IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Bruno Andrade Gomes

dezembrol | 2016

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

O Smartphone como cartão de aluno

Bruno Andrade Gomes

Relatório para obtenção de licenciatura

em Engenharia Informática

Dezembro 2016

Área curricular: Projeto de Informática

Ano letivo: 2015/2016

O Smartphone como cartão de aluno

Orientador: Professor António Mário Ribeiro Martins

Autor: Bruno Andrade Gomes

Identificação

Aluno: Bruno Andrade Gomes

Número: 1010391

Curso: Engenharia Informática

Data de Nascimento: 16/09/1991

Ano: 2016

Instituição de ensino: Escola Superior de Tecnologia e Gestão

Morada: Av. do rio diz 6500-855 Guarda

Telefone: +351 967 249 405

Agradecimentos

A todos aqueles que de certa forma me ajudaram ou apoiaram ao longo da realização deste Projeto, os meus agradecimentos.

Em primeiro lugar gostaria de agradecer ao professor António Mário Ribeiro Martins por ter aceitado o desafio de ser meu orientador de Projeto e por toda a ajuda, apoio e sobretudo pela sua disponibilidade que me ofereceu ao longo da conceção e realização deste.

Gostaria de agradecer à professora Natália Gomes pelo apoio dado na realização do relatório, sobretudo nos diagramas relativos à conceção das aplicações.

Aos meus pais pelo apoio e motivação dada, não só na realização do projeto como também ao longo da licenciatura.

Quero também agradecer à minha namorada pelo apoio, motivação e ajuda que me deu na conceção e realização do Projeto.

Por último, gostaria de agradecer a alguns dos meus amigos que de alguma forma me apoiaram, ao longo da realização do Projeto.

Resumo

Este relatório descreve o trabalho realizado na disciplina 'Projeto de Informática' do curso de Engenharia de Informática, tendo como tema "O smartphone como cartão de aluno".

O principal objetivo deste projeto é simplificar a vida tanto a alunos como a funcionários do instituto, oferecendo, dentro do campus do IPG, um sistema de pagamento rápido, usando a tecnologia NFC. Também se pretende criar um módulo que permita comprar uma senha da cantina online, oferecendo, assim, uma certa mobilidade. Atualmente para obterem uma senha da cantina, os alunos têm, necessariamente, que se deslocar ao campus do IPG. Pretende-se, assim, criar uma aplicação para smartphones android, um servidor para elo de ligação entre a base de dados com as informações dos alunos, e uma aplicação para smartphones (criada por motivos de segurança). Foi, assim, desenvolvido um protótipo funcional para obtenção de bens dentro do campus do Instituto e por último uma aplicação para obtenção de senhas.

Neste documento encontra-se a análise de requisitos, conceção e realização das aplicações a desenvolver. As aplicações são programadas em JAVA e usa-se o Android Studio, para criar a aplicação para smartphones, e as restantes aplicações são criados no NetBeans. As base de dados criadas no Wamp Server em mySQL.

Palavras chave: Java, Android, NFC, HostApduService, mySql, SQLite,Sockets

Abstract

This report describes the work accomplished in the discipline 'Informatics Project' of the Computer Engineering graduation, with the theme 'The smartphone as a student ID'.

The main objective of the project it's simplify the life of the students and staff, offering inside of the campus, a simple and fast method of payment, using NFC technology. Also, it will be developed a module that allows buying meal voucher without the need of being physically present in the campus, in the present cenario every student need to buy the meal voucher from within the campus. It will be created an android application and a server to create a link between the database, with the student information, and the application for smartphones (for safety measures). Then, a functional prototype was developed to obtain goods within the campus of the Institute and lastly an application for obtaining meal vouchers.

In this document you can find the requirements analysis, design and implementation of the applications to be developed. The applications are developed in JAVA using Android Studio for smartphones, for the other applications they are developed in Netbeans and the databases created in WAMP Server in MySQL.

Keywords: Java, Android, NFC, HostApduService, mySql, SQLite, Sockets;

Índice

Indice de figuras

Indice de tabelas

Lista de siglas

AID- Application Identifier

APDU- Application Protocol Data Unit

BD- Base de dados

HCE- Serviço de emulação de cartão(Host Card Emmulation)

NFC- Near Field Communication

UML- Linguagem de Modelagem Unificada

1 Introdução

Como sabemos, ao longo destes últimos anos, os dispositivos móveis (smartphones e tablets) têm vindo a demonstrar ser um bem essencial na vida das pessoas. É difícil hoje em dia encontrar alguém que não use um simples aparelho destes no seu dia a dia (Soomro, 2013). Partindo deste princípio surgiu a ideia criar uma aplicação que permita utilizar o dispositivo móvel como cartão de aluno.

A aplicação tem por base a utilização da tecnologia Near Field Communication (NFC), uma tecnologia que permite trocar informações sem fios e de forma segura entre dispositivos compatíveis que estejam próximos um do outro (Near Field Communication, 2014). Será possível assim, enviar os dados do aluno para um recetor NFC, de forma a que este passe a realizar compras de bens (exemplo: produtos do bar). Pretende-se ainda adicionar um módulo de reserva de senhas na cantina do IPG utilizando a aplicação.

Juntamente com essa aplicação, será também necessário criar uma base de dados para os alunos, onde serão guardados as suas identificações pessoais, movimentos de contas e saldo disponível. Em simultâneo, será também criado uma aplicação que permita o registo de compra de bens pela instituição e outra que permita a aquisição da senha desde que esta, tenha sido previamente reservada pelo aluno na aplicação para smartphones.

O relatório é constituído por 5 capítulos. O capítulo 1 trata-se de uma abordagem geral ao tema e objetivos do projeto. O capítulo 2 apresenta conceitos teóricos relevantes para a realização do projeto. O capítulo 3 engloba a análise e conceção do projeto. O capítulo 4 é apresentado para retratar como o projeto foi implementado. E por fim no capítulo 5 apresenta-se as conclusões.

2 Enquadramento Teórico

Sobre este projeto é importante salientar a definição e a importância do Near Field Communication (NFC). Como já foi referido, o NFC é uma tecnologia que permite a troca de dados de um dispositivo para outro sem a necessidade de fios e de forma segura desde que os dispositivos sejam compatíveis (Near Field Communication, 2014). Já existem aplicações no mercado para simular um cartão multibanco. Os mais conhecidos são o Apple Pay e o Samsung Pay. Um projeto similar, ao que é apresentado, foi criado na universidade de Vila Nova.

- **Apple Pay**

Sobre o Apple Pay é uma aplicação que substitui o cartão de crédito e que permite efetuar compras a partir de um dispositivo móvel da Apple utilizando a tecnologia NFC e usa um sistema de validação a partir de um sensor biométrico (Touch ID) para efetuar a compra/transação. Quanto à sua segurança, o Apple Pay funciona em conjunto com uma aplicação de nome Passbook onde são guardados os dados encriptados dos cartões multibanco, credito e débito. Esta aplicação apenas se encontra disponível para o Iphone 6 (Tecmundo, 2014). A figura seguinte ilustra a arquitetura de pagamento do Apple Pay.

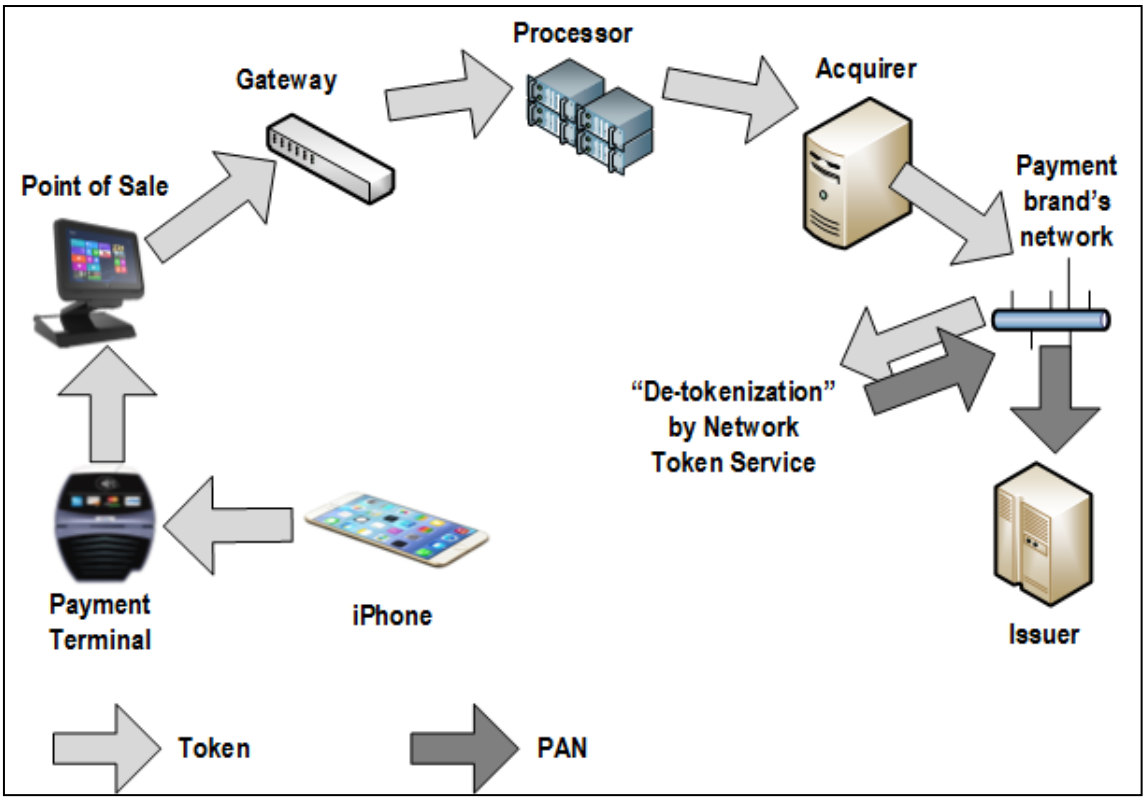

Figura 1: Arquitetura Apple Pay (Tecmundo, 2014)

- **Samsung Pay**

Quanto ao Samsung Pay, é uma aplicação que usa a tecnologia Magnetic Secure Transmission (MST) para transmitir os dados do cartão para os terminais de pagamento, e também funciona com os métodos de pagamento que usam NFC. As compras são autentificadas e validadas através de um leitor de impressões digitais e os dados são guardados em tokens de forma segura. Esta plataforma de pagamento está disponível para o Samsung Galaxy S6 e S6 Edge sendo estes os primeiros a tê-la (Wikipédia, Samsung Galaxy S6, 2015).

- **Villanova WildCard**

Em termos académicos a universidade de Villanova, situada nos Estados Unidos da América, implementou no seu campus um sistema para os alunos

utilizarem o seu smartphone como cartão de aluno, para efetuar pagamentos dentro da área do campus. A tecnologia que utilizaram para esse sistema foi a tecnologia NFC embora apenas tenha sido desenvolvida uma aplicação para Iphones. Antes da fase de implementação do projeto foi realizado um estudo sobre os benefícios que a tecnologia NFC poderia trazer aos alunos e à universidade.

O projeto iniciou a sua primeira fase com a criação de uma aplicação para o Iphone, de nome *The Villanova Wild Card*. A aplicação tinha por objetivo guardar as credenciais dos alunos para ser usado juntamente com a tecnologia NFC e simular um cartão. Nesta primeira fase também foram instalados sistemas para utilizar esta aplicação começando pelas zonas de acesso às residências da universidade, livrarias, máquinas de venda e para obtenção de refeições. Durante esta fase foram realizados inquéritos aos alunos e membros da universidade (STAFF/funcionários) que estavam a testar o sistema, do qual, obtiveram um feedback bastante positivo. Na segunda fase do projeto substituíram as fechaduras das portas que eram mecânicas por fechaduras que usam um cartão para acesso, permitindo assim aos utilizadores da aplicação bloquear e desbloquear as portas utilizando o Iphone. Segundo o diretor da universidade, o projeto foi bem recebido tanto pelos alunos como pelos funcionários da universidade e estão satisfeitos por tê-lo implementado (University Business , 2013).

2.1 NFC

È importante apresentar alguns conceitos sobre esta tecnologia, visto que, esta tem uma enorme importância no desenvolvimento deste projeto. Refiro aqui temas como dispositivos, modos e princípios de funcionamento, normas e aplicações do NFC.

2.1.1 Dispositivos

Os componentes mais ativos na tecnologia NFC, como não podia deixar de ser, são os dispositivos preparados para esta tecnologia. Temos de diferenciar os tipos de dispositivos presentes numa comunicação diferenciando assim o sujeito passivo e o sujeito ativo. O sujeito passivo não tem fonte de alimentação própria pelo que necessita de um sujeito ativo para iniciar a comunicação. Estes sujeitos são maioritariamente cartões *contacless e* tags NFC*.* Quanto ao sujeito ativo, este possui fonte de alimentação própria ao que este inicia o processo de comunicação criando um campo eletromagnético criando assim a energia para o sujeito passivo.

Grande parte dos dispositivos NFC existentes no mercado são smartphones. Mas esta tecnologia também se encontra presente em Headsets Bluetooth, terminais que podem efetuar pagamentos *contacless,* entre outros.

2.1.2 Aplicações

A tecnologia NFC tem uso nas seguintes áreas de aplicação:

- **Comercio**: pagamentos *contacless.*
- **Bluetooth**: envio dos parâmetros de coneção via NFC
- **Wifi**: envio dos parâmetros de coneção via NFC
- **Partilha de dados**: transferir dados de um smartphone para outro exemplo Android Beam
- **Ler e escrever tags**

2.1.3 Princípios de funcionamento

Saliento que uma tag NFC ou um cartão *contacless* são sempre sujeitos passivos. Um leitor NFC e um Smartphone podem ser tanto um sujeito passivo como ativo.

Por assim dizer existem dois modos de funcionamento:

- **Modo passivo:** comunicação entre um sujeito passivo e um activo. O iniciador cria um campo eletromagnético para efetuar a transmissão de dados com o dispositivo alvo. O dispositivo alvo é alimentado por esse campo. A transmissão de dados é bidirecional. A transmissão é feita num sistema de pedidos-respostas onde o iniciador envia um pedido e o dispositivo alvo devolve a resposta desse pedido. Exemplo: o iniciador pede o número do cartão e o dispositivo alvo devolve esse número.
- **Modo Ativo:** comunicação entre dois sujeitos ativos. O dispositivo iniciador cria o campo eletromagnético para o segundo dispositivo. Assim que o segundo dispositivo deteta esse campo, processa-o e responde criando o mesmo campo que o iniciador. A transmissão de dados é bidirecional mas é feita alternadamente (A envia dados para B ou B para A). Exemplo: transferir um ficheiro de um smartphone para outro.
-

Numa comunicação NFC temos de diferenciar as duas diferentes funções de funcionamento dos dispositivos NFC, sendo elas: dispositivo iniciador e o dispositivo alvo, tendo o primeiro a função de iniciar a comunicação.

2.1.4 Modos de funcionamento

Os modos de funcionamento dos dispositivos NFC são:

- **Modo Escrita/Leitura:** O dispositivo funciona como leitor e pode ler ou escrever em tags NFC e em dispositivos NFC passivos com a norma ISSO/IEC 14443.
- **Modo peer-to-peer(P2P):** Neste modo a transmissão é feita de forma bidirecional entre dois dispositivos que se encontram no mesmo modo de partilha de dados. É usado para transmitir dados entre dois dispositivos Exemplo: Envio de ficheiros, enviar contactos.
- **Modo de emulação de cartão:** Neste modo, o dispositivo NFC emula um *smart card contacless* na norma ISSO/IEC 14443 e comporta-se como um cartão. Sendo assim irá existir um sistema de comunicação pedidos-respostas. É usado tipicamente para pagamentos com o smartphone.

2.1.5 Emulação de Cartão

Existem duas variantes da emulação de cartão para Android:

- **Emulação com segurança:** aqui a comunicação é efetuada diretamente com o controlador NFC do dispositivo não passando pelo CPU do Android.
- **Host based card emulation(HCE):** conhecido como software de emulação de cartão. Aqui o controlador NFC reencaminha os dados recebidos para o CPU do Android onde as aplicações se encontram a decorrer. A partir daí, o sistema operativo escolhe a aplicação para tratar desses dados entrando dessa mesma forma, num modo pedidos-respostas.

Como o projeto vai ser implementado em HCE é importante salientar algumas características deste mesmo.

