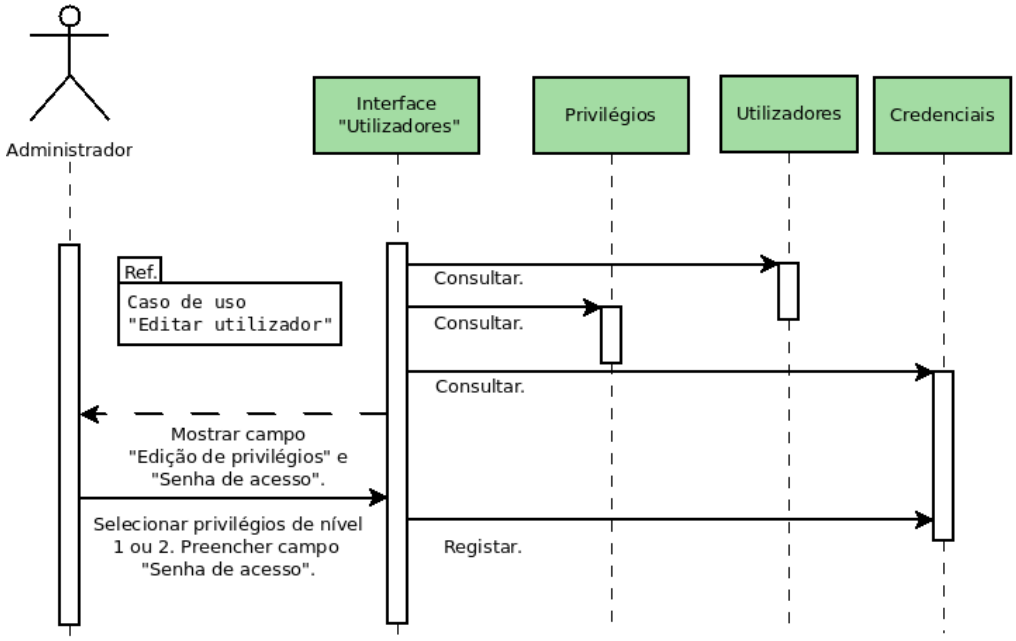
Nome	Efetuar reservas através de documento CSV
Objetivo	Realizar reservas de salas de aula através da informação existente no documento CSV obtido na plataforma escolar.
Atores envolvidos	Funcionário
Pré-condição	Login válido
Prioridade	Alta
Fluxo principal	**1.** O ator dirige-se à opção “Realizar requisições por CSV”, opção existente na janela principal. **2.** O sistema apresenta uma janela com campos onde se identificam os períodos letivos (semestres) e a opção para realizar requisições. **3.** O ator escolhe os períodos letivos e seleciona a opção “Realizar requisições”. **4.** O sistema compara o tempo atual com os períodos introduzidos e calcula as requisições conforme o período mais ajustado. O sistema regista as requisições e apresenta a mensagem de conclusão da operação. (4A) (4B) (4C)
Fluxos alternativos	O ator pode encerrar a janela a qualquer altura (exceto após selecionar a opção “Realizar requisições”).

	4A **a)** O utilizador não existe na base de dados, o sistema regista novo utilizador. O recurso não existe na base de dados, o sistema regista novo recurso. A atividade letiva não existe na base de dados, o sistema regista nova atividade. A turma não existe na base de dados, o sistema regista nova turma. A disciplina não existe na base de dados, o sistema regista a nova disciplina.
Fluxos de exceção	**4B** **a)** Não existe ligação à base de dados, o sistema a mostra a mensagem: “Não existe ligação à base de dados”.
	4C **a)** A transação não foi bem sucedida na globalidade. O sistema cancela toda a operação e apresenta a mensagem do erro.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar se as requisições são efetuadas conforme o tempo escolhido, evitando feriados e tempos de paragem. **2.** Verificar se as requisições não são duplicadas. **3.** Verificar se os relacionamentos estão corretos e válidos.



Nome	Editar utilizadores com gestão de privilégios.
Objetivo	Gerir privilégios dos utilizadores do sistema.
Atores envolvidos	Administrador
Pré-condição	Login válido
Prioridade	Média
Fluxo principal	**1.** O ator executa o caso de uso “Editar utilizador”. **2.** O sistema verifica a presença de um utilizador com privilégios elevados (nível 1) e para além das opções presentes no caso de uso “Editar Utilizador”, exibe as opções “Editar privilégios” e “Senha de acesso”. (2A)

	3. O ator seleciona o privilégio pretendido. Preenche o campo “Senha de acesso”. Confirma a opção. (3A) 4. O sistema regista a opção.
Fluxos alternativos	Todos os verificados para o caso de uso “Editar utilizador”.
	3A a) O ator não preenche o campo “Senha de acesso”. b) O sistema verifica o nível do privilégio. Com privilégios altos (1 e 2), o sistema impede o registo. O sistema Mostra mensagem: “Não preencheu o campo ‘Senha de acesso’ ”.
Fluxos de exceção	**2A** **a)** Não existe ligação à base de dados, o sistema apresenta a mensagem do erro ocorrido: “Não existe ligação à base de dados”.
Pós-condição	Não existe.
Casos de teste	**1.** Verificar a alteração dos privilégios do utilizador.



7.2. Algoritmos.

```
ALGORITMO "Data anterior a tantos dias (ndias)"
VARIÁVEIS
        ano: Inteiro, mês: Inteiro , dia : Inteiro,
        ndias: Inteiro, meses[13] : Inteiros, ndias_auxiliar: Inteiro,
        ndia_do_ano: Inteiro, anoauxiliar: Inteiro
INICIO
        LEIA ndias, ano, mês, dia
        meses[0] ← 0, meses[1] ←31, meses[2] ← 28, meses[3] ← 31, meses[4] ← 30,
        meses[5] ← 31, meses[6] ← 30, meses[7] ← 31, meses[8] ←31, meses[9] ←30,
        meses[10] ← 31, meses[11] ← 30,          meses[12] ← 31
        SE (ndias > 0) ENTÃO
                SE ( ( dia – ndias ) <= 0 ) ENTÃO
                        LEIA ndia_do_ano
                        ndias_auxiliar ← ndia_do_ano - ndias
                        SE ( ndias_auxiliar > 0 ) ENTÃO
                                ndias_auxiliar ← ndias_auxiliar - 1
                                dia ← 1
                                mês ← 1
                                RETORNA  dataApos ( ndias_auxiliar )
                        SENÃO
                                SE ( Bissexto ( ano - 1 ) ) ENTÃO
                                        anoauxiliar ← - 366
                                SENÃO
                                        anoauxiliar ← - 365
                                FIM SE
                                ano ← ano - 1
                                ENQUANTO ( ndias_auxiliar < anoauxiliar ) FAÇA
                                        SE ( Bissexto ( ano ) ) ENTÃO
                                                ndias_auxiliar ← ndias_auxiliar + 366
                                        SENÃO
                                                ndias_auxiliar ← ndias_auxiliar + 365
                                        FIM SE
                                        ano ← ano - 1
                                        SE ( ano = 0 ) ENTÃO
                                                ano ← ano - 1
                                        FIM SE
                                        SE ( Bissexto ( ano ) ) ENTÃO
                                                anoauxiliar ← - 366
                                        SENÃO
                                                anoauxiliar ← - 365
                                FIM ENQUANTO
                                SE ( Bissexto ( ano ) ) ENTÃO
                                        meses[2] ← 29
                                FIM SE
                                SE ( ano > 0 ) ENTÃO
                                        PARA decremento DE 12 ATÉ 1 FAÇA
                                                mês ← decremento
                                                SE ( ndias_auxiliar <= - meses[decremento] ) ENTÃO
                                                        ndias_auxiliar ← ndias_auxiliar + meses [decremento]
                                                SENÃO
                                                        dia ←meses [decremento] + ndias_auxiliar
                                                        INTERROMPE
                                                FIM SE
                                                SE ( ( ndias_auxiliar = 0 )  & ( mês = 1 )  & ( ano = 1 ) ) ENTÃO
                                                        dia ← 1
                                                        ano ← -1
```

```
                                        FIM SE
                                FIM PARA
                        SENÃO
                                ndias_auxiliar = - ndias_auxiliar
                                PARA incremento DE 1 ATÉ 12 FAÇA
                                        mês ← incremento
                                        SE ( ndias_auxiliar > meses [incremento] ) ENTÃO
                                                ndias_auxiliar ← ndias_auxiliar - meses [incremento]
                                        SENÃO
                                                dia ← ndias_auxiliar
                                                INTERROMPE
                                        FIM SE
                                FIM PARA
                        FIM SE
                        RETORNA dia, mês, ano
                FIM SE
            dia ←dia + ndias
            RETORNA dia, mês, ano
        FIM SE
RETORNA dia, mês, ano
FIM ALGORITMO
```

ALGORITMO " Cálculo de dias entre datas (data_a, data_b)"

```
VARIÁVEIS
ano_a: Inteiro
ano_b: Inteiro
dia_do_ano_a: Inteiro
dia_do_ano_b: Inteiro
ano_auxiliar:  Inteiro
dias_retornados: Inteiro
INICIO
        LEIA ano_a, ano_b, dia_do_ano_a, dia_do_ano_b
        SE ( ( ano_a – ano_b ) > 0 ) ENTÃO
                ano_auxiliar ← ano_b + 1
                SE ( Bissexto ( ano_b ) ) ENTÃO
                        dias_retornados ←366 - dia_do_ano_b
                SENÃO
                        dias_retornados ← 365 - dias_do_ano_b
                FIM SE
                ENQUANTO ( ano_auxiliar < ano_a ) FAÇA
                        SE ( Bissexto ( ano_auxiliar ) ) ENTÃO
                                dias_retornados ←dias_retornados + 366
                        SENÃO
                                dias_retornados ←dias_retornados + 365
                        FIM SE
                        ano_auxiliar ← ano_auxiliar + 1
                FIM ENQUANTO
                dias_retornados ←dias_retornados + dia_do_ano_a
                RETORNA dias_retornados
        SENÃO SE ( ( ano_a – ano_b ) < 0 ) ENTÃO
                ano_auxiliar ← ano_b - 1
                dia_retornados ←( - dia_do_ano_b )
                ENQUANTO ( ano_auxiliar >= ano_a ) FAÇA
                        SE ( Bissexto ( ano_auxiliar ) ) ENTÃO
                                dias_retornados ←dias_retornados - 366
                        SENÃO
                                dias_retornados ←dias_retornados - 365
                        FIM SE
                        ano_auxiliar ← ano_auxiliar - 1
                FIM ENQUANTO
                dias_retornados = dias_retornados + dia_do_ano_a
                RETORNA dias_retornados
```

```
        SENÃO
                dias_retornados = dia_do_ano_a - dia_do_ano_b
                RETORNA dias_retornados
        FIM SE
FIM ALGORITMO
```

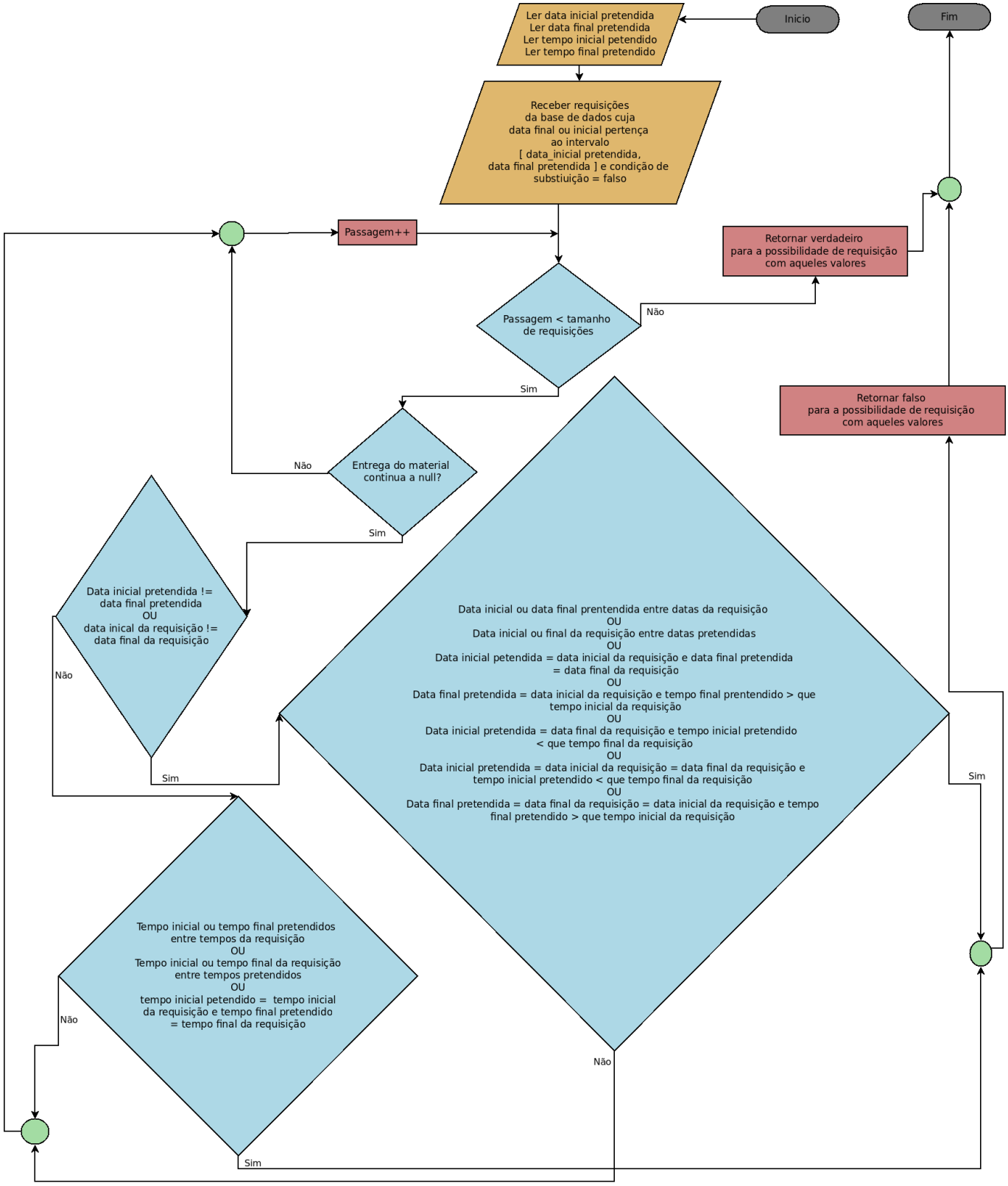
ALGORITMO " Adicionar segundos a uma tempo (segundos a adicionar)"

```
CONSTANTES
segundo_atual: Inteiro
minuto_atual: Inteiro
hora_atual: Inteiro
VARIÁVEIS
segundo: Inteiro
minuto: Inteiro
hora_ Inteiro
segundos_a_adicionar: Inteiro
INICIO
LEIA segundos_a_adicionar
minuto ← 0
hora ← 0
segundo ← segundo_atual + segundos_a_adicionar
ENQUANTO ( segundo  > 59 ) FAÇA
        minuto ← minuto + 1
        segundo ← segundo - 60
        SE ( minuto = 60 ) ENTÃO
                minuto = 0
                hora = hora + 1
                SE (hora = 24) ENTÃO
                        hora = 0
                FIM SE
        FIM SE
FIM ENQUANTO
RETORNA hora, minuto, segundo
FIM ALGORITMO
```

ALGORITMO " Compara o tempo com a diferença em segundos (tempo_a, tempo_b)"

```
CONSTANTES
hora_a: Inteiro, minuto_a: Inteiro, segundo_a: Inteiro
hora_b: Inteiro, minuto_b: Inteiro, segundo_b: Inteiro
VARIÁVEIS
segundos
INICIO
segundos = ( hora_a – hora_b ) * 3600
segundos = segundos + ( minuto_a – minuto_b ) * 60
segundos = segundos + ( segundo_a - segundo_b )
RETORNA segundos
FIM ALGORITMO
```

Algoritmo “Disponibilidade do recurso”



7.3. Triggers.

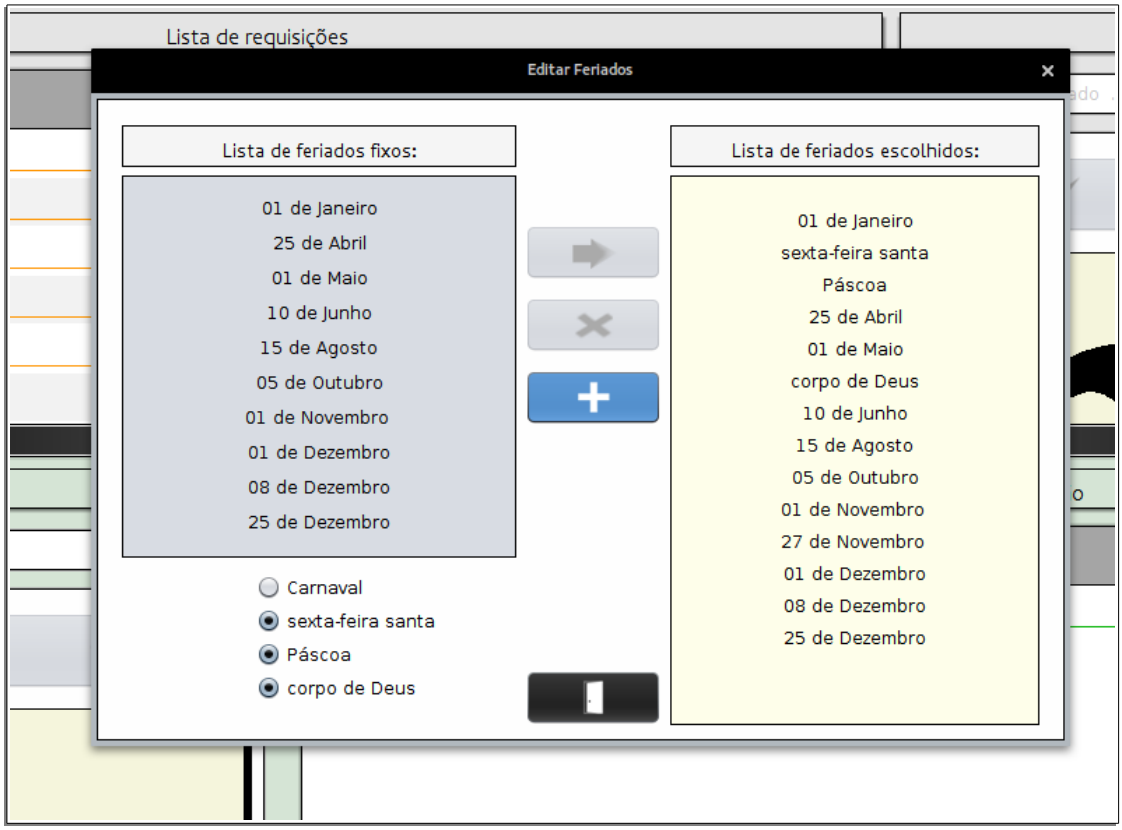
CREATE TRIGGER `after_delete_materials` AFTER DELETE ON `Materials`
FOR EACH ROW UPDATE TypesOfMaterial set TypesOfMaterial.total = (SELECT count(*) from Materials where OLD.id_tipo = Materials.id_tipo) Where TypesOfMaterial.id_tipo = OLD.id_tipo

CREATE TRIGGER `after_material_loan` AFTER UPDATE ON `Materials`
FOR EACH ROW UPDATE TypesOfMaterial SET TypesOfMaterial.livres = (SELECT COUNT(*) from Materials where Materials.id_tipo = NEW.id_tipo and Materials.estado = 0) where TypesOfMaterial.id_tipo = NEW.id_tipo

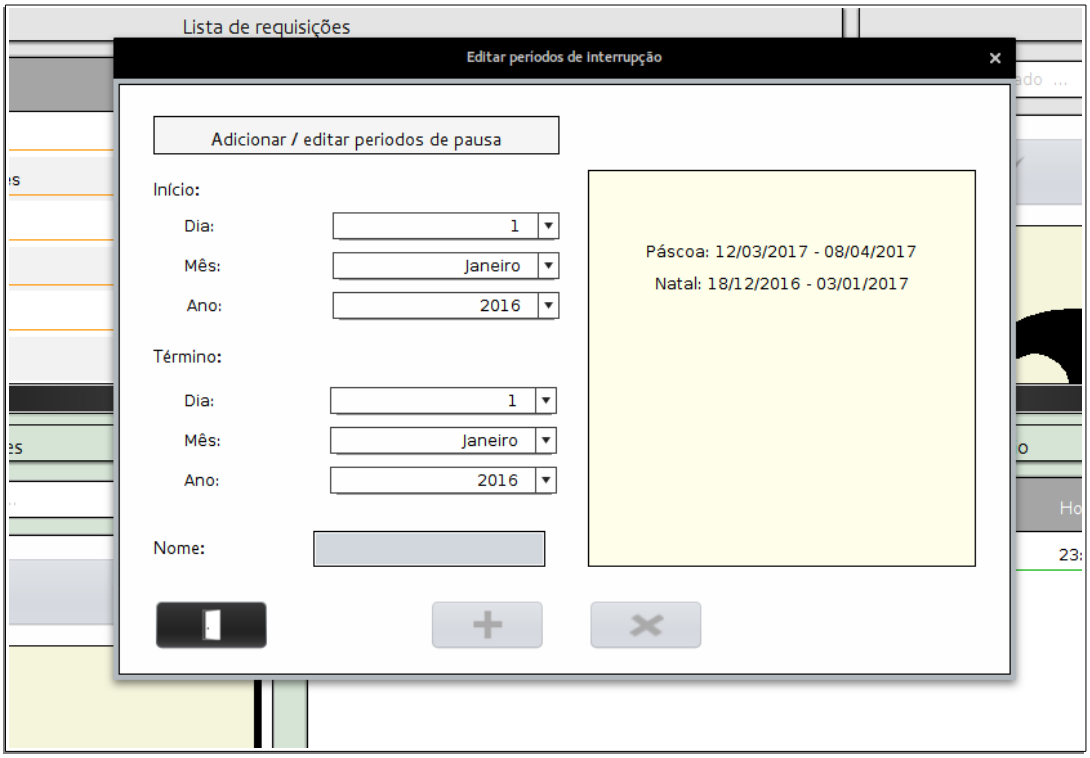
CREATE TRIGGER `after_insert_materials` AFTER INSERT ON `Materials`
FOR EACH ROW UPDATE TypesOfMaterial set TypesOfMaterial.total = (SELECT count(*) from Materials where NEW.id_tipo = Materials.id_tipo) Where TypesOfMaterial.id_tipo = NEW.id_tipo