- A emulação de cartão está presente desde a API 19 do Android (Android KitKat 4.4) tornando possível, desde essa API, a implementação do conceito pagamento por smartphone.
- O HCE é implementado como componente de serviço.
- Cada serviço HCE é identificado pelo *Application ID(*AID) definido na norma ISO/IEC 7816/4.
- O recetor envia o AID da aplicação solicitada, sendo este, recebido e tratado pelo sistema Android. O sistema android escolhe e inicia o serviço HCE que tem o AID pretendido e por fim, este passa a tratar diretamente dos pedidos da parte do recetor e envia as respostas correspondentes a esses pedidos.
- Enquanto a ligação entre recetor e o dispositivo não terminar, o serviço HCE escolhido irá sempre tratar dos pedidos da parte do recetor (pedidos-respostas).

2.1.6 Normas e protocolos

As normas para comunicação são:

- **ISO/IEC 18902:** norma para a camada física e comunicação entre dois dispositivos NFC.

- **ISO/IEC 21481:** define os diferentes mecanismos de seleção de modos de comunicação entre diferentes tecnologias NFC que operam numa frequência de 13.56 Mhz.

- **ISO/IEC 14443:** Norma que descreve cartões *contacless,* o seu modo de funcionamento e protocolos de transmissão entre estes e o recetor. Alcance 7-15 cm.

Os cartões contacless e os serviços de Emulação de cartão encontram-se na norma ISO/IEC 14443 é importante salientar os protocolos de comunicação destes. Estes estão divididos em quatro partes:

- **Parte 1:** Características físicas
- **Parte 2:** Frequência, alimentação e sinal de interface
- **Parte 3:** Inicialização e anti colisão.
- **Parte 4:** Protocolos de transmissão.

Cada parte irá estar definido como ISO/IEC 14443-1,14443-2, 14443-3 e 14443-4 repectivamente. Cada parte irá atuar na camada OSI. A figura seguinte retrata cada parte da comunicação e relaciona-os com o modelo OSI.

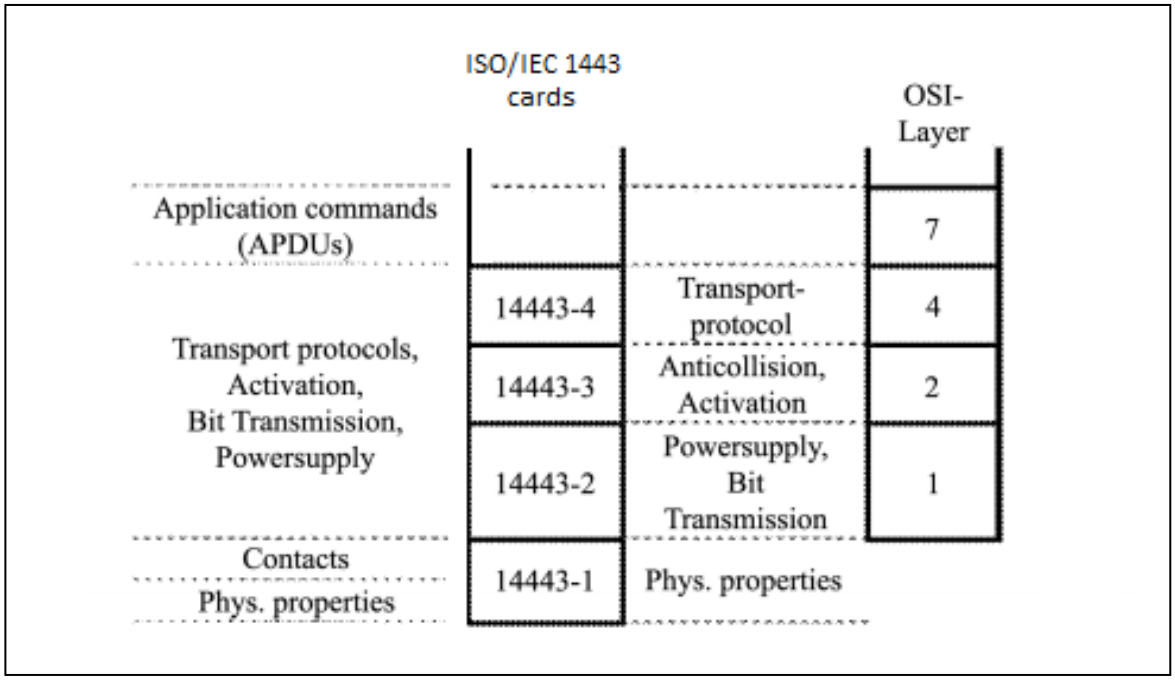

Figura 2: Protocolo de comunicação ISO/IEC 14443 para smart cards

2.1.7 APDU

Os APDUs são usados para trocar dados entre os smart cards e os recetores NFC pelo que atuam na camada de aplicação do modelo OSI.

O processo de comunicação, entre recetores e cartões, como foi referido anteriormente é um modelo de pedidos-repostas pelo que o recetor envia um *command* APDU ao cartão e por fim o cartão envia um *response* APDU. Deve-se por isso diferenciar a constituição entre os dois APDU.

Command APDU

A tabela seguinte representa como um command APDU é constituído.

Tabela 1: Constituição de um command APDU

Campo	CLA	INS	P1	P2	Lc	Data	Le
Tamanho (Bytes)	1	1	1	1	0,1,3	Variável	0-3

- **CLA** - Instruction class byte
- **INS** - Instruction number byte
- **P1, P2** – parâmetros adicionais
- **Lc** – tamanho dos dados enviados em bytes
- **Le** – tamanho esperado da resposta

Response APDU

A tabela seguinte representa como um response APDU é constituído.

Tabela 2: Constituição de um Response APDU

Campo	Dados	SW1	SW2
Tamanho (Bytes)	Variável	1	1

Os campos SW1 e SW2 são campos de estado em que o valor ’90’ e ‘00’ irá ser correspondentemente no SW1 e SW2 se a operação tiver sucesso.

2.2 Comunicação entre aplicações

As tecnologias que permitem a comunicação entre as diferentes aplicações são a tecnologia NFC e de Sockets.

Sockets

Quanto às aplicações que comunicam usando esta tecnologia são a aplicação para *smarphones* e o servidor. Quando por parte do utilizador existe pedidos de informação, esse pedido é enviado ao servidor que por sua vez trata de reenviar as informações pretendidas efetuando pesquisas à base de dados. Para melhor eficácia de comunicação e programação, convém diferenciar os diferentes pedidos do utilizador em forma de tópico. A seguir apresento as mensagens enviadas para cada pedido por parte do utilizador para o servidor e a sua resposta:

- Pedido de autentificação
  Mensagem: Login*username*password*
  Resposta: Login*estado*
  Caso os dados enviados se encontrem corretos envia nome, numero, curso do aluno.
- Pedido de informação de ementas
  Mensagem: Senhas*
  Resposta: Senhas*n_senhas a enviar*id da senha 1*prato de carne 1*prato de peixe 1*sopa*valor 1*dia de utilização 1*id da senha 2*prato de carne 2*…
- Requisição de uma senha
  Mensagem: Compra*id senha*prato escolhido*
  Resposta: Compra*estado da compra*
- Movimentos
  O mesmo princípio do pedido de informação de ementas

Para que a comunicação seja efetuada é necessário estar ligado à internet. As páginas/atividades do Android tratam de enviar informação para o

servidor e a AsyncTask (thread do android) trata de receber essas informações, guarda-las se for necessário e iniciar outras atividades.

NFC

O NFC trata da comunicação da aplicação dos s*martphones* com a aplicação de aquisição de bens e de obtenção de senhas. Estas duas últimas aplicações referidas possuem um recetor NFC para realizar a comunicação. O utilizador ao colocar o *smarphone* sobre o recetor NFC envia em string o número de aluno para depois ser utilizado pelas aplicações. Refiro que para que os dados sejam enviados o aluno tem de já ter efetuado login na aplicação para *smartphones*, garantindo assim, que o seu número de aluno se encontre na base de dados. A comunicação por parte dos smartphones é efetuada através de um serviço HCE pelo que, não é necessário a aplicação estar a "correr".

3 Análise de requisitos e conceção

Este capítulo trata das especificações técnicas e de software necessário para as diferentes aplicações. O projeto constitui na criação da aplicação para o Android, a aplicação de aquisição de bens, a aplicação de obtenção de senhas e por fim do servidor.

3.1 Metodologias

Este projeto irá estar sujeito a imensas mudanças ao longo do seu desenvolvimento e as tarefas irão ser divididas em pequenas partes simples que no fim irão compor um sistema complexo. Ao longo do processo de desenvolvimento deste projeto é necessário ter uma constante comunicação com o orientador de projecto. Tendo isso em conta a metodologia de desenvolvimento de software que irá ser aplicado neste projeto é a Programação Extrema que a seguir se define.

Sobre a Programação Extrema pode-se dizer que é uma metodologia ágil para equipas pequenas e médias. A equipa que adota esta metodologia sabe que irá desenvolver os seus softwares com requisitos vagos e que o seu trabalho desenvolvido estará sempre sujeito a mudanças. Sendo assim, é necessário ter um acompanhamento por parte do cliente e aplicar sempre pequenas mudanças ao longo do processo de desenvolvimento. Esta metodologia apresenta alguns princípios básicos. Resumidamente os principais são: o feedback rápido por parte do cliente, simplicidade no desenvolvimento de software, abertura a novas mudanças, efetuar as mesmas e por fim, apresentar um software com qualidade (Wikipédia, Programação Extrema, 2015).

3.2 Recursos

Passo aqui a tratar das tecnologias, softwares e hardwares a usar neste projeto.

- **Tecnologias**

NFC- Tecnologia que irá permitir efectuar os pagamentos comunicando entre smartphone/cartão com o recetor presente nas aplicações de aquisição de bens e de obtenção de senhas.

Sockets- Tecnologia usada para comunicação entre a aplicação do Android e o servidor.

- **Software**

As plataformas de programação escolhidas foram o Android Studio para criar a aplicação para os *smartphones* Android e o Netbeans para criar as aplicações de aquisição de bens, obtenção de senhas e servidor. As base de dados irão ser criadas em Wamp Server

O Netbeans é uma plataforma de desenvolvimento de software *open source* e sem custos para ser adquirido. Permite criar aplicações para computador, telemóvel e para a WEB programando em C#, JAVA, C++, HTML5, PHP e C.

Sobre o Android Studio, é uma plataforma para desenvolver aplicações para dispositivos que usem o Android como sistema operativo.

- **Hardware**

ACR122U – Receptor NFC.

Samsung Galaxy Grand Prime – smarthone com NFC e com a versão do Android 4.3 (mínimo necessário).

3.3 Atores e funcionalidades

Respetivamente aos atores das aplicações, podemos diferenciar dois tipos: Utilizador/Cliente e o funcionário do Instituto.

O utilizador trata-se do individuo que utiliza a aplicação para *smarthpones* e usufrui da aplicação de obtenção de senhas e de pagamentos por NFC. O funcionário do instituto é quem utiliza a aplicação de aquisição de bens para registo de bens requeridos pelo utilizador da aplicação. Neste caso o nome mais correto para utilizador da aplicação seria cliente, pelo que, no que toca à aplicação de aquisição de bens, irei referir o termo cliente em vez do anterior.

No que toca à questão de funcionalidades do utilizador da aplicação, este deve:

- Consultar os seus movimentos;
- Consultar ementas;
- Requisitar senhas;
- Ao colocar o seu *smartphone* sobre recetor NFC deve poder efetuar pagamentos ou obter a senha (se já foi previamente requisitada);

Em relação ao funcionário do instituto, este possui funcionalidades mais contextualizadas no âmbito de registo de bens pedidos pelo cliente.

3.4 Android

Nesta fase do projeto trato da conceção e análise de requisitos da aplicação para smartphones Android.

3.4.1 Análise de requisitos

A análise de requisitos é uma documentação que analisa e verifica os requisitos de um sistema e as suas restrições. Os requisitos expressam as necessidades dos clientes e as características que o software tem de cumprir. Podemos encontrar uma explicação mais clara em (Maria Clara Silveira, 2013)

Para a implementação deve-se salientar que, sempre que existe um pedido de dados ou confirmação por parte da aplicação, esse pedido é enviado ao servidor e são tratadas por este. Por fim o servidor envia a resposta ou dados sendo estes recebidos pela Async Task Cliente que as interpreta e trata.

O utilizador deve:

- Efetuar o seu login na aplicação;
- Apos o login ser efetuado, o utilizador deve ter ao seu dispor um menu com opções de consulta de movimentos, ementas/senhas disponiveis, horários escolares, horários de avaliações, ativação de pagamentos por NFC e por fim, opção de logout;
- Relativamente às ementas, o utilizador ao selecionar o dia desejado, deve poder consultar a ementa desse dia e o preço da compra de senha e por fim deve ter uma opção de compra de senha onde de seguida o utilizador escolhe se quer caarne ou peixe como prato;
- O utilizador ao escolher um movimento realizado, deve poder consultar os detalhes desse movimento;

- O utilizador ao selecionar a opção de visualiazação de horários escolares ou de avaliações, deve poder realizar o download do que pretende;
- Quando selecionada a opção ativação de pagamentos por NFC, o utilizador deve dispor de uma opção de envio de código para o seu email e outra opção para inserção do código recebido para poder ativar a opção de pagamentos.

3.4.2 Casos de uso

O diagrama de caso de uso, como podemos ver na Figura 5, descreve a funcionalidade proposta para um novo sistema que será projetado. Apartir deste diagrama podemos levantar requisitos funcionais do sistema e descrever as suas funcionalidades.

O utilizador desta aplicação deve poder efetuar o seu login e receber as suas informações pessoais. A aplicação deve fornecer ao utilizador funcionalidades como: visualização dos movimentos, consulta de ementas, compra de senhas e pagamentos por NFC. Tendo em conta estas funcionalidades o diagrama de casos de uso, ilustrado na Figura 5, é criado.

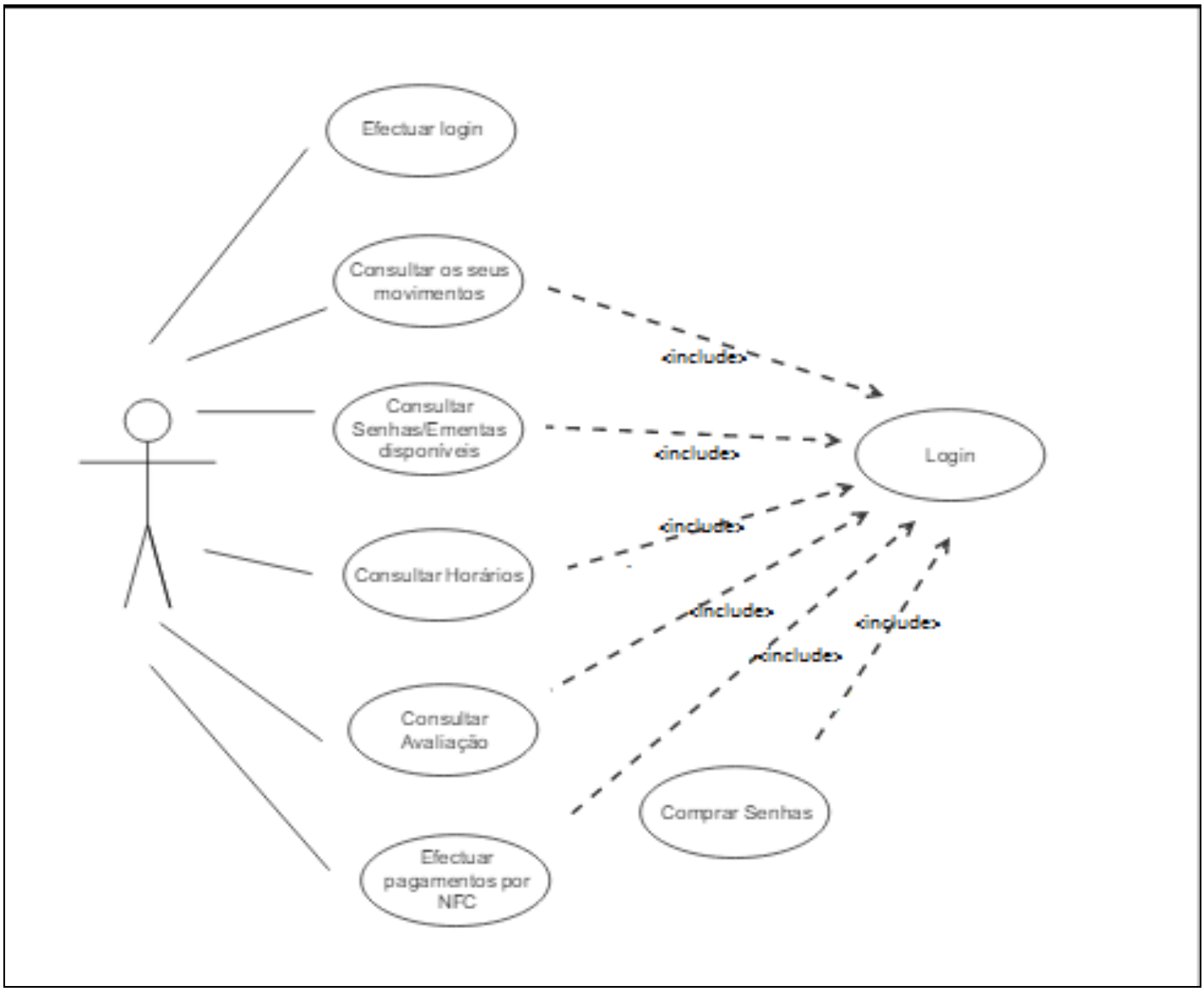

Figura 3: Diagrama de casos de usos da aplicação

3.4.2.1Descrição dos casos de uso e diagramas de sequência

A descrição dos casos de uso irá permitir relacionar os diferentes componentes a atuar no sistema (para um caso de uso referido) e a maneira como estes se comportam uns com os outros e com os pedidos e ações do utilizador.