7.4. Interfaces.

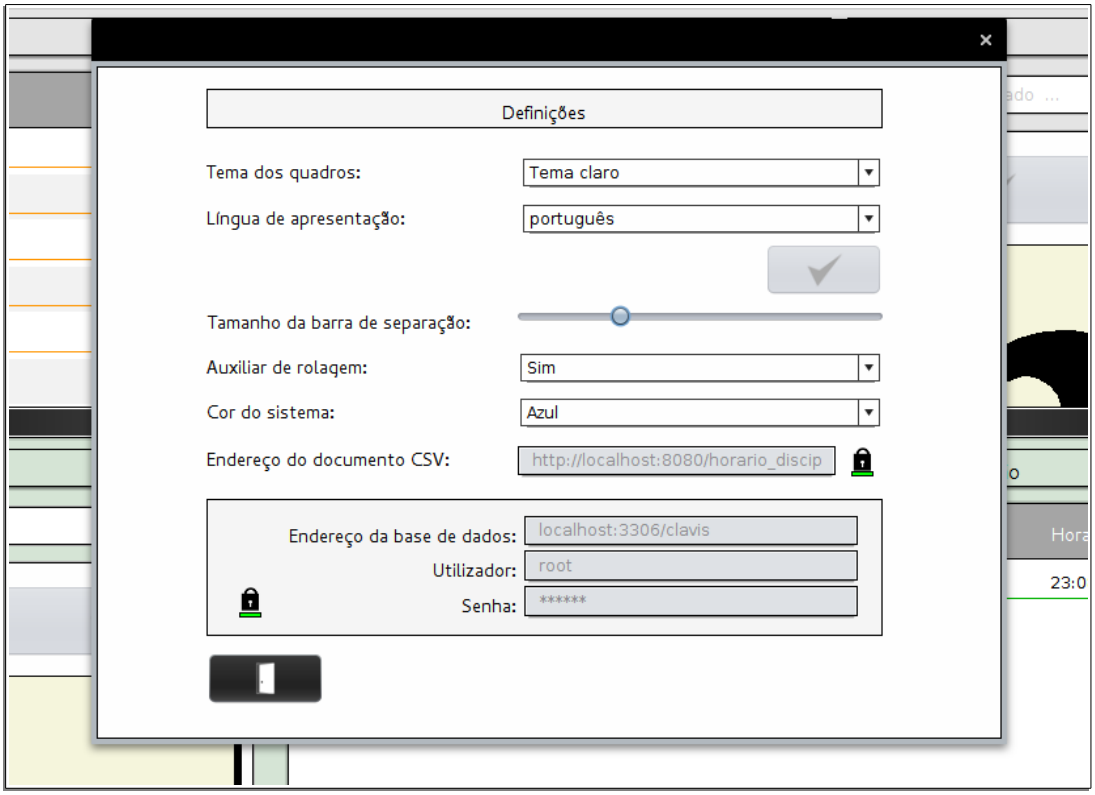
"Editar feriados"



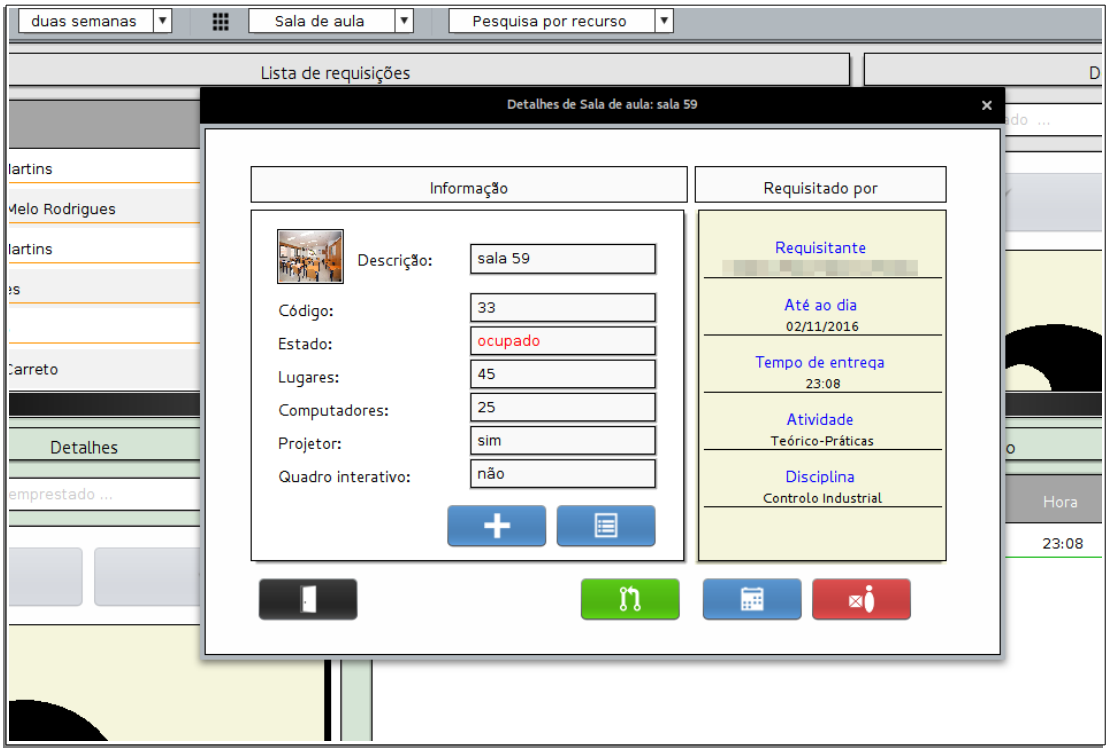
“Editar períodos de interrupção / férias”



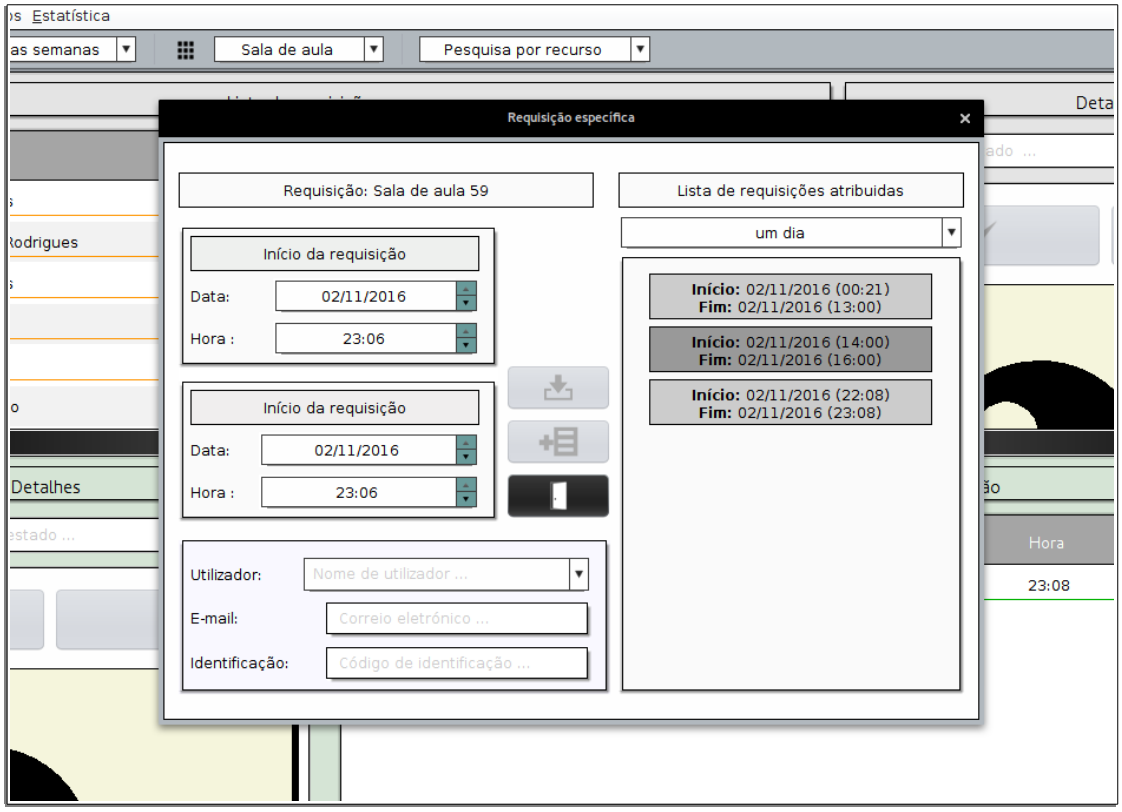
“Definições”



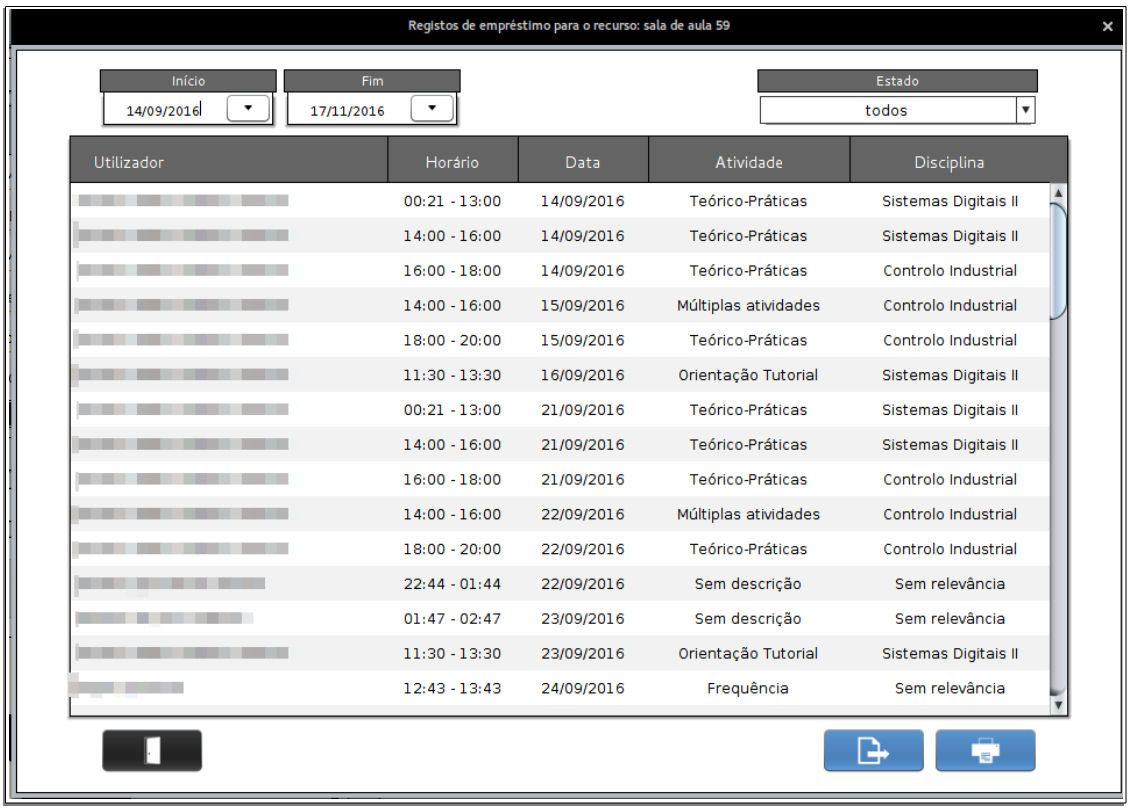
“Detalhes”



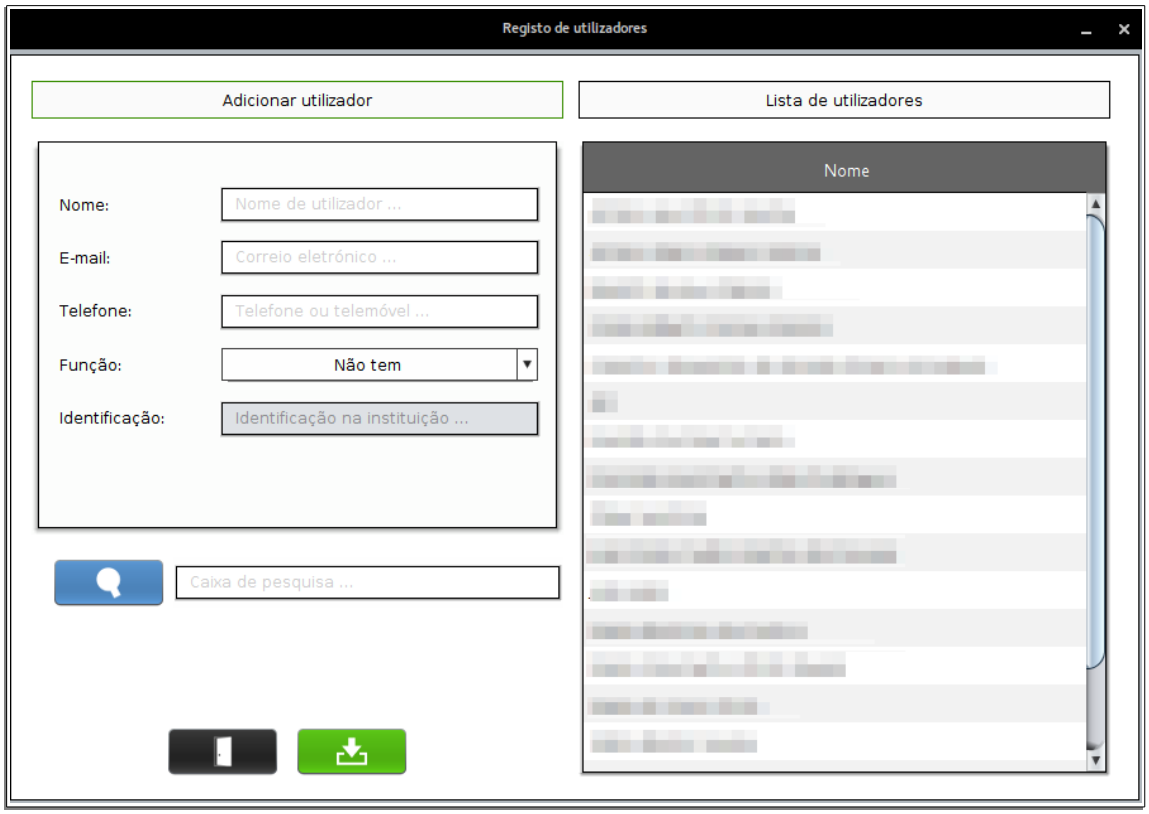
“Requisição específica de um recurso”



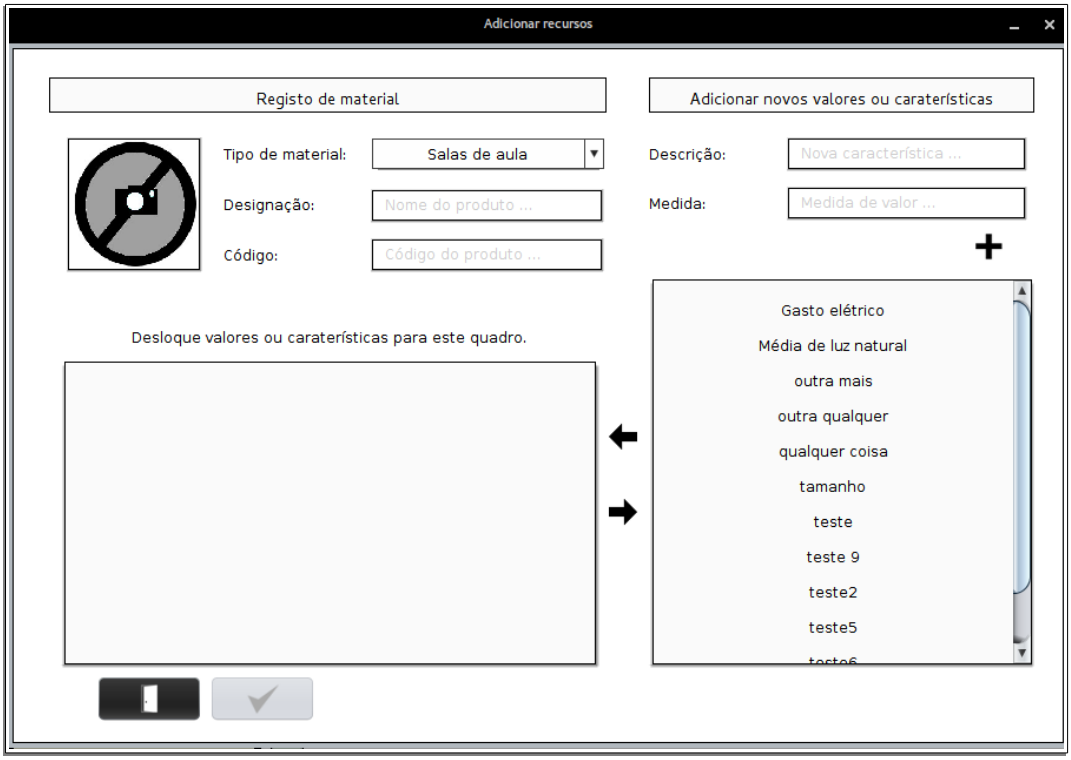
“Registo de atividade por recurso”



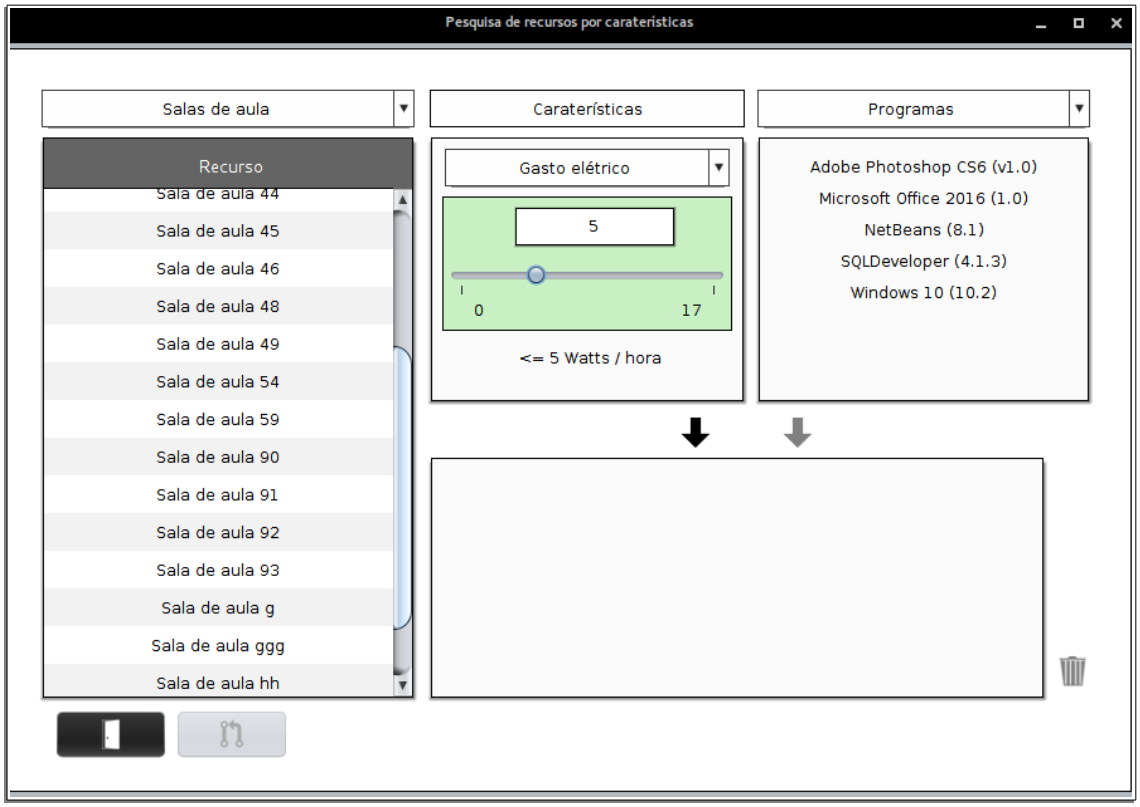
“Interface Utilizadores”