Efetuar Login

O objetivo principal aqui, é o utilizador validar as credenciais inseridas por ele próprio.

Nome	Efetuar Login
Objetivo	O utilizador efetuar o seu login na aplicação
Ator	Utilizador
Pré-Condição	-
Prioridade	Alta
Fluxo Principal	1. O utilizador inicia a aplicação. 2. O sistema inicia a atividade de Login. 3. O utilizador insere as suas credenciais e carrega no botão login. 4. O sistema envia os dados ao servidor. 5. O servidor recebe os dados. 6. O servidor envia a resposta para a aplicação a) Pesquisa na base de dados se os dados estão corretos- b) Envia a resposta da verificação das credenciais. a) Se as credencias estiverem, corretas envia o nome, número e o curso do utilizador juntamente com essa verificação. 7. A AsyncTask/sistema recebe os dados enviados do servidor a) Se a informação do aluno estiver correta inicia a atividade menu se não mostra mensagem de erro.
Fluxos alternativos	-
Fluxo de exceção	4,5,6,7 a) Não existe ligação ao servidor. 6 a) Não existe ligação à base de dados.
Pós-Condição	Login efetuado, Atividade menu iniciada

Casos de teste	-

O seguinte diagrama de sequências, resultante da descrição do caso de uso anterior, ilustra a sequência de eventos após a iniciação da aplicação até se receber a mensagem de validação de login e, relaciona de forma gráfica os diferentes elementos presentes neste caso de uso.

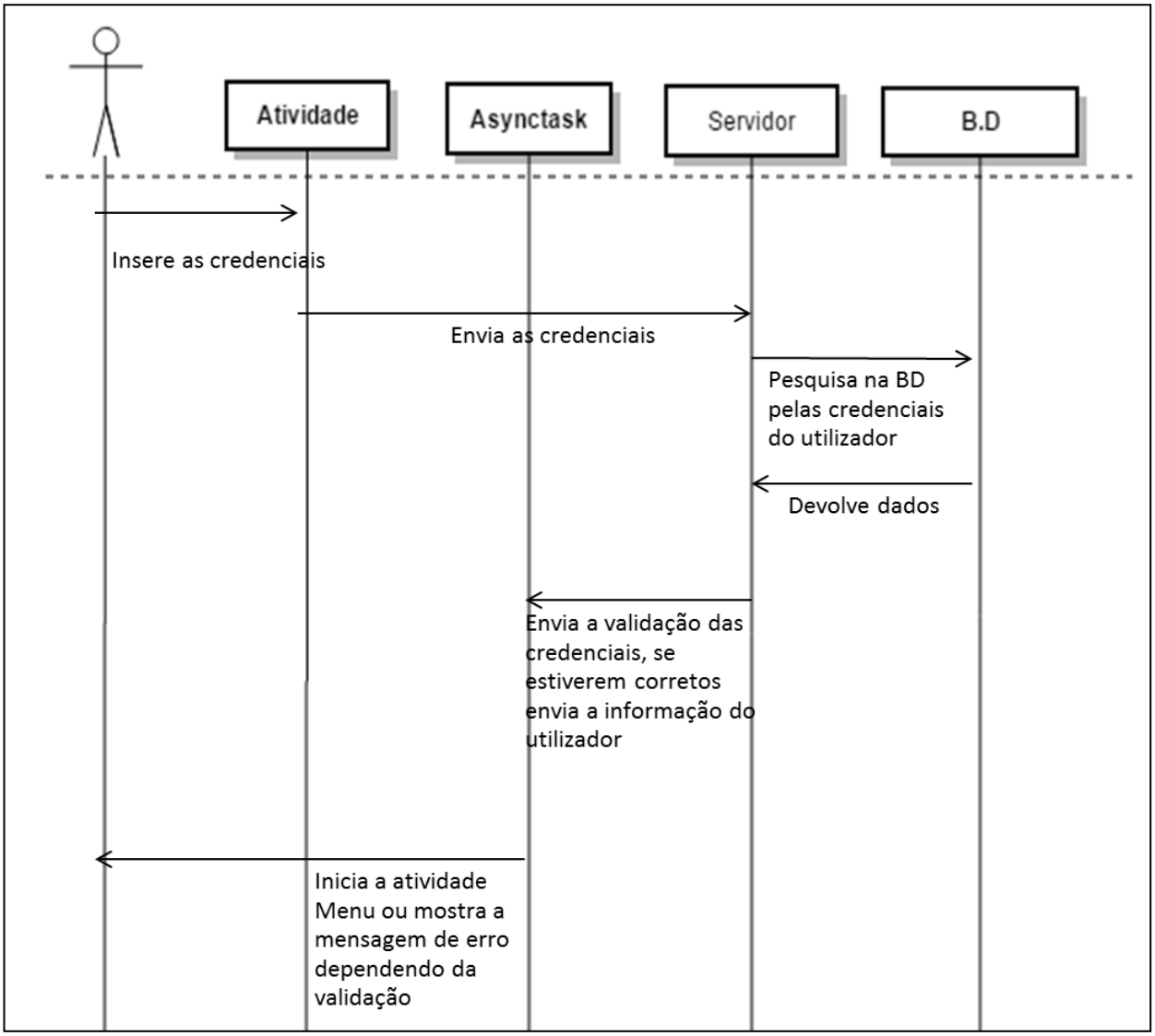

Figura 4: Diagrama de sequencias para login

Compra de Senhas

A descrição do caso de uso seguinte retrata o módulo de registo/compra de senha online.

Nome	Compra de senhas
Objetivo	Registar uma senha do aluno
Ator	Utilizador
Pré-Condição	Login efetuado, opção de consultar ementa escolhida
Prioridade	Normal
Fluxo Principal	1. O utilizador escolhe o dia que deseja comprar a senha. 2. O sistema disponibiliza a informação sobre o dia escolhido apresentando o prato de carne, peixe, sopa, sobremesa e valor a pagar. 3. O utilizador carrega no botão comprar. 4. O sistema mostra uma mensagem de escolha de prato(peixe ou carne) 5. O utilizador escolhe o prato pretendido. 6. O sistema envia o id da senha e o prato escolhido para o servidor. 7. O servidor insere na base de dados a senha comprada pelo utilizador da aplicação retira o saldo e envia uma mensagem indicando o estado da compra (efetuado ou não) para a aplicação 8. A AsyncTask lê a informação recebida e mostra uma mensagem ao utilizador com o estado de compra.
Fluxos Alternativos	-
Fluxo de exceção	6,7,8 a) Não á comunicação entre as aplicações. 7 a) O utilizador não tem saldo b) Erro de ligação à base de dados.
Pós-Condição	Nada
Casos de teste	-

O diagrama de sequências, ilustrado na figura seguinte, resulta da descrição do caso de uso anterior. Mais uma vez identificamos os componentes: servidor, AsyncTask, base de dados online e o ator.

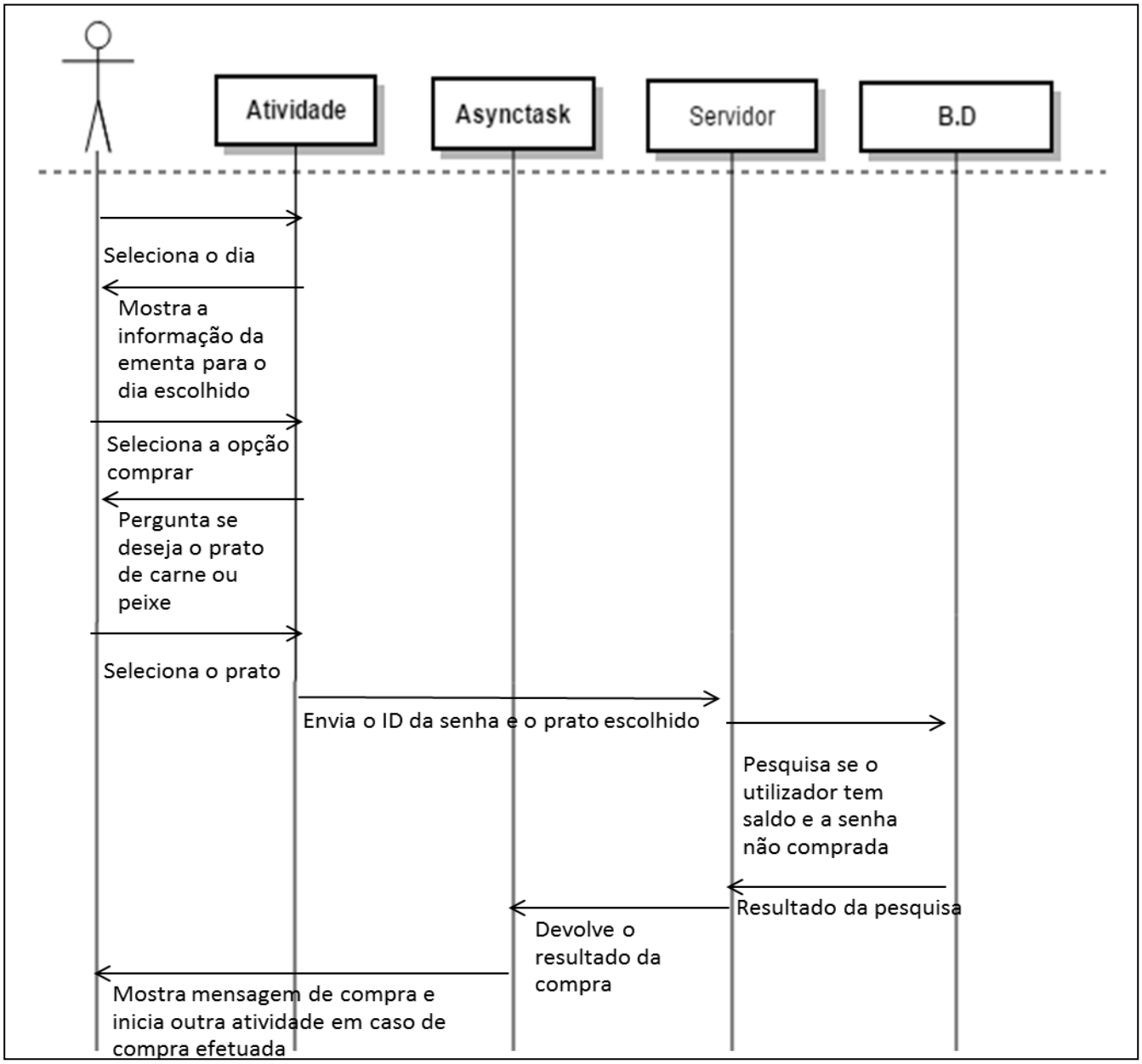

Figura 5: Diagrama de sequencias para compra de senhas

Consultar ementa

A descrição do caso de uso seguinte descreve o caso de consulta de ementa que é uma opção/funcionalidade oferecida pela aplicação ao utilizador.

Nome	Consultar Ementas
Objetivo	Mostras as ementas disponíveis
Ator	Utilizador
Pré-Condição	Login efetuado, estar no Menu
Prioridade	Normal
Fluxo Principal	1. O utilizador escolhe a opção ementas. 2. O sistema envia os dados ao servidor. 3. O servidor recebe os dados. 4. O servidor envia a resposta para a aplicação a) Pesquisa na base de dados as ementas que foram registadas e que ainda podem ser obtidas como senha pelo utilizador. b) Envia uma mensagem para a aplicação com as ementas dos dias. 5. A AsyncTask recebe os dados e inicia a atividade de consulta de ementas. 6. O utilizador escolhe o dia. 7. O sistema mostra a ementa do dia escolhido iniciando outra atividade.
Fluxos alternativos	-
Fluxo de exceção	2,3,4,5 b) Não á comunicação entre as aplicações. 4 c) Erro de ligação à base de dados.
Pós-Condição	Nada
Casos de teste	-

3.4.3 Base de dados

Base de dados uma coleção de dados guardados de forma segura que se encontram organizados e interligados, de forma, a permitir métodos eficazes de manutenção e consulta. Existindo uma explicação mais correta em (José Carlos Fonseca, 2013).

Em termos aplicacionais, a base de dados presente nesta aplicação serve para guardar os dados recebidos do servidor, que irá pertencer ao IPG, para uso futuro por parte do utilizador. Sem a criação da base de dados, não seria possível obter o número de aluno do utilizador para enviar ao recetor e o utilizador teria de inserir sempre as suas credenciais para efetuar o seu login.

3.4.3.1Diagrama de Classes

O principal objetivo do diagrama de classes é permitir a visualização das classes, os seus respectivos atributos e métodos e demonstrar, como se relacionam entre si numa aplicação, sendo esta uma resumida definição do Modelo ER fornecida por (Devmedia, 2016).

O Diagrama de Classes para a base de dados da aplicação pode ser consultada na figura 8.

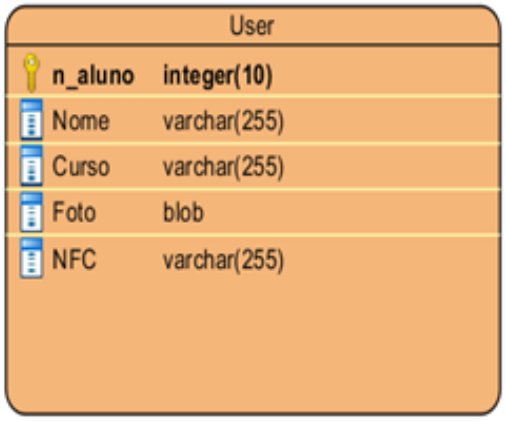

Figura 6: Diagrama de classes Android

3.4.3.2 Dicionário de Dados

Um dicionário de dados é uma documentação onde se descreve as entidades, atributos e relacionamentos presentes numa Base de Dados. Pode-se obter uma explicação mais clara em (José Carlos Fonseca, 2013).

Nas tabelas 3,4 e 5 retrato as entidades envolvidas no Modelo ER ilustrado na figura 8.

Tabela 3: Tabela utilizadores

Tabela User			
Campo	Tipo de Dados	Descrição	Observações
N_aluno	INTEGER	Número do aluno	Chave Primária
Nome	Text	nome do Aluno	Não Nulo
Curso	Text	Curso do Aluno	Não Nulo
Foto	BLOB(10)	Foto do Aluno	Não Nulo
NFC	BOOLEAN	Estado de Ativação	Não Nulo

Tabela 4: Tabela Horários

Tabela horários			
Campo	Tipo de Dados	Descrição	Observações
Id_server	INTEGER	Id recebido pelo servidor	Chave Primária
Ficheiro	BLOB	ficheiro do horário	Não Nulo

Devo frisar que o id_server corresponde ao id do horário da base de dados online (Base de dados dos alunos).

.

Tabela 5: Tabela de Avaliações

Tabela Avaliações			
Campo	Tipo de Dados	Descrição	Observações
Id_server	INTEGER	Id recebido pelo servidor	Chave Primária
Ficheiro	BLOB	ficheiro do horario	Não Nulo

3.4.4 Diagrama de componentes

Um Diagrama de componentes ilustra como as classes se devem organizar através da noção de componentes de trabalho. Permite descrever partes do software, que podem ser código fonte, bibliotecas ou programas executáveis. Para uma melhor descrição, pode-se consultar (Maria Clara Silveira, 2013) e (IBM, Diagrama de Componentes, 2016).

Os componentes para a aplicação são:

- **AsyncTask Cliente:** Thread de comunicação com o servidor e responsável por iniciar certas atividades.
- **Login, Menu, Senhas, Movimentos, Ativação do NFC, Detalhes das Senhas, Detalhes dos movimentos:** Atividades da aplicação.
- **User:** Classe para guardar informações do utilizador.
- **ACR122U:** Recetor NFC ligado às aplicações de compra e de obtenção de senhas.

- **HostApduService:** Serviço da aplicação que comunica com o recetor NFC.
- **Servidor:** Componente responsável por enviar as informações pedidas pela aplicação.