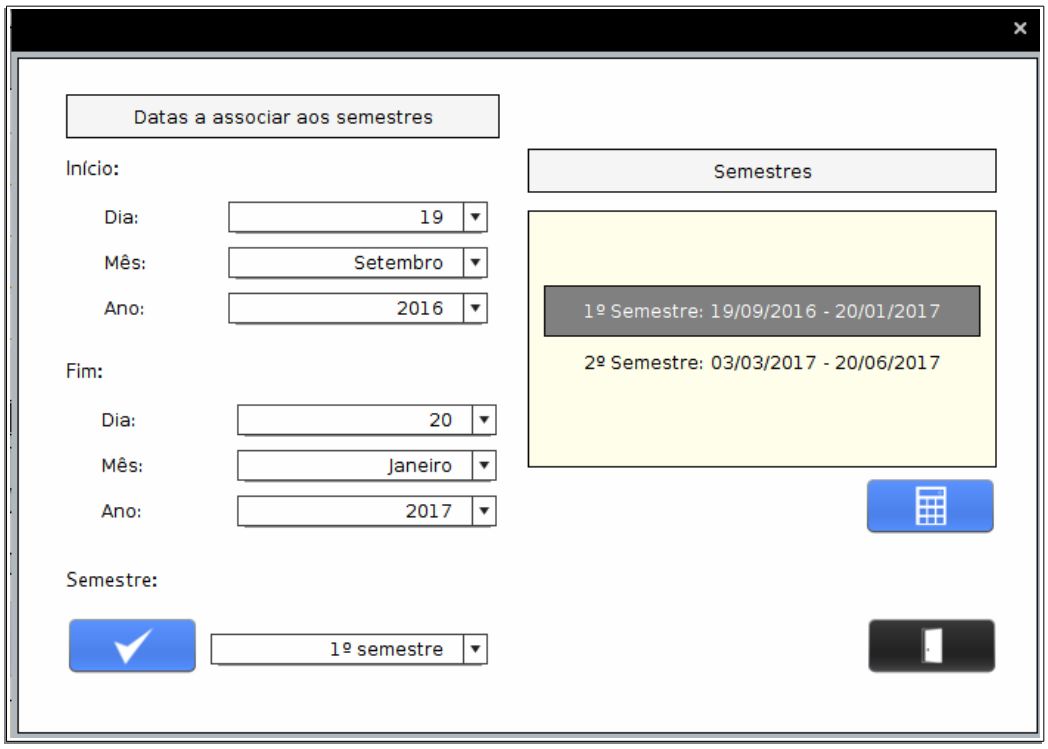
"Adicionar recursos"



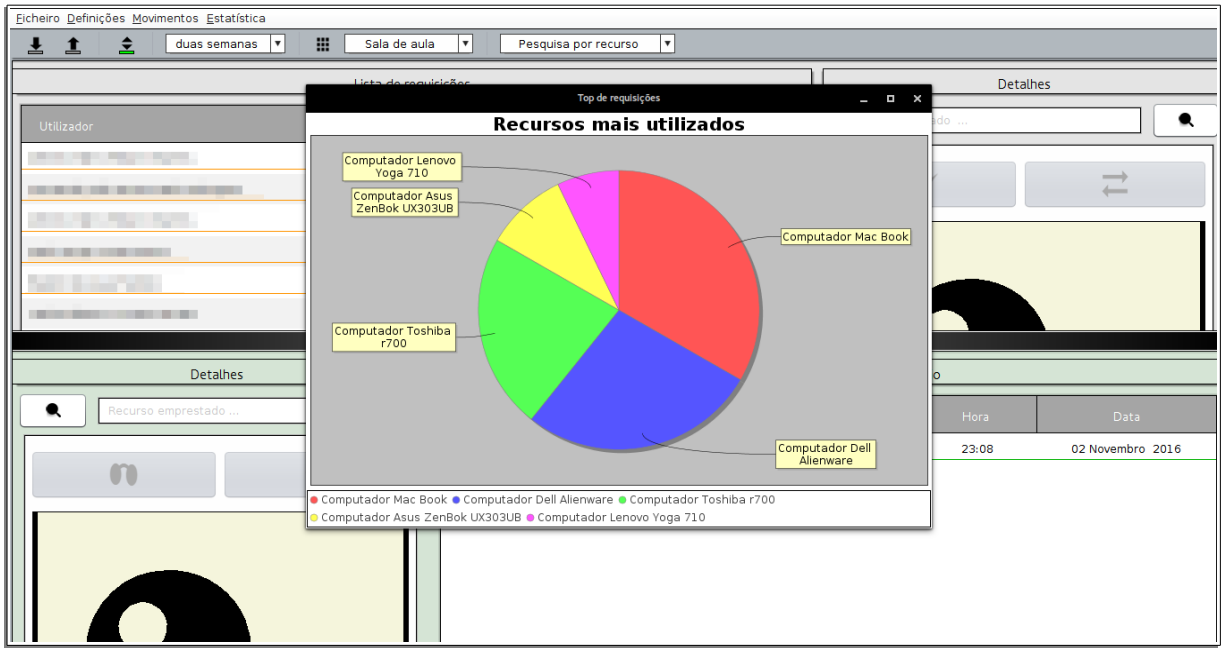
"Pesquisar por recursos"



“Requisições por CSV”



“Estatística”



7.5. Código.

Código em "Holiday.java"

```
package TimeDate;
import java.text.ParseException;
import java.util.logging.Level;
import java.util.logging.Logger;
public class Holiday implements Comparable<Holiday> {
    private static final long serialVersionUID = 1L;
    private int mes;
    private int dia;
    private boolean dinamico;
    private Langs.Locale locale;
    private static Date pascoa;
    private boolean ponte;
    private boolean expandido;
    public Holiday() {
        mes = 0;
        dia = 0;
        locale = new Langs.Locale();
        dinamico = false;
        ponte = false;
        expandido = false;
    }
    public Holiday(int dia, int mes) {
        this.mes = mes;
        this.dia = dia;
        dinamico = false;
        locale = new Langs.Locale();
        ponte = false;
        expandido = false;
    }
    public Holiday(Holiday hol) {
        this.mes = hol.getMonth();
        this.dia = hol.getDay();
        locale = hol.getLanguage();
        this.dinamico = hol.isDinamic();
        this.ponte = hol.isAdjusted();
    }
    public int getMonth() {
        return mes;
    }
    public void setMonth(int mes) {
        if ((mes > 0) && (mes < 13)) {
            this.mes = mes;
        } else {
            this.mes = 0;
        }
    }
    public int getDay() {
        return dia;
    }
    public void setDay(int dia) {
        this.dia = dia;
    }
    @Override
    public String toString() {
```

```
    String sdia;
    String smes;
    if (dia < 10) {
        sdia = "0" + dia;
    } else {
        sdia = "" + dia;
    }
    if (mes < 10) {
        smes = "0" + mes;
    } else {
        smes = "" + mes;
    }
    return sdia + " / " + smes;
}
public String toLongString() {
    String sdia;
    if (dia < 10) {
        sdia = "0" + dia;
    } else {
        sdia = "" + dia;
    }
    String[] meses = {"Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho", "Agosto", "Setembro",
            "Outubro", "Novembro", "Dezembro"};
    return sdia + " " + locale.translate("de") + " " + locale.translate(meses[mes - 1]);
}
public Langs.Locale getLanguage() {
    return locale;
}
public void setLanguage(Langs.Locale locale) {
    this.locale = locale;
}
public void setLanguage(String locale) {
    if (this.locale == null) {
        this.locale = new Langs.Locale();
    }
    this.locale.setLocale(locale);
}
public WeekDay getWeekDay(int ano) {
    Date dat = new Date(dia, mes, ano);
    WeekDay di = null;
    try {
        di = new WeekDay(dat);
        di.setLanguage(locale);
    } catch (ParseException ex) {
        Logger.getLogger(Holiday.class.getName()).log(Level.SEVERE, null, ex);
    }
    return di;
}
public static Holiday[] getMobileHolidays(int ano) {
    calcEaster(ano);
    Date corpo_de_Deus = pascoa.dateAfter(60);
    Date carnaval = pascoa.dateBefore(47);
    Date sexta_feira_santa = pascoa.dateBefore(2);
    Holiday[] holidays2 = {
        new Holiday(carnaval.getDay(), carnaval.getMonth()),
        new Holiday(sexta_feira_santa.getDay(), sexta_feira_santa.getMonth()),
        new Holiday(pascoa.getDay(), pascoa.getMonth()),
        new Holiday(corpo_de_Deus.getDay(), corpo_de_Deus.getMonth())
    };
    return holidays2;
}
public static Holiday getEaster(int ano) {
    if ((pascoa != null) && (pascoa.isValid())) {
```

```
        Holiday pasc = new Holiday(pascoa.getDay(), pascoa.getMonth());
        pasc.setDinamic(true);
        return pasc;
    } else {
        calcEaster(ano);
        Holiday pasc = new Holiday(pascoa.getDay(), pascoa.getMonth());
        pasc.setDinamic(true);
        return pasc;
    }
}
public static Holiday getGoodFriday(int ano) {
    if ((pascoa != null) && (pascoa.isValid())) {
        Date sexta_feira_santa = pascoa.dateBefore(2);
        Holiday sexta = new Holiday(sexta_feira_santa.getDay(), sexta_feira_santa.getMonth());
        sexta.setDinamic(true);
        return sexta;
    } else {
        calcEaster(ano);
        Date sexta_feira_santa = pascoa.dateBefore(2);
        Holiday sexta = new Holiday(sexta_feira_santa.getDay(), sexta_feira_santa.getMonth());
        sexta.setDinamic(true);
        return sexta;
    }
}
public static Holiday getCorpusChristi(int ano) {
    if ((pascoa != null) && (pascoa.isValid())) {
        Date corpo_de_Deus = pascoa.dateAfter(60);
        Holiday corpo = new Holiday(corpo_de_Deus.getDay(), corpo_de_Deus.getMonth());
        corpo.setDinamic(true);
        return corpo;
    } else {
        Date corpo_de_Deus = pascoa.dateAfter(60);
        Holiday corpo = new Holiday(corpo_de_Deus.getDay(), corpo_de_Deus.getMonth());
        corpo.setDinamic(true);
        return corpo;
    }
}
public static Holiday getCarnival(int ano) {
    if ((pascoa != null) && (pascoa.isValid())) {
        Date carnaval = pascoa.dateBefore(47);
        Holiday carn = new Holiday(carnaval.getDay(), carnaval.getMonth());
        carn.setDinamic(true);
        return carn;
    } else {
        calcEaster(ano);
        Date carnaval = pascoa.dateBefore(47);
        Holiday carn = new Holiday(carnaval.getDay(), carnaval.getMonth());
        carn.setDinamic(true);
        return carn;
    }
}
private static void calcEaster(int ano) {
    int m = 0;
    int n = 0;
    if (ano >= 2000) {
        if ((ano >= 2000) && (ano <= 2099)) {
            m = 24;
            n = 5;
        } else if ((ano >= 2100) && (ano <= 2199)) {
            m = 24;
            n = 6;
        } else if ((ano >= 2200) && (ano <= 2299)) {
            m = 25;
```

```
            n = 0;
        } else if ((ano >= 2300) && (ano <= 2399)) {
            m = 26;
            n = 1;
        } else if ((ano >= 2400) && (ano <= 2499)) {
            m = 25;
            n = 1;
        }
        int a = ano % 19;
        int b = ano % 4;
        int c = ano % 7;
        int d = ((19 * a) + m) % 30;
        int e = ((2 * b) + (4 * c) + (6 * d) + n) % 7;
        int p;
        if ((p = (e + 22 + d)) > 31) {
            if ((p = d + e - 9) > 25) {
                p = p - 7;
                pascoa = new Date(p, 4, ano);
            } else {
                pascoa = new Date(p, 4, ano);
            }
        } else {
            pascoa = new Date(p, 3, ano);
        }
    }
}
@Override
public int compareTo(Holiday o) {
    int valor;
    if ((valor = Integer.compare(this.getMonth(), o.getMonth())) == 0) {
        valor = Integer.compare(this.getDay(), o.getDay());
    }
    return valor;
}
public final boolean isDinamic() {
    return dinamico;
}
public final void setDinamic(boolean dinamico) {
    this.dinamico = dinamico;
}
public boolean isAdjusted() {
    return ponte;
}
public void adjust() {
    TimeDate.Date dat = new TimeDate.Date();
    dat.setDay(dia);
    dat.setMonth(mes);
    TimeDate.WeekDay di;
    try {
        switch ((di = new WeekDay(dat)).getDayNumber()) {
            case 5:
                dat = dat.dateAfter(1);
                this.ponte = true;
                break;
            case 3:
                dat = dat.dateBefore(1);
                this.ponte = true;
                break;
            default:
                this.ponte = false;
                break;
        }
    } catch (ParseException ex) {
```

```
            Logger.getLogger(Holiday.class.getName()).log(Level.SEVERE, null, ex);
        }
        dia = dat.getDay();
        mes = dat.getMonth();
    }
    public boolean isExpanded() {
        return expandido;
    }
    public TimeDate.Date setExpanded() {
        TimeDate.Date dat = new TimeDate.Date();
        dat.setDay(this.dia);
        dat.setMonth(this.mes);
        switch (TimeDate.WeekDay.getDayWeek(dat)) {
            case 3:
                dat = dat.dateBefore(1);
                this.expandido = true;
                return dat;
            case 5:
                dat = dat.dateAfter(1);
                this.expandido = true;
                return dat;
            default:
                this.expandido = false;
                return null;
        }
    }
    public void clearExpandedState() {
        this.expandido = false;
    }
    public void clearAdjustedState() {
        this.ponte = false;
    }
}
```