Relacionando os componentes anteriormente referidos obtém-se o seguinte Diagrama de componentes.

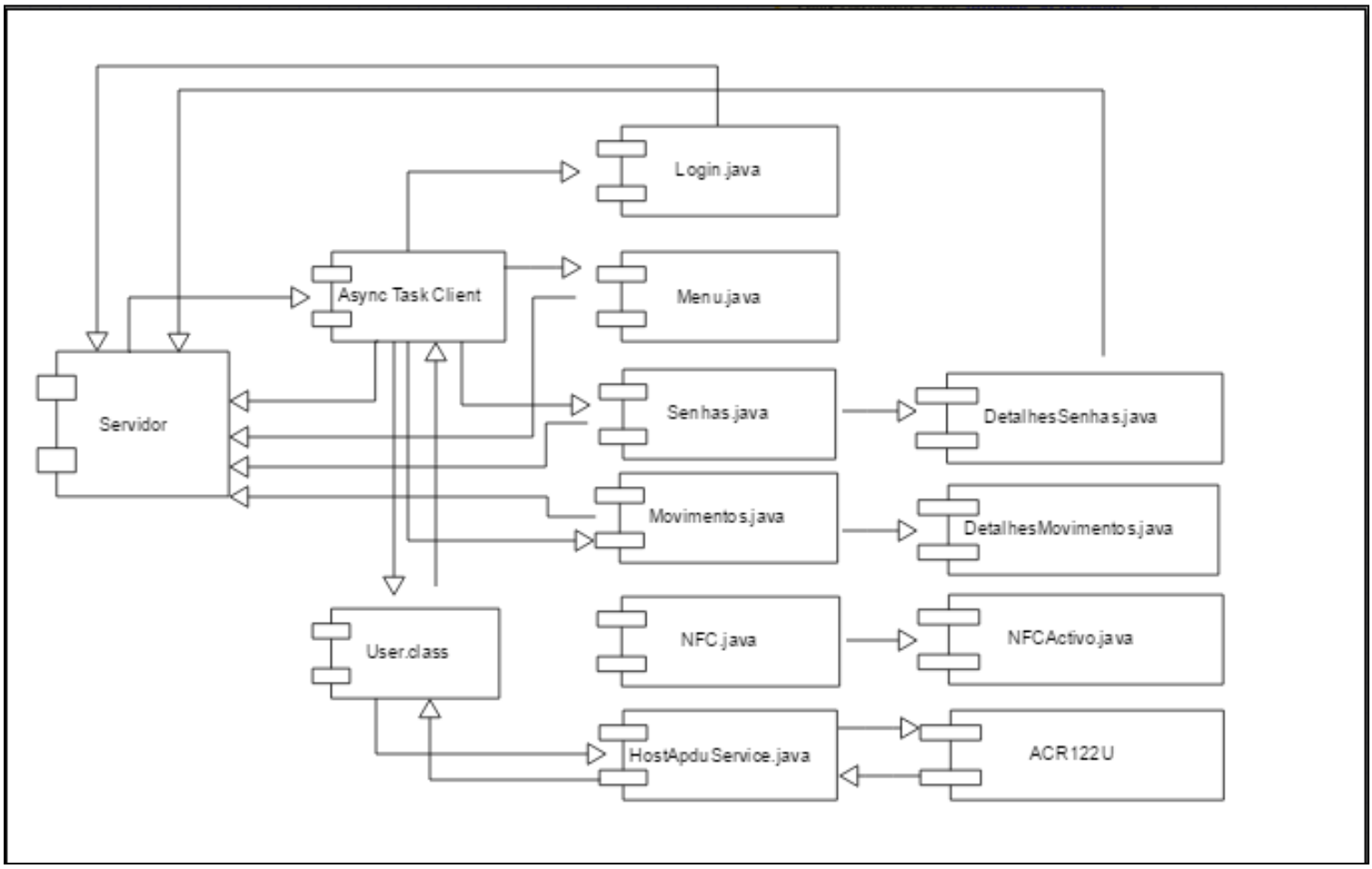

Figura 7: Diagrama de componentes da aplicação

3.4.5 Diagrama de Estados do HCE e recetor

Um diagrama de estado representa as mudanças de estado que um objeto pode ter ao longo de uma atividade/processo. Este diagrama mostra o comportamento do objeto e as transações que influenciam a sua mudança. Para uma melhor explicação pode consultar (Wikipédia, Diagrama de estados, 2016).

Sobre a questão de pagamentos por NFC, quando da parte da aplicação do recetor existe um pedido de dados do utilizador, uma resposta é enviada da aplicação para o recetor. A resposta é modicada pelos seguintes fatores: O número de aluno e o estado da ativação de pagamentos por NFC da aplicação. Para se enviar a resposta tem de se obter o número de aluno presente na base de dados da aplicação e verificar se o utilizador tem a opção de pagamentos ativa e só por fim, é que a resposta é enviada.

Nas figuras seguintes encontram-se o diagrama de estados para pagamentos por NFC para este projeto. A primeira figura (Figura 6) retrata a parte da aplicação android e a segunda figura (Figura7) retrata o recetor.

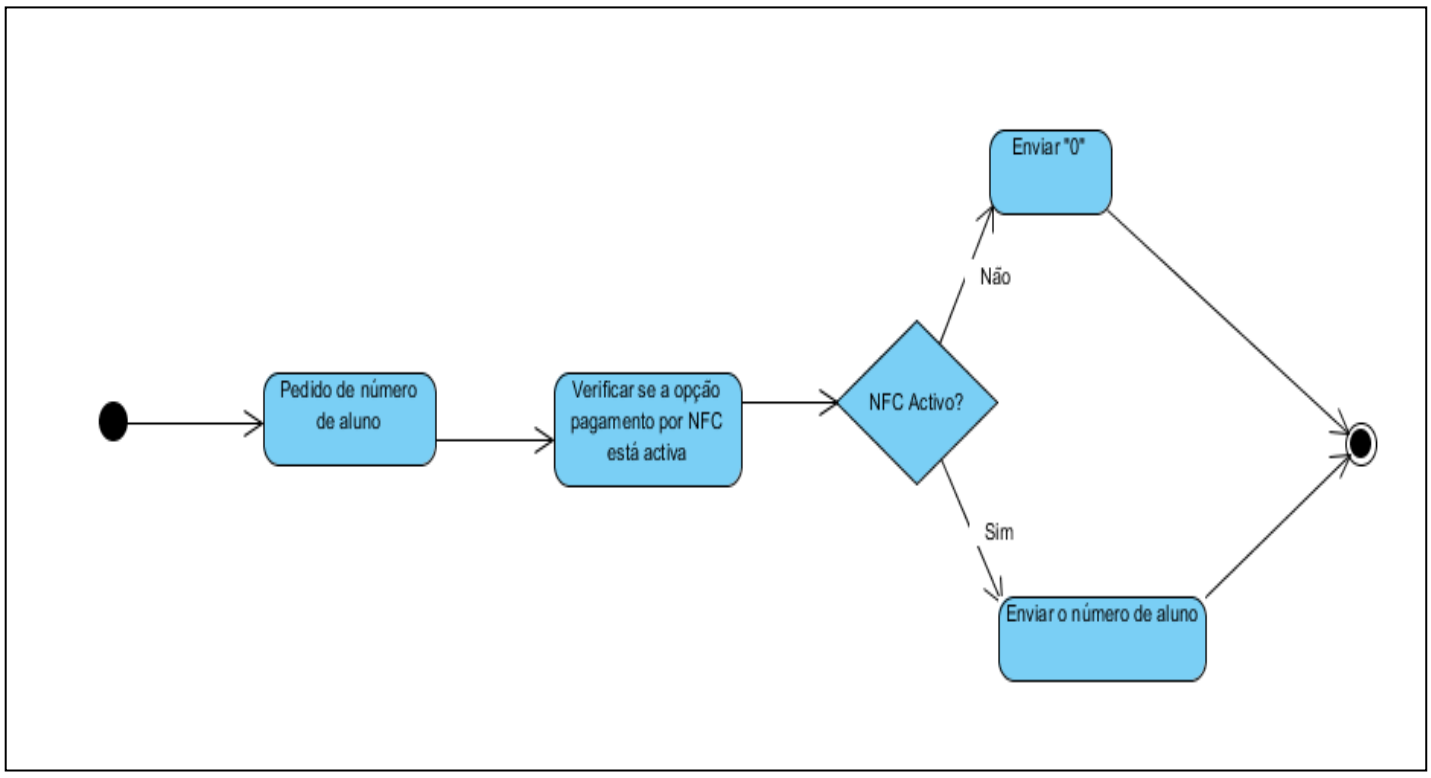

Figura 8: Diagrama de estados HCE para a aplicação

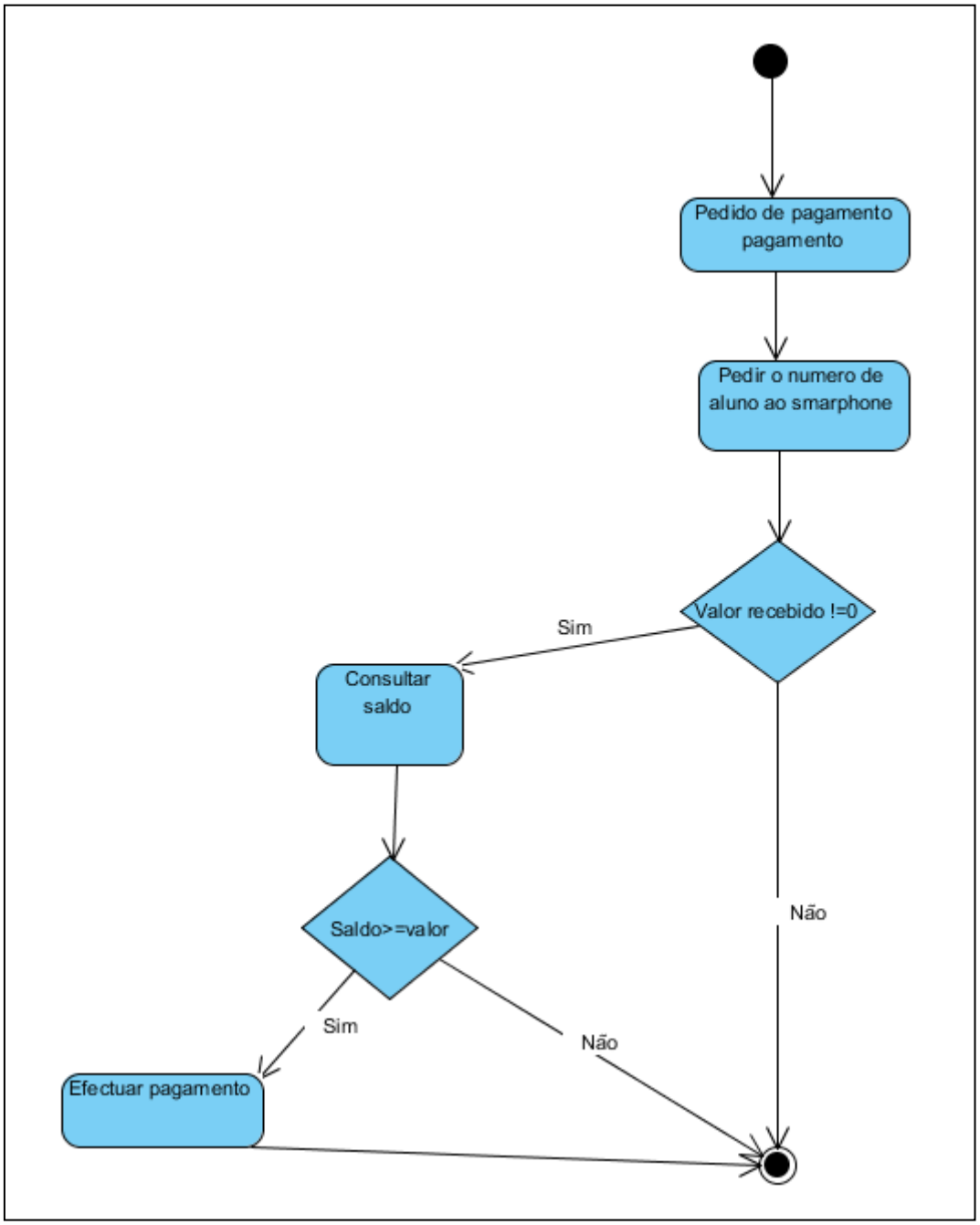

Figura 9: Diagrama de estados para a aplicação do recetor

3.4.6 Diagrama de Navegação

Os diagramas de navegação fornecem uma visualização de um elemento de contexto especificado e são semelhantes, em termo de funcionalidade a um navegador da web. Um diagrama de navegação guarda o histórico do que foi visualizado pelo utilizador sendo dessa forma possível, a navegação para páginas anteriormente visualizadas. Uma definição mais pormenorizada pode encontra-se em (IBM, Diagrama de Navegação, 2016).

No que diz respeito a aplicações para android, convém sublinhar que as páginas são chamadas de Atividade. Enumerando as atividades do Android para serem desenvolvidas obtemos:

- **Login:** Atividade inicial onde o utilizador insere as suas credenciais.
- **Menu:** Atividade onde é apresentado as opções que o utilizador tem.
- **Horários, Movimentos, Senhas, Avaliações:** Atividades onde são apresentados certos dados ao utilizador.
- **NFC:** Atividade onde o utilizador ativa a opção de pagamentos por NFC
- **Logout:** Terminar a “sessão” e eliminação de dados.

O diagrama de navegação que resulta das atividades anteriormente referidas pode ser visualizado na Figura 3.

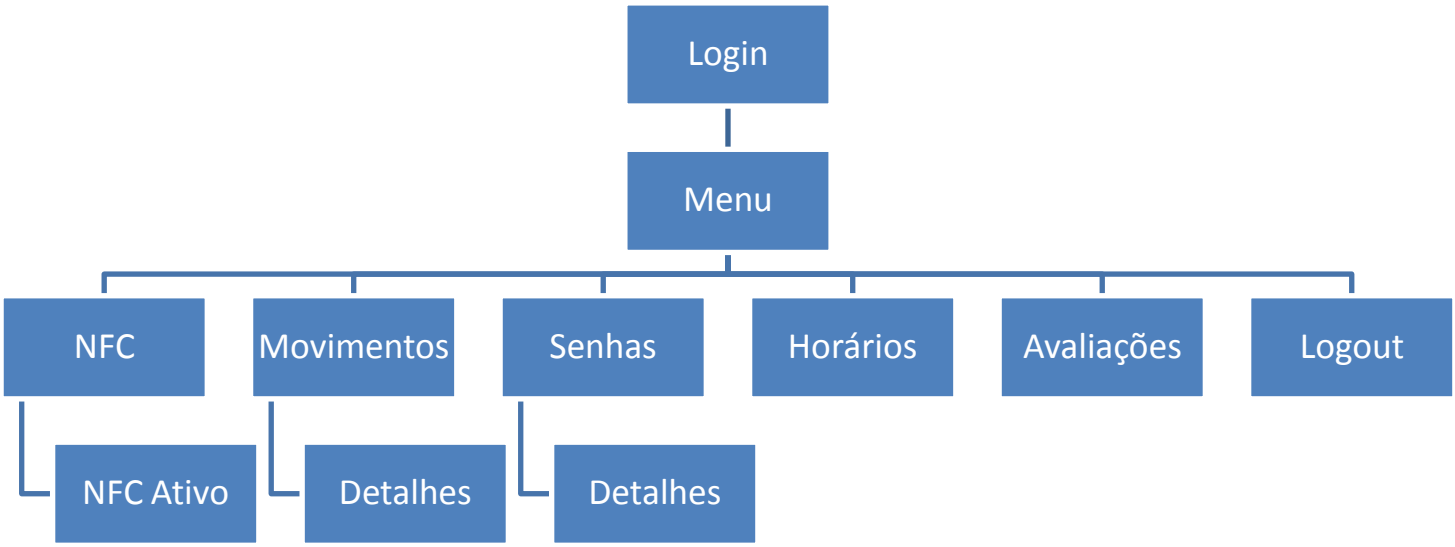

Figura 10:Diagrama de páginas Android

3.5 Aplicação de compra de bens e obtenção de senhas

Neste subcapítulo passo a tratar da conceção e analise das aplicações de compra de bens e de obtenção de senhas. Devo salientar que as aplicações de computador funcionam em conjunto com a aplicação para *smartphones*. Trato aqui da base de dados e análise de requisitos para as aplicações de compra e de obtenção de senha.

3.5.1 Análise de requisitos

Antes de tudo é preciso salientar os sujeitos destas aplicações. Sendo assim, os intervenientes desta são o funcionário do instituto e o cliente.

O funcionário é a pessoa que inicia o processo de pagamento e insere os produtos para registo da compra.

O Cliente é aquele que requere os produtos ou senhas e que efetua o pagamento por dinheiro, cartão ou smartphone.

Quando se é utilizado o termo pagamento significa que uma pesquisa à base de dados “Alunos” terá de ser efetuada para se poder verificar se o Cliente que está a efetuar o pagamento pode ou não efetua-lo. Engloba ações como verificar se o cliente tem saldo disponível para pagamento e retirar ao saldo o valor deste.

- **Aplicação de compras**

Nesta aplicação deve ser apresentado uma solução de fácil entendimento e uso. O funcionário deve dispor de opções de visualização de produtos por categorias e por produtos mais vendidos. O nome e o valor do produto devem ser apresentados antes de este mesmo ser selecionado de forma a se obter

maior rapidez de comunicação cliente-funcionário (pensando um pouco nos pedidos de preços dos produtos por parte do cliente). O funcionário do instituto deve ter:

- Visualização constante sobre os produtos pedidos por parte do Cliente pelo que é necessário implementar uma lista de compras na aplicação.
- Opção de apagar um produto que se encontra na lista de compras.
- Opção de incrementar ou decrementar uma unidade à quantidade de um produto na lista.
- Limpar a lista por completo.
- Caso um produto não tenha sido registado, o funcionário deve ter a opção de colocar um produto desconhecido e o seu preço.
- Opção para pagamento.
- Quando a opção de pagamento é escolhida o funcionário deve ter visível as opções pagamento por dinheiro, cartão ou sair das formas de pagamento.

Por parte do cliente, este apenas tem de efetuar o pagamento utilizando um cartão previamente programado, um smartphone, utilizando os dois primeiros sobre um recetor NFC. Esta aplicação deve obter informação da base de dados “Compras” e quando o pagamento que está a ser efetuado, deve-se obter informação da base de dados “Alunos”.

- **Aplicação de obtenção de senhas**

Esta aplicação é destinada apenas ao cliente. O cliente ao passar o cartão ou o smartphone por cima do recetor NFC deve poder obter a senha do dia em questão se, e só se, foi previamente adquirida pela aplicação para smartphones. A informação é obtida através da base de dados “Alunos”.

3.5.2 Base de dados

Nesta fase do projeto tenho de referenciar 2 bases de dados distintas. A primeira base de dados trata-se de um repositório onde se guarda as informações relativas aos produtos e bens da qual a aplicação de compras obtém, altera e adiciona informação. A segunda base de dados trata-se de uma coleção de dados que é utilizada pelo servidor, aplicação de obtenção de senhas e por fim, no que diz respeito aos pagamentos por NFC (pesquisa da identificação do aluno, subtração de saldo e inserção de movimentos). A primeira base de dados encontra-se localmente, ou seja, no computador. A segunda é online.

3.5.2.1Módulo de compras

- **Diagrama de Classes**

A figura seguinte retrata o Diagrama de classes para o módulo de compras.

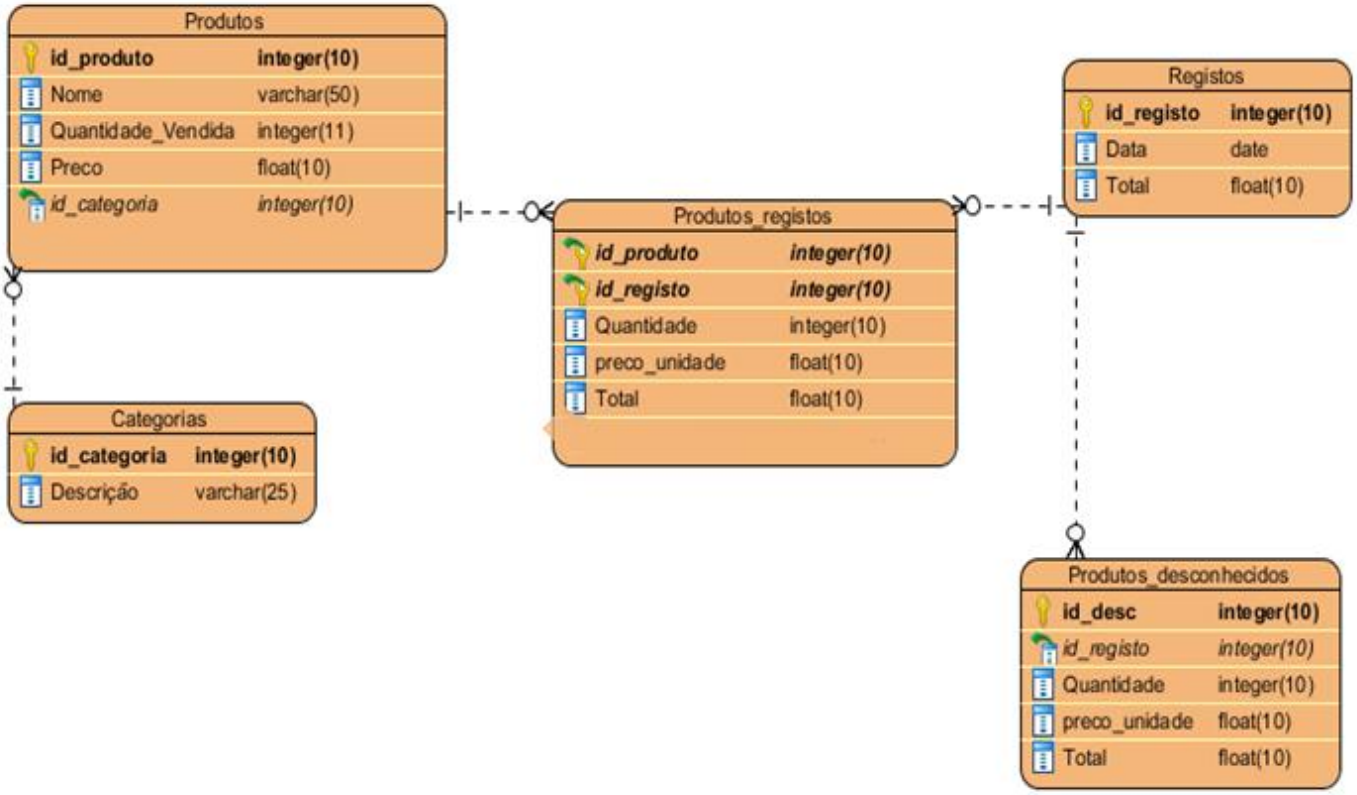

Figura 11: Diagrama de Classes do modulo de compras

- **Dicionário de dados**

Tabela 6: Tabela Categorias

Tabela categorias			
Campo	**Tipo de Dados**	**Descrição**	**Observações**
Id_categoria	INTEGER(11)	Id da categoria	Incremento automático, Chave Primária
Descrição	VARCHAR(25)	Nome da categoria	Não Nulo

Tabela 7: Tabela Produtos

Tabela Produtos			
Campo	**Tipo de Dados**	**Descrição**	**Observações**
Id_produto	INTEGER(11)	Id do produto	Incremento Automático, Chave Primária
Nome	VARCHAR(50)	Nome do produto	Não Nulo
Quantidade_Vendida	INTEGER(11)	Quantidade vendida	Não Nulo
Preço	FLOAT	Preço do produto	Não Nulo
Id_categoria	INTEGER(11)	Id_categoria	Não Nulo

Tabela 8: Tabela produtos_registos

Tabela Produtos_registos			
Campo	**Tipo de Dados**	**Descrição**	**Observações**
Id_produto	INTEGER(11)	Id do produto	Incremento Automático, Chave Primária
Id_registo	INTEGER(11)	Id do registo	Não Nulo
Quantidade	INTEGER(11)	Quantidade do produto	Não Nulo
Preço_unidade	Float	Preço unitário	Não Nulo
Total	Float	Total da compra	Não Nulo

Tabela 9: Produtos desconhecidos registados

Tabela PROD_DESC_REGISTOS			
Campo	**Tipo de Dados**	**Descrição**	**Observações**
Id_desc	INTEGER(11)	Id do desconhecido	Incremento Automático, Chave Primária
Id_registo	INTEGER(11)	Id do registo	Não Nulo
QUANTIDADE	INTEGER(10)	Quantidade	Não Nulo
Preço_unidade	float	Preço da unidade	Não Nulo
total	float	Total da compra	Não Nulo

Tabela 10: Tabela Registos

Tabela registos			
Campo	**Tipo de Dados**	**Descrição**	**Observações**
Id_registo	INTEGER(11)	Id do registo	Incremento Automático, Chave Primária
Data	Datetime	Data da compra	Não Nulo
Total	Float	Total da compra	Não Nulo

3.5.2.2 Online (informações dos alunos)

Nesta base de dados encontram-se as informações dos alunos que irão ser usadas pelo servidor, aplicação de compras (no caso de a compra ser efetuada por cartão ou smartphone) e pela aplicação de obtenção de senhas.

- **Diagrama de Classes**

A figura seguinte ilustra o Diagrama de Classes para a base de dados online.

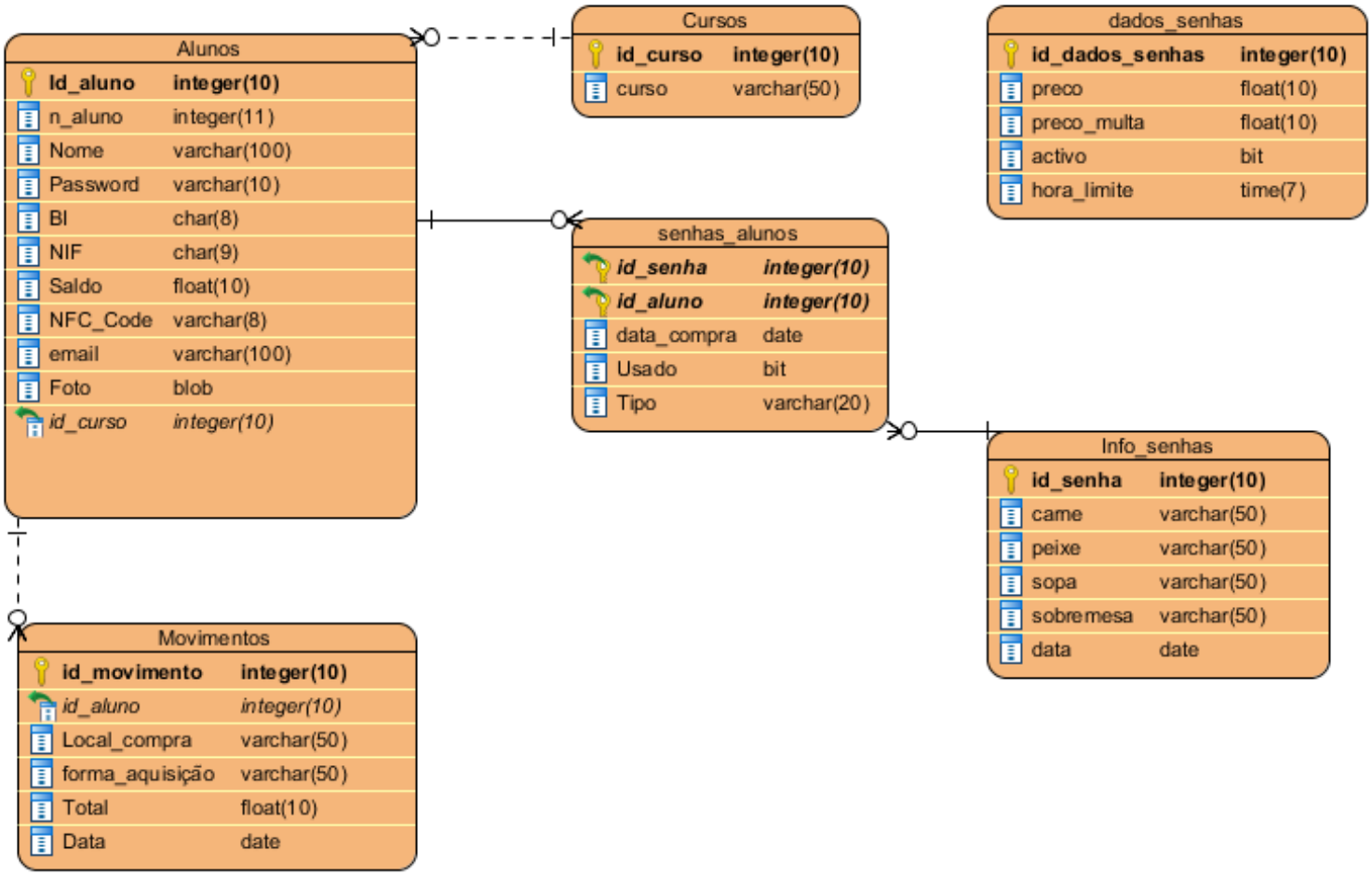

Figura 12: Diagrama de Classes da base de dados Alunos (Online)

- **Dicionário de Dados**

Tabela 11: Tabela Alunos

Tabela Alunos			
Campo	Tipo de Dados	Descrição	Observações
Id_aluno	INTEGER(11)	Id do aluno	Incremento Automático, Chave Primária
N_aluno	VARCHAR(11)	Número do Aluno	Não Nulo
Nome	VARCHAR(100)	Nome do Aluno	Não Nulo
Password	VARCHAR(10)	Password do Aluno	Não Nulo
BI	CHAR(8)	Bilhete de Identidade	Não Nulo
NIF	CHAR(9)	Número de Identificação Fiscal	Não Nulo
Curso	INTEGER(11)	Curso do Aluno	Não Nulo,Chave Estrangeira
Saldo	FLOAT	Saldo do Aluno	Não Nulo
NFC_Code	VARCHAR(8)	Código de Ativação	Não Nulo
Email	VARCHAR(100)	Email do Aluno	Não Nulo
Foto	LONGBLOB	Foto do Aluno	Não Nulo

Tabela 12: Tabela Cursos

Tabela Cursos			
Campo	Tipo de Dados	Descrição	Observações
Id_curso	INTEGER(11)	Id do curso	Incremento Automático, Chave Primária
Curso	VARCHAR(50)	Nome do Curso	Não Nulo

Tabela 13: Tabela de informações sobre senhas

Tabela Info_senhas			
Campo	Tipo de Dados	Descrição	Observações
Id_senha	INTEGER(11)	ID da senha	Incremento automático, Chave Primária
Carne	VARCHAR(50)	Prato de Carne	Não Nulo
Peixe	VARCHAR(50)	Prato de Peixe	Não Nulo
Sopa	VARCHAR(50)	Sopa	Não Nulo
Sobremesa	VARCHAR(50)	Sobremesa	Não Nulo
Data	DATE	Data da senha	Não Nulo

Tabela 14: Tabela Movimentos

Tabela Movimentos			
Campo	Tipo de Dados	Descrição	Observações
Id_movimento	INTEGER(11)	Id do Movimento	Incremento Automático, Chave Primária
Id_aluno	Integer(11)	Id do Aluno	Não Nulo
Local_compra	VARCHAR(50)	Local do Movimento	Não Nulo
Forma_aquisição	VARCHAR(50)	(Cartao, Smartphone)	Não Nulo
Total	FLOAT	Valor total do movimento	Não Nulo
Data	DATETIME	Data e Hora	Não Nulo

Tabela 15: Tabela preço das senhas

Tabela preço Senhas			
Campo	Tipo de Dados	Descrição	Observações
Id_preço_senha	INTEGER(11)	Id do Preço da Senha	Incremento Automático, Chave Primária
Preço	FLOAT	Preço da senha	Não Nulo
Preço_multa	FLOAT	Preço da senha com multa	Não Nulo
Activo	Tiny_int	Estado no Corrente Ano	Não Nulo
hora_limite	Time	Hora limite para se poder comprar a senha	Não Nulo

Tabela 16: Tabela relacional senhas e alunos

Tabela senhas_alunos			
Campo	Tipo de Dados	Descrição	Observações
Id_senha	INTEGER(11)	Id da senha	Incremento Automático, Chave Primária
id_aluno	INTEGER(11)	id do Aluno	Não Nulo, Chave Primaria
Data_compra	DATE TIME	Data da compra	Não Nulo
Usado	TINY_INT	Estado da Senha	Não Nulo
Tipo	VARCHAR(20)	Prato Escolhido	Não Nulo

4 Realização

Este capitulo apresenta a solução ao problema referido no capítulo anterior. Falo aqui como as aplicações foram criadas e como a aplicação para smartphones comunica com o servidor.

4.1 Servidor: comunicação com a aplicação

O servidor foi desenvolvido para comunicar com a AsyncTask do android. Com isto, quando existe um pedido por parte da aplicação, o servidor trata de processar a informação relacionada com o pedido e enviar de volta a resposta a esse pedido. Enumerando esses pedidos e processos envolvidos estão presentes:

- **Login**

  -Pesquisa na base de dados Online se as credenciais recebidas estão corretas.