Código “Thread para controlo do *scroll* das tabelas”

```
private void controlScroll() {
    if (scrollAtivo) {
        scroll = new Timer(1000, (ActionEvent e) -> {
            TimeDate.Time tempo = new TimeDate.Time();
            TimeDate.Date dat = new TimeDate.Date();
            int i = 0;
            if (lista_req.getRequestList().getRequests().size() > 0) {
                if (jScrollPaneRequisicoes.getVerticalScrollBar().isVisible()) {
                    for (Keys.Request req : lista_req.getRequestList().getRequests()) {
                        int val = tempo.compareTime(req.getTimeBegin());
                        if ((req.getBeginDate().getDay() == dat.getDay()) && (req.getBeginDate().getMonth() ==
                        dat.getMonth()) && (req.getBeginDate().getYear() == dat.getYear())) {
                            int valfinal;
                            if (new TimeDate.Date().getDayYear() < req.getEndDate().getDayYear()) {
                                valfinal = tempo.compareTime(req.getTimeEnd()) + ((req.getEndDate().getDayYear() - new
                                TimeDate.Date().getDayYear()) * 86400);
                            } else {
                                valfinal = tempo.compareTime(req.getTimeEnd());
                            }
                            if ((val < lista_req.getBeforeHour()) && (valfinal >= lista_req.getAfterHour())) {
                                int max = jScrollPaneRequisicoes.getVerticalScrollBar().getMaximum();
                                int nlinhas = lista_req.getRequestList().getRequests().size();
                                max = max / nlinhas;
```

```
                    if (i > 0) {
                        i = i - 1;
                    }
                    int mudanca = max * i;
                    if (!isRowVisible(lista_req.getTable(), i)) {
                        jScrollPaneRequisicoes.getVerticalScrollBar().setValue(mudanca);
                    }
                    break;
                } else {
                    i++;
                }
            }
        }
    }
}
if (lista_dev.getRequestList().getRequests().size() > 0) {
    if (jScrollPaneDevolucoes.getVerticalScrollBar().isVisible()) {
        for (Keys.Request req : lista_dev.getRequestList().getRequests()) {
            if ((req.getBeginDate().getDay() == dat.getDay()) && (req.getBeginDate().getMonth() ==
            dat.getMonth()) && (req.getBeginDate().getYear() == dat.getYear())) {
                int valfinal = tempo.compareTime(req.getTimeEnd());
                if (valfinal >= lista_dev.getAfterHour() + lista_dev.getPermittedDelay()) {
                    int max = jScrollPaneDevolucoes.getVerticalScrollBar().getMaximum();
                    int nlinhas = lista_dev.getRequestList().getRequests().size();
                    max = max / nlinhas;
                    if (i > 0) {
                        i = i - 1;
                    }
                    int mudanca = max * i;
                    if (!isRowVisible(lista_dev.getTable(), i)) {
                        jScrollPaneDevolucoes.getVerticalScrollBar().setValue(mudanca);
                    }
                    break;
                } else {
                    i++;
                }
            }
        }
    }
}
});
scroll.start();
}
}
```

Código “Ajustamento do thread anterior”

```
AdjustmentListener list = new AdjustmentListener() {
    int val = 0;
    int val2 = 0;
    @Override
    public void adjustmentValueChanged(AdjustmentEvent e) {
        if ((scroll != null) && (scroll.isRunning())) {
            if ((val != jScrollPaneRequisicoes.getVerticalScrollBar().getValue())||(val2 !=
            jScrollPaneDevolucoes.getVerticalScrollBar().getValue())) {
                scroll.stop();
                scroll.setInitialDelay(30000);
                scroll.start();
            }
        }
        val = jScrollPaneRequisicoes.getVerticalScrollBar().getValue();
```

```
            val2 = jScrollPaneDevolucoes.getVerticalScrollBar().getValue();
        }
    };
    jScrollPaneRequisicoes.getVerticalScrollBar().addAdjustmentListener(list);
    jScrollPaneDevolucoes.getVerticalScrollBar().addAdjustmentListener(list);
```