  -Envia de volta para a aplicação android se as credenciais estão corretas.

  -Se estiverem corretas, envia os dados do aluno.

- **Senhas**

  -Pesquisa na base de dados Online as senhas disponíveis para compra e as que já foram obtidas pelo utilizador da aplicação, tendo em conta, a hora limite para obtenção da senha do dia atual.

  -Envia as informação relativas a essas senhas para a aplicação android.

- **Pedido de email**

  -Pesquisa na base de dados Online pelo email do utilizador da aplicação do android.

  -Envia um email para o utilizador com o código de ativação NFC do utilizador e devolve para a aplicação que foi enviado um email.

- **Ativar NFC**

  -Lê e confirma com auxílio da base de dados se os dados enviados pela aplicação estão corretos.

  -reenvia a confirmação.

- **Compra de senha**

  -Verifica se o utilizador da aplicação tem saldo para efetuar a compra e se a compra ainda não foi efetuada. Caso ambos se verifiquem, efetua a compra e devolve o resultado da compra(efetuada ou não).

  -Em caso de se tratar do dia em questão aplica a relativa multa e verifica se ainda não foi ultrapassado a hora limite.

- **Horários e avaliações**

  -Envia a descrição e a identificação dos horários e avaliações presentes na base de dados relativo ao curso do utilizador.

- **Logout**

  -termina a ligação.

4.2 Servidor: Downloads

Para não interromper a ligação entre o servidor e a aplicação, foi criado uma nova Thread para efetuar downloads da fotografia, horários e avaliações. Dependente do download pedido, efetua-se uma pesquisa à base de dados e envia-se os dados/ficheiro para a aplicação.

4.3 Android: AsyncTask comunicação ao servidor

Esta AsyncTask é responsável pelo funcionamento da aplicação sendo que, se não houver ligação ao servidor todas as mensagens enviadas pela aplicação ao servidor são impedidas. Se existir credenciais do utilizador registado na base de dados da aplicação a AsyncTask quando iniciada trata de enviar essas credencias ao servidor para serem validadas e iniciar a atividade relativa ao menu (login automático).

Quando informação por parte do servidor é recebida, a Asynctask interpreta os dados e inicia as atividades correspondentes aos dados recebidos:

- **Login**
  -se for validado, inicia a atividade menu, se não mostra uma mensagem de erro.

- **Movimentos**

  - Recebe e guarda os dados para serem posteriormente utilizados nas atividade relativas aos movimentos.

  -Inicia a atividade dos movimentos.

- **Senhas**

  -Guarda os dados para serem utilizados nas atividades relativas às senhas

  -Inicia a atividade das senhas.

- **Compra de senha**

  -Mostra uma mensagem de forma a esclarecer se a compra foi efetuada com sucesso.

- **Horários e avaliações**

  -Guarda os dados para serem utilizados

  -Inicia a atividade correspondente ao pedido.

- **Email**

  -Mostra uma mensagem a dizer que o email foi enviado

- **Ativar NFC**

  -Se a ativação for bem sucedida, mostra uma mensagem e insere na base de dados o estado de ativação(ativado ou não ativado).

4.4 Android: AsyncTask download

Esta AsyncTask tem como propósito realizar download da foto do utilizador, horários ou de avaliações. Quando é utilizada é necessário salientar o tipo de download a efetuar (foto, horário, avaliação) e o id da base de dados online correspondente. No caso da foto do utilizador será o número de aluno. Após iniciada envia para o servidor esses dados e espera pelo ficheiro por ser enviado por parte do servidor. Quando recebido guarda o ficheiro no smarphone (não na base de dados).

4.5 Android: Login

Esta atividade é o ponto de partida da aplicação desenvolvida para smartphones iniciando assim, a Thread de comunicação ao servidor do Android (AsyncTask) . Na figura seguinte pode-se observar a atividade em questão.

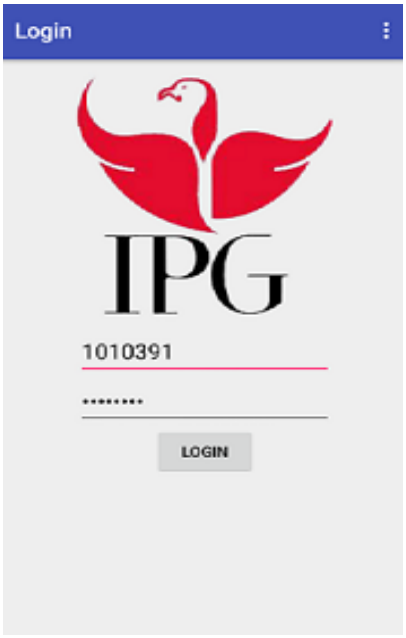

Figura 13: Atividade login

Aqui, o utilizador da aplicação insere as suas credenciais para efetuar o login (numero de aluno e password). Após isso os dados são enviados para o servidor que irá validar as credenciais enviadas e por fim entregar uma resposta à aplicação (Login efetuado ou recusado).

4.6 Android: Menu

Assim que esta atividade é iniciada guarda os dados do utilizador na base de dados. Caso o download da fotografia ainda não se tenha efetuado, inicia o download da mesma. Se já tiver sido efetuado mostra a fotografia do utilizador. Quanto à informação disponível nesta atividade, encontra-se o número de aluno, nome e curso. Apresenta-se ainda um menu com várias opções. As opções desse menu e as suas funcionalidades são as seguintes:

- Movimentos, Senhas, Horários, Avaliações
  -Envia uma mensagem para o servidor a pedir a informações sobre a opção escolhida.

- NFC
  -Se o utilizador já ativou a opção de pagamentos por NFC mostra uma mensagem a informá-lo, se não inicia a atividade NFC.

- Logout
  -Elimina os dados da base de dados da aplicação do android.
  -Enviar uma mensagem para o servidor para efetuar o Logout.
  -Inicia a atividade Login.

As seguintes figuras representam a atividade menu com e sem fotografia do utilizador.

Figura 15: Atividade menu com foto

Figura 14: Atividade menu sem foto

4.7 Android: Movimentos

Nesta atividade são apresentados os 10 últimos movimentos do utilizador. A atividade criada encontra-se na figura seguinte.

Figura 16: Atividade dos movimentos

Os movimentos são apresentados numa listview dispondo de informações como: local de compra, valor, data e forma de aquisição. A forma de aquisição é disposta sobre forma de imagem. O resto da informação encontra-se em texto. Apenas os dez últimos movimentos são apresentados. Ao escolher um movimento, uma nova atividade irá ser apresentada ao utilizador, ver figura que se segue.

Figura 17: Movimentos detalhados

Aqui, a informação sobre o movimento escolhido é demonstrado de forma mais detalhada e de forma individual. É possível visualizar o local, a data, hora, valor e forma por texto.

4.8 Android: Horários e Avaliações

A atividade dos horários e a das avaliações são idênticas sendo que, a única forma de as diferenciar é o tipo de download pedido ao servidor. Na figura seguinte observa-se a atividade relativa aos horários.

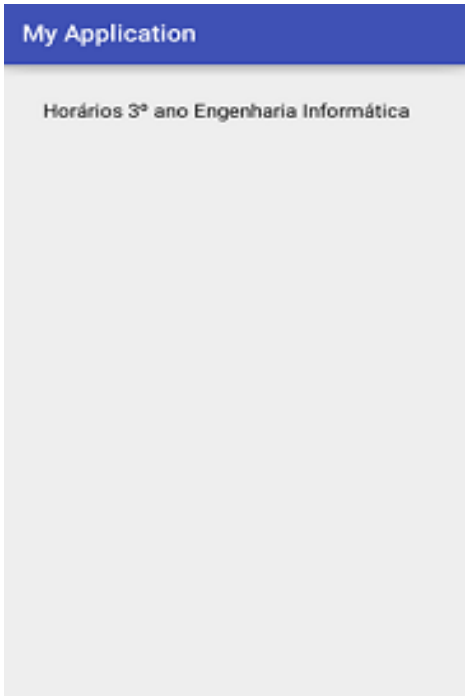

Figura 18: Atividade horários

Tanto os horários como as avaliações disponíveis para download encontram-se apresentados numa listview. Ao escolher o pretendido a Thread de download da aplicação(AsyncTask) é iniciada.

4.9 Android: Senhas

Nesta atividade são mostrados informações básicas das senhas . Apresenta o dia da semana, a data e por fim se a compra da senha já foi efetuada ou não utilizando uma imagem. Assim que o utilizador selecionar um dia, a atividade de senhas detalhada irá ser iniciada. A atividade dispõe ao utilizador a informação sobre a ementa do dia escolhido, e o preço. Ao pressionar o botão de compra envia-se uma mensagem ao servidor com a identificação da senha escolhida. Apenas são apresentadas as ementas que estão disponíveis na base de dados Online tendo em conta o dia atual. Exemplo: Se o dia em que for consultado for 02-12-2016 as ementas/senhas inseridos na tabela “info_senhas” cuja a “data” é superior 02-12-2016.

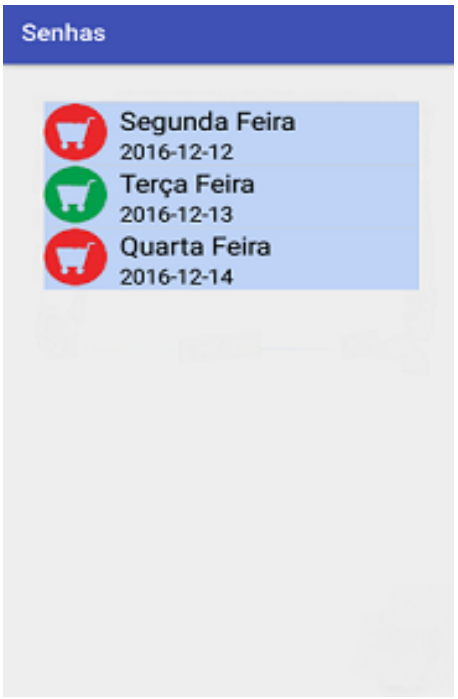

Figura 19: Senhas Disponíveis

Figura 20: Atividade para detalhe das senhas

4.10 Android: Ativar NFC

O utilizador aqui tem a opção de pedir que seja enviado um email para a sua conta de email ou pedir para confirmar se o código de ativação do NFC se encontra correto. Mais uma vez, ambas as opções são pedidas ao servidor. A atividade criada encontra-se na figura seguinte.

Figura 21: Atividade para ativar o NFC

4.11 Aplicação de compras

Esta aplicação destina-se tanto para obtenção de produtos no bar como na papelaria.

Trata-se de uma aplicação de registo de produtos/bens em que o funcionário do instituto regista estes mesmos consoante o pedido por parte do cliente. A aplicação encontra-se na figura seguinte.

Figura 22: Aplicação de compras

Assim que a aplicação é iniciada, uma pesquisa à base de dados da aplicação é efetuada com o objetivo de obter as categorias dos produtos inseridos na base de dados. As categorias são apresentadas no painel esquerdo da aplicação. A categoria “Favoritos” representa os produtos mais vendidos no local onde a aplicação está a funcionar. Ao pressionarmos um

botão de categoria, é pesquisado todos os produtos relativos à escolha feita e apresentado no painel do centro.

Relativamente ao painel do centro, cada botão presente apresenta a descrição do produto e o seu preço. Estes estão programados para registar os produtos na lista presente no painel do lado direito.

No Painel do lado direito, encontra-se a lista dos produtos registos para compra. Nessa mesma lista é possível observar a quantidade pedida, a descrição e o total relativo a cada produto. Apresenta-se também o valor total a pagar pelos produtos registados. Relativamente à lista, dispõe-se de da opção de limpar a lista totalmente, incrementar ou decrementar um produto selecionado na lista, inserir um valor (produto desconhecido) e por fim, efetuar o pagamento. Ao selecionar esta última opção, uma janela relativa ao pagamento é apresentado ao funcionário do instituto. A figura seguinte retrata a janela apresentada.

Figura 23: Painel de pagamento

Quando a janela relativa aos pagamentos é apresentada, a Thread de leitura de cartão é iniciada. Na janela dispõe-se de opções de pagamento por dinheiro, cartão e sair. O total a pagar é encontra-se no canto superior direito. Se nenhum cartão for inserido no recetor NFC, a opção pagamento por cartão não é possível ser concretizada.

Caso exista um cartão de aluno inserido no recetor NFC, uma pesquisa à base de dados online é efetuada com o propósito de obter o NIF, número de aluno e o nome do aluno. Na figura seguinte pode-se observar a janela de pagamento com o cartão do aluno presente.

Figura 24: Painel de pagamento com cartão presente

O NIF do cliente é pesquisado por questões de maior rapidez de registo de faturas.

Na escolha da opção pagamento por cartão, é verificado se o cliente dispõe de saldo suficiente para efetuar o pagamento. Caso tenha saldo suficiente, o total a pagar é retirado do saldo corrente do cliente.

4.12 Aplicação de obtenção de senhas

Esta aplicação não dispõe de ambiente gráfico. A *Thread* de leitura do cartão encontra-se sempre a decorrer. Assim que um cartão for inserido no recetor NFC é pesquisado na base de dados online se portador do cartão comprou a senha do dia em questão e se ainda não a obteve. Caso ambos se verifiquem a senha é impressa e obtida pelo portador do cartão. Nesta aplicação apenas se pode obter a senha do dia em questão.

5 Conclusão

Foi feita a análise de requisitos das aplicações nas suas diferentes versões, para o servidor virtual, para interface dos utilizadores de smarphones android e para o funcionário do Instituto efetuar a gestão da aplicação. Foi simulado um servidor que poderá vir a ser utilizado no IPG se a aplicação for instalada pelo Instituto. A aplicação para smartphones encontra-se a funcionar e estará à disposição dos utilizadores interessados nos serviços que esta oferece, caso seja aceite pela instituição.

O serviço HCE é de implementação bastante rápida e fácil de entender. A transmissão de dados por NFC é bastante rápida, pelo que os métodos criados para pagamentos e de para obter senhas tornam-se também mais rápidos, eliminando o tempo de espera nas filas dos bares no IPG.

Tendo isto em conta, pode-se concluir que os objetivos previstos foram todos alcançados.

Bibliografia

Devmedia. (2016). *Modelo ER*. Obtido de Devmedia: http://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332

Fonseca, J.C. (2013). Introdução às base de dados.

IBM. (2016). *Diagrama de Componentes*. Obtido de IBM: http://www.ibm.com/support/knowledgecenter/pt-br/SS8PJ7_9.0.0/com.ibm.xtools.modeler.doc/topics/ccompd.html

IBM. (2016). *Diagrama de Navegação*. Obtido de IBM: http://www.ibm.com/support/knowledgecenter/pt-br/SSRTLW_9.1.0/com.ibm.xtools.topicbrowse.doc/topics/cbrowse.html

Netbeans. (2015). Obtido de Netbeans: https://netbeans.org/

Silveira M. (2013). Apontamentos fornecidos pela docente.

Soomro, T. R. (2013). *Impact of Smartphone's on Society.* European Journal of Scientific Research.