Código “PersonalComboBox.java”

```
package Components;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.event.ActionListener;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.BorderFactory;
import javax.swing.ComboBoxModel;
import javax.swing.DefaultComboBoxModel;
import javax.swing.JList;
import javax.swing.ListCellRenderer;
import javax.swing.plaf.basic.BasicComboPopup;
public class PersonalCombo {
    public static final java.awt.Color FOREGROUND_SELECT_COLOR = new java.awt.Color(0, 0, 0);
    public static final java.awt.Color BACKGROUND_SELECT_COLOR = java.awt.Color.DARK_GRAY;
    public static final java.awt.Color BACKGROUND_COLOR = new java.awt.Color(213, 213, 213);
    public static final int LEFT = javax.swing.JLabel.LEFT;
    public static final int RIGHT = javax.swing.JLabel.RIGHT;
    public static final int CENTER = javax.swing.JLabel.CENTER;
    public static final String HELP_TEXT = "Escolha uma das opções ...";
    private javax.swing.JComponent perdefocus;
    private java.awt.Color selectForegroundColor;
    private java.awt.Color selectBackgroundColor;
    private java.awt.Color backgroundColor;
    private String textodeajuda;
    private int posicaotexto;
    private javax.swing.JComboBox<Object> combo;
    public PersonalCombo(javax.swing.JComponent perdefocus) {
        combo = new javax.swing.JComboBox<>();
        this.selectForegroundColor = FOREGROUND_SELECT_COLOR;
        this.selectBackgroundColor = BACKGROUND_SELECT_COLOR;
        this.backgroundColor = BACKGROUND_COLOR;
        textodeajuda = HELP_TEXT;
        combo.setMaximumRowCount(10);
        this.perdefocus = perdefocus;
    }
    public void addcopyPaste(Langs.Locale lingua) {
        javax.swing.JTextField fil = (javax.swing.JTextField) combo.getEditor().getEditorComponent();
        fil.addMouseListener(Components.PopUpMenu.simpleCopyPaste(lingua, fil, false));
    }
    public void create() {
        combo.setEditable(true);
        combo.setEnabled(true);
        combo.setAutoscrolls(true);
        combo.addItem("");
        javax.swing.JTextField fil = (javax.swing.JTextField) combo.getEditor().getEditorComponent();
        if (combo.getSelectedIndex() == 0) {
            fil.setForeground(new Color(205, 205, 205));
```

```
} else {
   fil.setForeground(selectForegroundColor);
}
combo.setForeground(selectForegroundColor);
fil.setSelectionColor(this.selectBackgroundColor);
BasicComboPopup popcar = (BasicComboPopup) combo.getAccessibleContext().getAccessibleChild(0);
popcar.getList().setSelectionBackground(this.selectBackgroundColor);
popcar.getList().setBorder(BorderFactory.createEmptyBorder(1, 2, 1, 2));
combo.setFocusable(true);
combo.setBackground(this.getBackgroundColor());
combo.setSelectedIndex(0);
combo.getEditor().getEditorComponent().addMouseListener(new MouseAdapter() {
   @Override
   public void mousePressed(MouseEvent e) {
      if (e.getButton() != MouseEvent.BUTTON3) {
         if (combo.getSelectedIndex() == 0) {
            javax.swing.JTextField fil = (javax.swing.JTextField) combo.getEditor().getEditorComponent();
            fil.setText("");
         }
         fil.grabFocus();
      }
   }
   @Override
   public void mouseReleased(MouseEvent e) {
      if (e.getButton() != MouseEvent.BUTTON3) {
         if (combo.getSelectedIndex() == 0) {
            javax.swing.JTextField fil = (javax.swing.JTextField) combo.getEditor().getEditorComponent();
            fil.setCaretColor(java.awt.Color.BLACK);
            fil.setForeground(java.awt.Color.BLACK);
         }
         fil.grabFocus();
      }
   }
});
Object child = combo.getAccessibleContext().getAccessibleChild(0);
BasicComboPopup popup = (BasicComboPopup) child;
javax.swing.JList<?> list = popup.getList();
list.setCellRenderer(new ListCellRenderer<Object>() {
   @Override
   public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean isSelected,
   boolean cellHasFocus) {
      javax.swing.JLabel label = new javax.swing.JLabel();
      label.setOpaque(true);
      label.setBackground(new java.awt.Color(255, 255, 255));
      label.setBorder(BorderFactory.createEmptyBorder(0, 10, 0, 10));
      label.setHorizontalAlignment(posicaotexto);
      label.setPreferredSize(new Dimension(170,22));
      if (index != 0) {
         label.setText(value.toString());
      } else {
         label.setForeground(new java.awt.Color(205, 205, 205));
         label.setText("");
      }
      if (isSelected) {
         label.setBackground(java.awt.Color.DARK_GRAY);
         label.setForeground(java.awt.Color.WHITE);
      }
      return label;
   }
});
list.addMouseListener(new MouseAdapter() {
   @Override
   public void mouseReleased(MouseEvent e) {
```

```
        if (combo.getSelectedIndex() == 0) {
            combo.getEditor().getEditorComponent().setForeground(new java.awt.Color(205, 205, 205));
            fil.setText(textodeajuda);
            if (perdefocus != null) {
                perdefocus.requestFocus();
            }
        } else {
            combo.getEditor().getEditorComponent().setForeground(selectForegroundColor);
        }
    }
});
combo.getEditor().getEditorComponent().addFocusListener(new FocusListener() {
    @Override
    public void focusGained(FocusEvent e) {
        if (combo.getSelectedIndex() < 1) {
            fil.setForeground(new java.awt.Color(1, 1, 1));
            fil.setCaretColor(new Color(1, 1, 1));
            fil.setText("");
            combo.setSelectedIndex(-1);
        }
    }
    @Override
    public void focusLost(FocusEvent e) {
        if (combo.getSelectedIndex() < 1) {
            fil.setForeground(new java.awt.Color(205, 205, 205));
            combo.setSelectedIndex(0);
            fil.setText(textodeajuda);
        }
    }
});
combo.getEditor().getEditorComponent().addKeyListener(new KeyAdapter() {
    int aux;
    boolean bauxiliar = false;
    int conta = 0;
    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
            if (perdefocus != null) {
                perdefocus.requestFocus();
            }
        } else if (e.getKeyCode() == KeyEvent.VK_BACK_SPACE) {
            if (combo.getSelectedIndex() == 0) {
                fil.setCaretColor(new Color(1, 1, 1));
                fil.setCaretPosition(0);
                fil.setText("");
            }
        } else if ((combo.getSelectedIndex() == 0) && (bauxiliar)) {
            fil.setCaretPosition(0);
            fil.setForeground(selectForegroundColor);
            fil.setCaretColor(new Color(1, 1, 1));
            setSelectedIndex(-1);
            fil.setText("");
            bauxiliar = false;
        } else if (!combo.isPopupVisible()) {
            combo.setPopupVisible(true);
        } else if ((combo.getSelectedIndex() == 1) && (e.getKeyCode() == KeyEvent.VK_UP)) {
            combo.getEditor().getEditorComponent().setForeground(new java.awt.Color(205, 205, 205));
            fil.setCaretColor(new Color(255, 255, 255));

        }
    }

    @Override
```

```
public void keyReleased(KeyEvent e) {
    int val = e.getKeyCode();
    if ((val == KeyEvent.VK_DOWN) || (val == KeyEvent.VK_UP)) {
        if (combo.getSelectedIndex() == 0) {
            combo.getEditor().getEditorComponent().setForeground(new java.awt.Color(205, 205, 205));
            fil.setText(textodeajuda);
            fil.setCaretColor(new Color(255, 255, 255));
            bauxiliar = true;
        } else {
            combo.getEditor().getEditorComponent().setForeground(new java.awt.Color(1, 1, 1));
            fil.setCaretColor(selectForegroundColor);
        }
        if ((val == KeyEvent.VK_DOWN) && (combo.getSelectedIndex() == -1)) {
            if (combo.getModel().getSize() > 1) {
                combo.setSelectedIndex(1);
            }
        } else if (((e.getKeyCode() == KeyEvent.VK_UP) && (combo.getSelectedIndex() == -1))) {
            if (combo.getModel().getSize() > 1) {
                combo.setSelectedIndex(1);
            }
        }
    } else if ((combo.getSelectedIndex() == 0) && (val == KeyEvent.VK_ENTER)) {
        if (perdefocus != null ) {
            perdefocus.requestFocusInWindow();
        }
    } else if ((combo.isPopupVisible()) && (e.getKeyCode() == KeyEvent.VK_BACK_SPACE) &&
       (fil.getText().equals("")) && (combo.getSelectedIndex() > 0)) {
        combo.getEditor().getEditorComponent().setForeground(new java.awt.Color(205, 205, 205));
        fil.setCaretColor(new Color(255, 255, 255));
        bauxiliar = true;
        combo.setSelectedIndex(0);
        fil.setText(textodeajuda);
    } else if ((e.getKeyCode() == KeyEvent.VK_BACK_SPACE) && (combo.getSelectedIndex() == 0)) {
        fil.setCaretColor(new Color(1, 1, 1));
    }
    if ((e.getKeyCode() != KeyEvent.VK_LEFT) && (e.getKeyCode() != KeyEvent.VK_RIGHT)) {
        if (combo.getSelectedIndex() != 0) {
            String va = fil.getText();
            aux = combo.getSelectedIndex();
            DefaultComboBoxModel<?> modelo = (DefaultComboBoxModel) combo.getModel();
            for (int i = 1; i <= modelo.getSize(); i++) {
                if (modelo.getElementAt(i) != null) {
                    String mov = va;
                    String mov2 = modelo.getElementAt(i).toString();
                    mov = mov.replaceAll("[áàãäâ]", "a");
                    mov = mov.replaceAll("[íìîïĩ]", "i");
                    mov = mov.replaceAll("[éèêëẽ]", "e");
                    mov = mov.replaceAll("[úùũüû]", "u");
                    mov = mov.replaceAll("[óòôõö]", "o");
                    mov = mov.replaceAll("[ñ]", "n");
                    mov2 = mov2.replaceAll("[áàãäâ]", "a");
                    mov2 = mov2.replaceAll("[íìîïĩ]", "i");
                    mov2 = mov2.replaceAll("[éèêëẽ]", "e");
                    mov2 = mov2.replaceAll("[úùũüû]", "u");
                    mov2 = mov2.replaceAll("[óòôõö]", "o");
                    mov2 = mov2.replaceAll("[ñ]", "n");
                    if ((mov2.toLowerCase().matches("(.*)" + mov.toLowerCase().replaceAll("[\\[\\]\\(\\)\\/\\{\\}\"\\#$
                    %&=\\\\]+", "") + "(.*)")) && (!mov.equals(""))) {
                        aux = i;
                        break;
                    }
                }
            }
```

```
                combo.setSelectedIndex(aux);
                if (val == KeyEvent.VK_ENTER) {
                    if (combo.getSelectedIndex() > 0) {
                        va = combo.getSelectedItem().toString();
                    }
                }
                if (fil.getCaretPosition() == fil.getText().length()) {
                    fil.setText(va);
                }
            }
        }
    }
}
);
}
public java.awt.Color getSelectForegroundColor() {
    return selectForegroundColor;
}
public void setSelectedIndex(int val) {
    combo.setSelectedIndex(val);
    if (val == 0) {
        combo.getEditor().getEditorComponent().setForeground(new java.awt.Color(205, 205, 205));
        ((javax.swing.JTextField) combo.getEditor().getEditorComponent()).setCaretColor(new Color(255, 255, 255));
        ((javax.swing.JTextField) combo.getEditor().getEditorComponent()).setText(textodeajuda);
    } else if (val > 0) {
        combo.getEditor().getEditorComponent().setForeground(this.selectForegroundColor);
        ((javax.swing.JTextField) combo.getEditor().getEditorComponent()).setCaretColor(this.selectForegroundColor);
    }
}
public void setEnabled(boolean cond){
    if (cond) {
        combo.setEnabled(true);
        this.setSelectedIndex(0);
    } else {
        combo.setEnabled(false);
        this.setSelectedIndex(-1);
    }
}
public int getItemCount() {
    return (combo.getItemCount() - 1);
}
public void setSelectedItem(Object val) {
    int situacao = -1;
    if (val != null) {
        for (int i = 0; i < combo.getModel().getSize(); i++) {
            Object element = combo.getModel().getElementAt(i);
            if (val.toString().equals(element.toString())) {
                situacao = i;
                break;
            }
        }
    }
    if (situacao == 0) {
        combo.getEditor().getEditorComponent().setForeground(new java.awt.Color(205, 205, 205));
        ((javax.swing.JTextField) combo.getEditor().getEditorComponent()).setCaretColor(new Color(255, 255, 255));
        ((javax.swing.JTextField) combo.getEditor().getEditorComponent()).setText(textodeajuda);
    } else if (situacao > 0) {
        combo.getEditor().getEditorComponent().setForeground(this.selectForegroundColor);
        ((javax.swing.JTextField) combo.getEditor().getEditorComponent()).setCaretColor(this.selectForegroundColor);
    }
    combo.setSelectedItem(val);
}
public void removeAllItems() {
```

```
    combo.removeAllItems();
    combo.addItem("");
    combo.setSelectedIndex(0);
    combo.getEditor().getEditorComponent().setForeground(new java.awt.Color(205, 205, 205));
    ((javax.swing.JTextField) combo.getEditor().getEditorComponent()).setCaretColor(new Color(255, 255, 255));
    ((javax.swing.JTextField) combo.getEditor().getEditorComponent()).setText(textodeajuda);
    //perdefocus.requestFocus();
  }
  public void setSelectForegroundColor(java.awt.Color selectForegroundColor) {
    this.selectForegroundColor = selectForegroundColor;
  }
  public java.awt.Color getSelectBackgroundColor() {
    return selectBackgroundColor;
  }
  public void setSelectBackgroundColor(java.awt.Color selectBackgroundColor) {
    this.selectBackgroundColor = selectBackgroundColor;
  }
  public String getHelpText() {
    return textodeajuda;
  }
  public void setHelpText(String textodeajuda) {
    this.textodeajuda = textodeajuda;
  }
  public java.awt.Color getBackgroundColor() {
    return backgroundColor;
  }
  public void setBackgroundColor(java.awt.Color backgroundColor) {
    this.backgroundColor = backgroundColor;
  }
  public int getHorizontalTextPosition() {
    return posicaotexto;
  }
  public void setHorizontalTextPosition(int posicaotexto) {
    this.posicaotexto = posicaotexto;
    javax.swing.JTextField fil = (javax.swing.JTextField) combo.getEditor().getEditorComponent();
    fil.setHorizontalAlignment(posicaotexto);
    ((javax.swing.JLabel) combo.getRenderer()).setHorizontalAlignment(posicaotexto);
  }
  public javax.swing.JComboBox<Object> getComboBox() {
    return this.combo;
  }
  public void setPreferredSize(Dimension dim) {
    combo.setPreferredSize(dim);
  }
  public void setMinimumSize(Dimension dim) {
    combo.setMinimumSize(dim);
  }
  public void setMaximumSize(Dimension dim) {
    combo.setMaximumSize(dim);
  }
  public void addItem(Object i) {
    combo.addItem(i);
  }
  public void removeItem(Object i) {
    combo.removeItem(i);
  }
  public void removeItemAt(int i) {
    combo.removeItemAt(i);
  }

  public void addActionListener(ActionListener l) {
    combo.addActionListener(l);
  }
```

```
    public int getSelectedIndex(){
        return combo.getSelectedIndex();
    }
    public ComboBoxModel<Object> getModel(){
        return combo.getModel();
    }
    public Object getSelectedItem(){
        return combo.getSelectedItem();
    }
    public void setComponentFocus(javax.swing.JComponent compo){
        perdefocus = compo;
    }
    public javax.swing.JComponent getComponentFocus(){
        return perdefocus;
    }
    public static void setHorizontalTextPosition(int posicaotexto, javax.swing.JComboBox<?> pos) {
        javax.swing.JTextField fil = (javax.swing.JTextField) pos.getEditor().getEditorComponent();
        fil.setHorizontalAlignment(posicaotexto);
        ((javax.swing.JLabel) pos.getRenderer()).setHorizontalAlignment(posicaotexto);
    }
}
```