Studio, A. (2015). *Android Studio*. Obtido de Android Developer: https://developer.android.com/studio/index.html

Tecmundo. (2014). *Apple Pay*. Obtido de Tecmundo: https://www.tecmundo.com.br/apple-pay/63027-funciona-apple-pay-o-novo-sistema-pagamentos-iphone.htm

University Business . (2013). Obtido de http://www.universitybusiness.com/article/using-nfc-replace-campus-one-cards-smartphones

Wikipédia. (2016). *Diagrama de Casos de Uso*. Obtido de Wikipédia: https://pt.wikipedia.org/wiki/Diagrama_de_caso_de_uso

Wikipédia. (2016). *Diagrama de estados*. Obtido de Wikipédia: https://pt.wikipedia.org/wiki/Diagrama_de_transi%C3%A7%C3%A3o_de_estados

Wikipédia. (2014). *Near Field Communication*. Obtido de Wikipédia: http://pt.wikipedia.org/wiki/Near_Field_Communication

Wikipédia. (2015). *Programação Extrema*. Obtido de Wikipédia: https://pt.wikipedia.org/wiki/Programação_extrema

Wikipédia. (2015). *Samsung Galaxy S6*. Obtido de Wikipédia: https://en.wikipedia.org/wiki/Samsung_Galaxy_S6

Listagem de anexos

Servidor

```
public class IPGServer {
    Connection connAlunos = null;

    //metodo para ligar à base de dados online
    public void ligarBD(){

        try {
            Class.forName("com.mysql.jdbc.Driver");
            connAlunos =
DriverManager.getConnection("jdbc:mysql://localhost/alunos","root","");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE, null,
ex);
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE, null,
ex);
        }

    }
    //contrutor
    public IPGServer(){
        //inicia a ligação à base de dados
        ligarBD();

        try {
            //cria o servidor
            ServerSocket s = new ServerSocket(8081);
            while(true){
                //aceita a ligação e inicia a thread que trata dos pedidos da aplicação
para smartphones
                Socket conexao = s.accept();
                ThreadIniateConnection tr = new ThreadIniateConnection(conexao);
```

```
            tr.start();
            System.out.println("ligado")
          }
       } catch (IOException ex) {
          Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE, null,
ex);
       }
   }

   public class ThreadIniateConnection extends Thread{
       //variaveis a usar

       BufferedReader in;
       PrintStream out;
       public int idaluno=0;
       public
       String naluno;
       Socket sock;
       String smensagem;

       //contrutor da thread, cria o buffered reader para leitura e printStream para
enviar mensagem
       public ThreadIniateConnection(Socket Conection) throws IOException{
          in= new BufferedReader(new
InputStreamReader(Conection.getInputStream()));
          out= new PrintStream(Conection.getOutputStream());

          sock=Conection;
       }
       //valida o login envia pela aplicação
       public boolean validarLogin(String login,String password){
          boolean conf=false;
          String user="";
          String pass="";
```

```
            if(login!=""&&password!=""){
                try {
                    //pesquisa pela credenciais
                    //se se obter dados é porque esse utilziador existe
                    Statement stmt = connAlunos.createStatement();
                    ResultSet rs = stmt.executeQuery("SELECT n_aluno,password
FROM alunos where n_aluno="+login);
                    while(rs.next()){
                        user=rs.getString(1);
                        pass=rs.getString(2);
                    }
                    //confirma se as password está correta
                    if(password.equals(pass)&&user!=""){
                        conf=true;
                        naluno=login;
                    }
                    rs.close();
                } catch (SQLException ex) {
                    Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
                }
            }
            //faz return à confirmação da validação
            return conf;
        }
        //ontem o id do aluno, não o numero de aluno é mais facil o registo a
apartir do id do aluno
        public int getIdAluno(int naluno){
            int id=0;
            try {

                Statement stmt = connAlunos.createStatement();
                ResultSet rs = stmt.executeQuery("SELECT id_aluno FROM alunos
where n_aluno="+naluno);
```

```
            while(rs.next()){
                id=rs.getInt(1);
            }
            rs.close();
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
        }

        return id;
    }

    public String formatText(String texto,int tamanho){
        return texto.substring(0,tamanho);
    }

    //pesquisa na base de dados pelos movimentos do utilizador da aplicação
e insere-os numa string preparando assim o envio da mensagem
    public String getMovimentos(){
        String str="";
        int x=0;
        try {
            Statement stmt = connAlunos.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT
tipo_compra,forma_aquisicao,total,data from movimentos where
id_aluno="+idaluno+" order by data desc limit 10");
            while(rs.next()){
                str+=rs.getString(1)+"*";
                str+=rs.getString(2)+"*";
                str+=rs.getFloat(3)+"*";
                str+=rs.getDate(4)+"*";
                str+=rs.getTimestamp(4)+"*";
                x++;
            }
```

```
            } catch (SQLException ex) {
                Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
            }
            return x+"*"+str;
        }
        //obtem as senhas disponiveis para compra
        //e diferencia as que já foram obtidas pelo utilizador das que nao foram
        //devolve em string
        public String Senhasdisponiveis(){
            ArrayList<Integer> aint = new ArrayList<Integer>();
            String str = "";
            String devolve = "";
            float preco =0;
            float preco_dia=0;
            Calendar c = Calendar.getInstance();
            Time time =  new
Time(c.get(Calendar.HOUR_OF_DAY),c.get(Calendar.MINUTE),c.get(Calendar
.SECOND));
            Time hora_agora=time;
            Date data_agora = new
Date(c.get(Calendar.YEAR),c.get(Calendar.MONTH),c.get(Calendar.DAY_OF_
MONTH));
            Time hora_limite = time;
            try {

                Statement stmt = connAlunos.createStatement();
                ResultSet rs = stmt.executeQuery("SELECT
preco,preco_dia,hora_limite FROM preco_Senhas where activo=1");
                while(rs.next()){
                    preco=rs.getFloat(1);
                    preco_dia=rs.getFloat(2);
                    hora_limite = rs.getTime(3);
                }
```

```
            //senhas compradas pelos alunos
            rs= stmt.executeQuery("Select id_senha from senhas where id_aluno
= "+idaluno+" and id_senha in (SELECT id_senha from info_senhas WHERE
data>=NOW())");
            while(rs.next()){
               aint.add(rs.getInt(1));
            }

            rs= stmt.executeQuery("Select NOW() from dual");
            while(rs.next()){
               data_agora=rs.getDate(1);
               hora_agora=rs.getTime(1);
            }


            //info da senhas
            rs= stmt.executeQuery("SELECT id_senha, carne, peixe, sopa,
sobremesa, data FROM info_senhas WHERE data>=NOW()order by data
asc");
            int x = 0;
            while(rs.next()){


               if(data_agora.equals(rs.getDate(6))){
                  if(hora_agora.before(hora_limite)){
                     String conf = "N*";
                     for(int y=0;y<aint.size();y++){

                        if(aint.get(y)==rs.getInt(1)){
                           conf="S*";
                        }
                     }
                     str+=rs.getInt(1)+"*"+conf;
                     str+=rs.getString(2)+"*";
```

```
            str+=rs.getString(3)+"*";
            str+=rs.getString(4)+"*";
            str+=rs.getString(5)+"*";
            str+=String.valueOf(rs.getDate(6))+"*";
            str+=preco_dia+"*";
            int dia_semana = rs.getDate(6).getDay()+1;
            str+=dia_semana+"*";
            x++;
        }
    }else{


        String conf = "N*";
        for(int y=0;y<aint.size();y++){

            if(aint.get(y)==rs.getInt(1)){
                conf="S*";
            }
        }
        //aqui sera posto
        str+=rs.getInt(1)+"*"+conf;
        str+=rs.getString(2)+"*";
        str+=rs.getString(3)+"*";
        str+=rs.getString(4)+"*";
        str+=rs.getString(5)+"*";
        str+=String.valueOf(rs.getDate(6))+"*";
        str+=preco+"*";
        int dia_semana = rs.getDate(6).getDay()+1;
        str+=dia_semana+"*";
        x++;

    }

}
```

```
            devolve = String.valueOf(x)+"*"+str;
            rs.close();
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
        }

        return devolve;
    }

    //obtem o curso do utilizador
    public String getCurso(){
        String str = "";
        try {
            Statement stmt = connAlunos.createStatement();

            ResultSet rs = stmt.executeQuery("SELECT descricao FROM cursos
WHERE id_curso=(Select curso from alunos where id_aluno="+idaluno+")");
            while(rs.next()){
                str=rs.getString(1);
            }
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return str;
    }
    //obtem o nome do utilizador
    public String getNome(){
        String str = "";
        try {
            Statement stmt = connAlunos.createStatement();
```

```
            ResultSet rs = stmt.executeQuery("Select nome from alunos where
id_aluno="+idaluno);
            while(rs.next()){
                str=rs.getString(1);
            }
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return str;
    }

    public InputStream getBlob(String sql){
        InputStream blob = null;
        try {

            Statement stmt = connAlunos.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT foto FROM alunos where
id_aluno="+idaluno);
            while(rs.next()){
                blob=rs.getBlob(1).getBinaryStream();
            }
            rs.close();
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return blob;
    }

    //regista a senha do que o utilizador pediu caso nao a tenha obtido ainda e
caso o utilizador tenha saldo
    public String comprarSenha(int idsenha,String tiposenha){
        String str = "O seu pedido foi recusado!";
```

```
        try {
            Statement stmt = connAlunos.createStatement();
            ResultSet rs = stmt.executeQuery("Select saldo from alunos where
id_aluno="+idaluno);
            double saldo = 0;
            while(rs.next()){
                saldo=rs.getDouble(1);
            }
            rs = stmt.executeQuery("Select preco from preco_senhas where
activo=1");
            double preco=0;
            while(rs.next()){
                preco=rs.getDouble(1);
            }
            if(saldo>=preco){
                PreparedStatement pr =connAlunos.prepareStatement("INSERT
INTO `senhas` (`id_senha`, `id_aluno`, `data_compra`, `usado`, `tipo`)
values(?,?,NOW(),?,?)");
                pr.setInt(1, idsenha);
                pr.setInt(2, idaluno);
                pr.setBoolean(3, false);
                pr.setString(4, tiposenha);
                pr.execute();
                stmt.executeUpdate("update alunos set saldo="+(saldo-
preco)+"where id_aluno="+idaluno);
                str ="Compra efectuada!";
            }else{
                str="Saldo insuficiente!";
            }
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
            str="Ocorreu um erro ao comprar a senha!";
        }
```

```
            return str;
        }
        //envia o email para o utilizador com o código NFC
        public String SendEmail(){
            String email="";
            String NFC_Code="";
            String devolve="Erro ao enviar o email!";
            try {
                Statement stmt = connAlunos.createStatement();
                ResultSet rs = stmt.executeQuery("SELECT email,NFC_Code FROM
alunos where id_aluno="+idaluno);
                while(rs.next()){
                    email=rs.getString(1);
                    NFC_Code=rs.getString(2);
                }
            } catch (SQLException ex) {
                Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
            }
            final String username = "ipgtestemail1@gmail.com";
            final String password = "ipgteste";

            Properties props = new Properties();
            props.put("mail.smtp.starttls.enable", "true");
            props.put("mail.smtp.auth", "true");
            props.put("mail.smtp.host", "smtp.gmail.com");
            props.put("mail.smtp.port", "587");

            Session session = Session.getInstance(props,
              new javax.mail.Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(username, password);
                }
            });
```

```
            try {

                Message message = new MimeMessage(session);
                message.setFrom(new InternetAddress("ipgtestemail1@gmail.com"));

                message.setRecipients(Message.RecipientType.TO,
                    InternetAddress.parse(email));
                message.setSubject("Activação do NFC");
                message.setText("O código de activação é: "+NFC_Code);

                Transport.send(message);
                devolve="Email enviado com sucesso!";
            } catch (MessagingException e) {
                throw new RuntimeException(e);
            }
            return devolve;
        }
        //faz a leitura até o proximo * da mensgem recebida pela aplicação
        public String getCampo(){
            char c = smensagem.charAt(0);
            String str="";
            while(!String.valueOf(c).equals("*")){
                str+=c;
                smensagem=smensagem.substring(1,smensagem.length());
                c = smensagem.charAt(0);
            }
            smensagem=smensagem.substring(1, smensagem.length());
            return str;
        }
        //verifica se o codigo de ativação NFC enviado pelo utilizador da aplicação
está correto
        public boolean activarNFC(String codigo){
            String NFC_Code="";
```

```
            try {
                Statement stmt = connAlunos.createStatement();
                ResultSet rs = stmt.executeQuery("SELECT NFC_Code FROM
alunos where id_aluno="+idaluno);
                while(rs.next()){
                    NFC_Code=rs.getString(1);
                }
            } catch (SQLException ex) {
                Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
            }
            if(NFC_Code.equals(codigo)){
                return true;
            }else{
                return false;
            }
        }

        public String asciiToHex(String asciiValue){
            char[] chars = asciiValue.toCharArray();
            StringBuffer hex = new StringBuffer();
            for (int i = 0; i < chars.length; i++)
            {
                hex.append(Integer.toHexString((int) chars[i]));
            }
            return hex.toString();
        }
        //obtem os horarios da base de dados e devolve uma string com os
horarios
        public String getHorarios(){
            String str = "";
            int x=0;
            try {
                Statement stmt = connAlunos.createStatement();
```

```
            ResultSet rs = stmt.executeQuery("SELECT id_horario,descricao
FROM Horarios WHERE id_curso=(Select curso from alunos where
id_aluno="+idaluno+")");
            while(rs.next()){
                str+=rs.getInt(1)+"*";
                str+=rs.getString(2)+"*";
                x++;
            }
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return x+"*"+str;
    }
    //obtem as Avaliações da base de dados e devolve uma string com os
avaliações
    public String getAvaliacoes(){
        String str = "";
        int x=0;
        try {
            Statement stmt = connAlunos.createStatement();

            ResultSet rs = stmt.executeQuery("SELECT id_Avaliacao,descricao
FROM Avaliacoes WHERE id_curso=(Select curso from alunos where
id_aluno="+idaluno+")");
            while(rs.next()){
                str+=rs.getInt(1)+"*";
                str+=rs.getString(2)+"*";
                x++;
            }
        } catch (SQLException ex) {
            Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
```

```
            }
            return x+"*"+str;
        }

        @Override
        public void run(){


            try {
                //enquanto existir ligação executa o codigo
                while(sock.isConnected()){
                    //le a mensagem recebida
                    smensagem = in.readLine();
                    //converte para topico
                    String comando=getCampo();

                    switch(comando){
                        case "login":
                            String login=getCampo();
                            String pass=getCampo();
                            //validação das credenciais recebidas e reenvio do seu
estado de confirmação
                            if(idaluno==0){
                                if(validarLogin(login,pass)){
                                    //envia as informações do aluno
                                    out.println("login*aceite*");
                                    idaluno=getIdAluno(Integer.parseInt(login));
                                    out.println(login);
                                    out.println(getNome());
                                    out.println(getCurso());
                                    out.println(pass);
                                }else{
                                    out.println("login*recusado*");
                                }
```

```
                }


                break;
            case "movimentos":
                //envia os movimentos
                out.println("movimentos*"+getMovimentos());

                break;
            case "senhas":
                //envia senhas
                out.println("senhas*"+Senhasdisponiveis());
                break;

            case "compra":
                //relacionado com a compra de senhas
                //faz a compra se for possivel e envia o estado da compra
                int idsenha=Integer.parseInt(getCampo());
                String tiposenha=getCampo();
                out.println("compra*"+comprarSenha(idsenha, tiposenha));
            break;
            case "NFC":

                switch (getCampo()){
                    case "activar":
                        //se for relativo a ativar verifica o codigo e envia se
está correto ou nao
                        if(activarNFC(getCampo())){
                            out.println("NFC*activar*s*Código correcto*");
                            //out.println(asciiToHex(naluno));
                        }else{
                            out.println("NFC*activar*n*Código incorrecto*");
                        }
```

```
                    break;
                case "pedido":
                    //envia o email para o aluno e manda para a aplicação
que enviou o email
                    out.println("NFC*pedido*"+SendEmail()+"*");
                    break;
                }

                break;
            case "horarios":
                //envia a lista dos horarios
                out.println("horarios*"+getHorarios());
                break;
            case "avaliacoes":
                //envia a lista das avaliacoes
                out.println("avaliacoes*"+getAvaliacoes());
                break;
            case "logout":
                //fecha a ligação
                sock.close();
                break;
            /*case "download":

                String tipodownload=getCampo();
                String iddownload=getCampo();
                String descricao=getCampo();

out.println("download*"+tipodownload+"*"+iddownload+"*"+descricao+"*");
                break;*/
            }
        }
    } catch (IOException ex) {
        Logger.getLogger(IPGServer.class.getName()).log(Level.SEVERE,
null, ex);
```

```
            }
        }
    }
}
```