Código “PopUpMenu.java”

```
package Components;
import com.sun.glass.events.KeyEvent;
import java.awt.Event;
import java.awt.Toolkit;
import java.awt.datatransfer.Clipboard;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.StringSelection;
import java.awt.datatransfer.Transferable;
import java.awt.datatransfer.UnsupportedFlavorException;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.io.IOException;
import javax.swing.JMenuItem;
import javax.swing.JPopupMenu;
import javax.swing.KeyStroke;
public class PopUpMenu extends JPopupMenu {
    private static final long serialVersionUID = 1L;
    JMenuItem[] item;
    public PopUpMenu() {
        item = new JMenuItem[0];
    }
    public PopUpMenu(JMenuItem[] item) {
        this.item = item;
    }
    public PopUpMenu(String[] titulos) {
        int i = 0;
        item = new JMenuItem[titulos.length];
        while (i < titulos.length) {
            item[i] = new JMenuItem(titulos[i]);
            i++;
        }
    }
    public PopUpMenu(String[] titulos, ActionListener[] act) {
        int i = 0;
        item = new JMenuItem[titulos.length];
```

```
    if (titulos.length == act.length) {
        while (i < titulos.length) {
            item[i] = new JMenuItem(titulos[i]);
            item[i].addActionListener(act[i]);
            i++;
        }
    }
}
public PopUpMenu(String[] titulos, ActionListener[] act, KeyStroke[] keys) {
    int i = 0;
    item = new JMenuItem[titulos.length];
    if (titulos.length == act.length) {
        while (i < titulos.length) {
            item[i] = new JMenuItem(titulos[i]);
            item[i].addActionListener(act[i]);
            item[i].setAccelerator(keys[i]);
            i++;
        }
    }
}
public PopUpMenu(String[] titulos, ActionListener[] act, String colar, String copiar) {
    int i = 0;
    item = new JMenuItem[titulos.length];
    while (i < act.length) {
        item[i] = new JMenuItem(titulos[i]);
        item[i].addActionListener(act[i]);
        if (i == 0) {
            if (colar.equals("")) {
                item[i].setEnabled(false);
            } else {
                item[i].setEnabled(true);
            }
            item[i].setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V, Event.CTRL_MASK));
        }
        if (i == 1) {
            if (copiar.equals("")) {
                item[i].setEnabled(false);
            } else {
                item[i].setEnabled(true);
            }
            item[i].setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C, Event.CTRL_MASK));
        }
        i++;
    }
}
public PopUpMenu(String[] titulos, Langs.Locale lang) {
    int inicio = 1;
    int fim = titulos.length;
    String[] auxiliar;
    item = new JMenuItem[fim - 1];
    while (inicio < fim) {
        auxiliar = titulos[inicio].split(";;;");
        if (auxiliar.length > 1) {
            item[inicio - 1] = new JMenuItem("<html> <div style='text-align:center;'><b>" + lang.translate(auxiliar[0]) +
            ": </b>" + lang.translate("Desde as").toLowerCase() + " " + auxiliar[1] + " " + lang.translate("até às") + " " +
            auxiliar[2] + ".</div></html>");
        } else {
            item[inicio - 1] = new JMenuItem(lang.translate(auxiliar[0]));
        }
        inicio++;
    }
}
public PopUpMenu(String[] titulos, ActionListener[] act, javax.swing.JButton bt1, javax.swing.JButton bt2, boolean
```

```
cond) {
    int i = 0;
    item = new JMenuItem[titulos.length];
    while (i < act.length) {
        item[i] = new JMenuItem(titulos[i], (int) titulos[i].charAt(0));
        if (i == 0) {
            item[i].setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C, Event.CTRL_MASK));
        }
        if (i > 2) {
            item[i].setAccelerator(KeyStroke.getKeyStroke(titulos[i].charAt(0), Event.ALT_MASK));
        }
        if (!cond) {
            if (i == 3) {
                if (!bt1.isEnabled()) {
                    item[i].setEnabled(false);
                } else {
                    item[i].setEnabled(true);
                }
            }
            if (i == 4) {
                if (!bt2.isEnabled()) {
                    item[i].setEnabled(false);
                } else {
                    item[i].setEnabled(true);
                }
            }
        } else {
            bt1.setEnabled(true);
            bt2.setEnabled(true);
        }
        item[i].addActionListener(act[i]);
        i++;
    }
}
public void create() {
    int i = 0;
    this.setFocusable(false);
    while (i < item.length) {
        this.add(item[i]);
        i++;
    }
}
public static java.awt.event.MouseListener simpleCopyPaste(Langs.Locale lingua, javax.swing.JComponent fil,
boolean focus) {
    JMenuItem[] itens = new JMenuItem[2];
    String[] titulos = {lingua.translate("Copiar"), lingua.translate("Colar")};
    Clipboard c = Toolkit.getDefaultToolkit().getSystemClipboard();
    itens[1] = new JMenuItem(titulos[0]);
    itens[0] = new JMenuItem(titulos[1]);
    ActionListener list = (ActionEvent ex) -> {
        if (fil instanceof javax.swing.JTextField) {
            StringSelection nome = new StringSelection(((javax.swing.JTextField) fil).getSelectedText());
            c.setContents(nome, nome);
        } else if (fil instanceof javax.swing.JComboBox<?>) {
            javax.swing.JComboBox<?> tor = ((javax.swing.JComboBox) fil);
            javax.swing.JTextField texto = (javax.swing.JTextField) tor.getEditor().getEditorComponent();
            StringSelection nome = new StringSelection(texto.getSelectedText());
            c.setContents(nome, nome);
        }
    };
    ActionListener list2 = (ActionEvent ex) -> {
        Transferable t = c.getContents(null);
        if (t.isDataFlavorSupported(DataFlavor.stringFlavor)) {
```

```
        try {
            Object ol = t.getTransferData(DataFlavor.stringFlavor);
            String colarl = (String) t.getTransferData(DataFlavor.stringFlavor);
            javax.swing.JTextField texto = new javax.swing.JTextField();
            if (fil instanceof javax.swing.JTextField) {
                texto = ((javax.swing.JTextField) fil);
            } else if (fil instanceof javax.swing.JComboBox<?>) {
                javax.swing.JComboBox<?> tor = ((javax.swing.JComboBox) fil);
                texto = (javax.swing.JTextField) tor.getEditor().getEditorComponent();
            }
            int pos = texto.getCaretPosition();
            int tamanho = texto.getText().length();
            if (texto.getSelectedText() == null) {
                String inicio = texto.getText().substring(0, pos);
                String fim = texto.getText().substring(pos, tamanho);
                int tam = colarl.length();
                colarl = inicio + colarl + fim;
                texto.setText(colarl);
                texto.setCaretPosition(pos+tam);
            } else {
                pos = texto.getSelectionStart();
                int posseguinte = texto.getSelectionEnd();
                String inicio = texto.getText().substring(0, pos);
                String fim = texto.getText().substring(posseguinte, tamanho);
                int tam = colarl.length();
                colarl = inicio + colarl + fim;
                texto.setText(colarl);
                texto.setCaretPosition(pos+tam);
            }
        } catch (UnsupportedFlavorException | IOException eo) {
        }
    }
};
itens[1].addActionListener(list);
itens[1].setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_C, Event.CTRL_MASK));
itens[0].addActionListener(list2);
itens[0].setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_V, Event.CTRL_MASK));
PopUpMenu menu = new PopUpMenu(itens);
menu.setFocusable(focus);
menu.create();
java.awt.event.MouseListener mouse = new java.awt.event.MouseAdapter() {
    @Override
    public void mouseReleased(MouseEvent e) {
        if (fil.isFocusOwner()) {
            if (e.getButton() == MouseEvent.BUTTON3) {
                if (fil instanceof javax.swing.JTextField) {
                    if (((javax.swing.JTextField) fil).getSelectedText() == null) {
                        itens[1].setEnabled(false);
                    } else {
                        itens[1].setEnabled(true);
                    }
                } else if (fil instanceof javax.swing.JComboBox<?>) {
                    javax.swing.JComboBox<?> tor = ((javax.swing.JComboBox) fil);
                    javax.swing.JTextField texto = (javax.swing.JTextField) tor.getEditor().getEditorComponent();
                    if (((javax.swing.JTextField) fil).getSelectedText() == null) {
                        itens[1].setEnabled(false);
                    } else {
                        itens[1].setEnabled(true);
                    }
                }
                Transferable t = c.getContents(null);
                if (t.isDataFlavorSupported(DataFlavor.stringFlavor)) {
                    try {
```

```
                    Object ol = t.getTransferData(DataFlavor.stringFlavor);
                    String colarl = (String) t.getTransferData(DataFlavor.stringFlavor);
                    if ((colarl == null) || (colarl.equals(""))) {
                        itens[0].setEnabled(false);
                    } else {
                        itens[0].setEnabled(true);
                    }
                } catch (UnsupportedFlavorException | IOException eo) {
                }
            }
            menu.show(e.getComponent(), e.getX(), e.getY());
        }
    }
}

    };
    return mouse;
  }
  public JMenuItem[] getitens() {
    return item;
  }
}
```