Async Task Android

```
public class serverConnection extends AsyncTask<Void, Void, Void> {
    //variavel usada para guardar as mensagens do servidor
    String smensagem = "";
    //variavel para obter as mensagens recebidas pelo servidor
    BufferedReader in = null;

    //metodo para obter os movimentos enviados pelo servidor
    public void getMovimentos() {
        //o primerio campo a ler da mensagem é o numero de movimentos recebidos
        int size = Integer.parseInt(getCampo());
        int x = 0;
        //enquanto nao se obter todos os movimentos executa o código
        while (x < size) {
            //cria um novo movimento
            ClassMovimentos mov = new ClassMovimentos();
            //Leitura dos campos recebidos, o nome das variaveis indica o que é lido
            mov.tipo_compra = getCampo();
            mov.forma_aquisicao = getCampo();
            mov.Total = getCampo();
            mov.Data = getCampo();
            mov.DataTime = getCampo();
            //guarda os movimentos num arrayList para depois ser usado nas atividades relativas aos movimentos
            ArrayListMovimentos.AddMovimento(mov);
            x++;
        }
    }
    //o mesmo principio do getMovimentos() mas relacionado aos horarios
    public void getHorarios(){
        int size = Integer.parseInt(getCampo());
```

```
    int x = 0;
    while (x < size) {
        ClassHorarios hor= new ClassHorarios();
        hor.id = Integer.parseInt(getCampo());
        hor.descricao = getCampo();

        ArrayListHorarios.addHorario(hor);
        x++;
    }
}

//o mesmo principio do getMovimentos()
public void getAvaliacoes(){
    int size = Integer.parseInt(getCampo());
    int x = 0;
    while (x < size) {
        ClassAvaliacoes ava= new ClassAvaliacoes();
        ava.id = Integer.parseInt(getCampo());
        ava.descricao = getCampo();

        ArrayListAvaliacoes.addAvaliacoes(ava);
        x++;
    }
}

//o mesmo principio do getMovimentos()
public void getSenhas() {

    int size = Integer.parseInt(getCampo());
    int x = 0;
    while (x < size) {
        ClassSenhas senha = new ClassSenhas();
        senha.id_senha = getCampo();
        senha.comprado = getCampo();
```

```
            senha.carne = getCampo();
            senha.peixe = getCampo();
            senha.sopa = getCampo();
            senha.sobremesa = getCampo();
            senha.data_usar = getCampo();
            senha.preco = getCampo();
            senha.dia_semana = getCampo();


            ArrayListSenhas.AddSenha(senha);
            x++;
        }

    }

    //metodo para obter os campos
    //le do primeiro carater até encontrar um * e apaga o que foi lido
    public String getCampo() {
        //obtem o primerio carater contido na mensagem
        char c = smensagem.charAt(0);
        String str = "";
        //enquanto nao se encontrar um * vai ler e inserir na string que se tem de
devolver e vai retirando da variavel que contem a mensagem
        while (!String.valueOf(c).equals("*")) {
            str += c;
            smensagem = smensagem.substring(1, smensagem.length());
            c = smensagem.charAt(0);
        }
        //refaz a mensagem retirando o primeiro carater que é um *
        smensagem = smensagem.substring(1, smensagem.length());

        return str;
    }
```

```
    //metodo para mostrar uma mensagem ao utilizador
    public void setToast(final String mensagem){
        ActivityMain.this.runOnUiThread(new Runnable() {

            public void run() {
                Toast.makeText(ActivityMain.this, mensagem,
Toast.LENGTH_SHORT).show();
            }
        });
    }



    @Override
    protected Void doInBackground(Void... params) {
        Intent intent;
        while (true) {

            try {
                //inicia a ligação por sockets ao servidor
                Socket sock = new Socket(ClassConnection.Server_IP,
ClassConnection.serverport);
                ClassConnection.Connect(sock);
                in = ClassConnection.getIn();

                try {

                    //caso já exista um utilizador inserido na base de dados faz login
automatico
                    if(db.userRegistered()){
                        ClassConnection.SendMessage("login*"+ClassUser.login+"*"+
ClassUser.password+"*");
                    }
                    while (sock.isConnected()) {
```

```
                    smensagem = in.readLine();
                    switch (getCampo()) {
                       //o comando recebido foi o movimento
                       case "movimentos":

                          //limpa o arrayList dos movimentos
                          ArrayListMovimentos.clearArray();
                          //obtem os movimentos recebidos
                          getMovimentos();
                          //inicia a ativade dos movimentos
                          intent = new Intent(getBaseContext(),
ActivityMovimentos.class);
                          intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                          getApplication().startActivity(intent);
                          break;
                       //o comando recebido foi das senhas
                       case "senhas":
                          //limpao arrayList das senhas
                          ArrayListSenhas.clearArray();
                          //obtem as senhas
                          getSenhas();
                          //inicia a atividade das senhas
                          intent = new Intent(getBaseContext(),
ActivitySenhas.class);
                          intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                          getApplication().startActivity(intent);
                          break;
                       //o comando recebido foi o login
                       case "login":
                          //se o login foi aceite
                          if (getCampo().equals("aceite")) {
                             //le as informações dos alunos
                             ClassUser.login=in.readLine();
                             ClassUser.nome=in.readLine();
```

```
                        ClassUser.curso=in.readLine();
                        ClassUser.password=in.readLine();

                        if(!ClassUser.logged){
                            intent = new Intent(getBaseContext(),
ActivityMenu.class);

intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                            getApplication().startActivity(intent);
                            ClassUser.logged=true;
                            finish();
                        }

                    } else {
                        //mostra mensagem de erro
                        setToast("Dados Incorrectos");
                    }
                    break;
                //o comando recebido é relativo às compras de senhas
                case "compra":

                    break;
                case "NFC":

                    break;
                case "avaliacoes":
                    //apaga os dados do arraylist avaliações
                    ArrayListAvaliacoes.clearArray();
                    //obtem as avaliações
                    getAvaliacoes();
                    //inicia a ativiade das avaliações
                    intent = new Intent(getBaseContext(),
ActivityAvaliacoes.class);
                    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

```
                        getApplication().startActivity(intent);
                        break;
                    case "horarios":
                        //o que o das avaliações
                        ArrayListHorarios.clearArray();
                        getHorarios();
                        intent = new Intent(getBaseContext(),
ActivityHorarios.class);
                        intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                        getApplication().startActivity(intent);
                        break;

                }
            }
        } catch (IOException e1) {
            e1.printStackTrace();

            ClassConnection.Disconnect();
        } catch (NullPointerException ex) {
            ClassConnection.Disconnect();
        }
    } catch (IOException e) {
        e.printStackTrace();
        ClassConnection.Disconnect();
    } catch (StringIndexOutOfBoundsException ex) {

    }
    //a exepções desligam impedem que as mensagens das ativades para o
servidor sejam enviadas até nova ligação
  }
 }
}
```

Classe User Android

```
public class ClassUser {
    //informações do utilizador
    //login é o numero do aluno
    public static String login="";
    public static String password="";
    public static String nome="";
    public static String curso="";
    public static boolean NFC=false;
    public static boolean logged=false;
    public ClassUser(){

    }
}
```

6 Classe Conection Android

```
public class ClassConnection {
    //dados da ligação por sockets
    public static Socket sock;
    public static String Server_IP="192.168.1.3";

    public static int serverport = 8081;
    public static int downloadport= 8082;
    //variavel que permite enviar dados ao servidor
    public static PrintStream out;
    public static BufferedReader in = null;
    //variavel que decide se uma mensagem pode ou não ser enviada para o servidor
    //fica a true quando a AsyncTask liga-se ao servidor e a false quando há uma exeção na AsyncTask exemplo: perda de ligação
```

```
    public static boolean connected=false;

    //metodo para obter o reader, usado na asynctask
    public static synchronized BufferedReader getIn(){
        return in;
    }
    //mete a variavel connected a true e cria o bufferedReader e o printStream
para enviar e receber mensagens do servidor
    public static synchronized void Connect(Socket socket){
        sock = socket;
        try {
            out = new PrintStream(sock.getOutputStream());
            in = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
            DataInputStream dIn  = new DataInputStream(sock.getInputStream());
            connected=true;
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    //mete a variavel connected a false impedindo a saida de mensagens
    public static synchronized void Disconnect(){
        connected=false;
    }
    //envia uma mensagem para o servidor, usado nas atividades e apenas
envia a mensagem se a variavel connected estiver a true
    public static synchronized boolean SendMessage(String Message){
        if(connected){
            out.println(Message);
            return true;
        }else{
            return false;
        }
    }
```

Classe Movimentos Android

```
public class ClassMovimentos {
    //informaçoes dos movimentos
    public String tipo_compra;
    public String forma_aquisicao;
    public String Total;
    public String Data;
    public String DataTime;
    public ClassMovimentos(String tipo_compra, String forma_aquisicao, String
Total, String Data){
        this.tipo_compra=tipo_compra;
        this.forma_aquisicao=forma_aquisicao;
        this.Total=Total;
        this.Data=Data;
    }
    public ClassMovimentos(){

    }
}
```

7 Classe Senhas Android

```
public class ClassSenhas {
    //informações das senhas
    public String id_senha;
    public String dia_semana;
    public String data_usar;
    public String comprado;
    public String sopa;
    public String peixe;
    public String carne;
```

```
    public String sobremesa;
    public String preco;

    public ClassSenhas(String dia_semana, String data_usar, String
Comprado){
        this.dia_semana=dia_semana;
        this.data_usar=data_usar;
        this.comprado=comprado;
    }
    public ClassSenhas(){

    }
    //contrutor da classe
    public ClassSenhas(String id_senha, String dia_semana, String data_usar,
String sopa, String peixe, String carne, String sobremesa, String preco, String
Comprado){
        this.id_senha=id_senha;
        this.dia_semana=dia_semana;
        this.data_usar=data_usar;
        this.sopa=sopa;
        this.carne=carne;
        this.sobremesa=sobremesa;
        this.preco=preco;
        this.peixe=peixe;
        this.comprado=Comprado;
    }
}
```

Classe Horários Android

```
public class ClassSenhas {
   //informações das senhas
   public String id_senha;
   public String dia_semana;
   public String data_usar;
   public String comprado;
   public String sopa;
   public String peixe;
   public String carne;
   public String sobremesa;
   public String preco;

   public ClassSenhas(String dia_semana, String data_usar, String
Comprado){
      this.dia_semana=dia_semana;
      this.data_usar=data_usar;
      this.comprado=comprado;
   }
   public ClassSenhas(){

   }
   //contrutor da classe
   public ClassSenhas(String id_senha, String dia_semana, String data_usar,
String sopa, String peixe, String carne, String sobremesa, String preco, String
Comprado){
      this.id_senha=id_senha;
      this.dia_semana=dia_semana;
      this.data_usar=data_usar;
      this.sopa=sopa;
      this.carne=carne;
      this.sobremesa=sobremesa;
      this.preco=preco;
```

```
        this.peixe=peixe;
        this.comprado=Comprado;
    }
}
```

Classe Avaliações Android

```
public class ClassAvaliacoes {
    //informações das avaliações
    //id da base de dados online
    public int id;
    public String descricao;

    public ClassAvaliacoes(int id, String descricao){
        this.id=id;
        this.descricao=descricao;
    }
    public ClassAvaliacoes(){

    }
}
```

Serviço HCE Android

```
public class HCEmulation extends HostApduService{
   //AID da aplicação não é usada mas está aqui para eu saber qual é o AID em hexadecimal
   private static final byte[] AID_SELECT_APDU = {
         (byte) 0x00, // CLA (class of command)
         (byte) 0xA4, // INS (instruction); A4 = select
         (byte) 0x04, // P1  (parameter 1)  (0x04: select by name)
         (byte) 0x00, // P2  (parameter 2)
         (byte) 0x07, // LC  (length of data)
         (byte) 0xF0, (byte) 0x39, (byte) 0x41, (byte) 0x48, (byte) 0x14, (byte) 0x81, (byte) 0x00,
         (byte) 0x00 // LE   (max length of expected result, 0 implies 256)
   };


   @Override
   public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {

      //inicia a ligação à base de dados
      DBHelper db = new DBHelper(this);
      db.getReadableDatabase();
      //o metodo da base de dados é usado
      //a descrição do método encontra no DBHelper
      //envia para o recetor a resposta
      //como os dados a enviar é apenas o n_aluno podemos dar return instantaneamente assim
      return db.responseAPDU();
   }

   @Override
```

```
    public void onDeactivated(int reason) {

    }
}
```

Thread de Recetor

```
public class ThreadCartao extends Thread{
    static private String id;
    /*static public float valor;
    String local = "Bar ESTG";
    PLU plu;
    Connection connAlunos;
    Pagamento pag;*/
    Cartao cartao;
    //AID da aplicação para smartphones
    private static final byte[] AID_SELECT_APDU = {
                        (byte) 0x00, // CLA (class of command)
                        (byte) 0xA4, // INS (instruction); A4 = select
                        (byte) 0x04, // P1  (parameter 1)  (0x04: select by name)
                        (byte) 0x00, // P2  (parameter 2)
                        (byte) 0x07, // LC  (length of data)
                        (byte) 0xF0, (byte) 0x39, (byte) 0x41, (byte) 0x48, (byte)
0x14, (byte) 0x81, (byte) 0x00,
                        (byte) 0x00 // LE   (max length of expected result, 0 implies
256)
          };



    public ThreadCartao(Cartao cartao){
        this.cartao=cartao;
```

```
    }

    @Override
    public void run(){
            //ligarBD();
        while(!cartao.presente){


            try {
                //obter o recetor
                TerminalFactory factory = TerminalFactory.getDefault();
                List<CardTerminal> terminals = factory.terminals().list();
                System.out.println("Terminals: " + terminals);

                // Uar o primeiro recetor
                CardTerminal terminal = terminals.get(0);



                // ligação para o cartão
                terminal.waitForCardPresent(30000);
                Card card = terminal.connect("*");

                System.out.println("card: " + card);
                CardChannel channel = card.getBasicChannel();
                try{
                    //bytes para autentificação de chaves
                    byte[] autenticate = {(byte)0xFF, (byte)0x82, (byte)0x00,
(byte)0x00, (byte)0x06, (byte)0xFF, (byte)0xFF, (byte)0xFF, (byte)0xFF,
(byte)0xFF, (byte)0xFF};
                    //bytes para carregar chaves tipo A
                    byte[] loadA = {(byte)0xFF, (byte)0x86, (byte)0x00, (byte)0x00,
(byte)0x05, (byte)0x01, (byte)0x00, (byte)0x00, (byte)0x60, (byte)0x00};
                    //bytes para ler o bloco 01 do smartcard
```

```
            byte[] read = {(byte)0xFF, (byte)0xB0, (byte)0x00, (byte)0x01,
(byte)0x10};
            //autentificação
            ResponseAPDU resposta = channel.transmit(new
CommandAPDU(autenticate));
            //carregar chaves tipo A
            resposta = channel.transmit(new CommandAPDU(loadA));
            //ler o bloco 01 que contem o numero de aluno
            resposta = channel.transmit(new CommandAPDU(read));

            try {
               //passar o que é recebido para hexadecimal
               String aux = new
String(resposta.getBytes(),"ASCII").substring(0,16);
               id="";
               char[] idarray = aux.toCharArray();
               for(int x=0;x<idarray.length;x++){
                  try{
                     char c = idarray[x];
                     //apenas lê os carateres que podem ser inteiros
                     int integer=Integer.parseInt(String.valueOf(c));
                     id+=integer;
                  }catch(NumberFormatException e){

Logger.getLogger(Pagamento.class.getName()).log(Level.SEVERE, null, e);
                  }
               }

            } catch (UnsupportedEncodingException ex) {

Logger.getLogger(Pagamento.class.getName()).log(Level.SEVERE, null, ex);
            }

            //cria uma nova classe cartão
```

```
                cartao = new Cartao(id,"Cartão");
            }catch(IllegalArgumentException ex){
                //quando em vez de estar um smartcard se encontra um
smarphone esta excepção é lançada
                //assim inicio aqui o codigo do recetor para smartphones
                //espera pela ligação outra vez
                long wait=30000;
                terminal.waitForCardPresent(wait);
                Card phone = terminal.connect("T=1");
                System.out.println("card: " + phone);
                CardChannel channelphone = phone.getBasicChannel();
                //envia o AID para iniciar o HCE da aplicação
                ResponseAPDU resposta = channelphone.transmit(new
CommandAPDU(AID_SELECT_APDU));
                //converter hexadecimal para String a mesma coisa que antes
                try {
                    String aux = new
String(resposta.getBytes(),"ASCII").substring(0,16);
                    id="";
                    char[] idarray = aux.toCharArray();
                    for(int x=0;x<idarray.length;x++){
                        try{
                            char c = idarray[x];
                            int integer=Integer.parseInt(String.valueOf(c));
                            id+=integer;
                        }catch(NumberFormatException e){

                        }
                    }

                } catch (UnsupportedEncodingException as) {

Logger.getLogger(Pagamento.class.getName()).log(Level.SEVERE, null, ex);
                }catch(IllegalArgumentException eax){
```

```
                }

                //cria uma nova classe cartão
                cartao = new Cartao(id,"Smartphone");
            }



        } catch (CardException ex) {
            Logger.getLogger(Pagamento.class.getName()).log(Level.SEVERE,
null, ex);
        }

    }

    }
}
```

Classe Cartão

```
public class Cartao {
    //classe criada para guardar os dados do aluno na aplicação de compras e
contem as informações das vendas
    static public String id;
    static public String nome;
    static public String forma;
    static public String NIF;
    public String local= "Bar ESTG";
    static public boolean presente=false;

    static Pagamento pag;
    //contem as informações dos produtos a vender
    static PLU plu;
    Connection connAlunos;

    public void ligarBD(){
        //ligação à base de dados online
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connAlunos =
DriverManager.getConnection("jdbc:mysql://localhost/alunos","root","");
        } catch (ClassNotFoundException ex) {

        } catch (SQLException ex) {

        }

    }
    //construtor para introduzir as informações do aluno
    public Cartao(String id,String forma){

        try {
```

```
            ligarBD();
            Statement stmt;
            stmt = connAlunos.createStatement();
            ResultSet rs = stmt.executeQuery("Select n_aluno,nome,NIF from
alunos where n_aluno="+id);

            int x=0;
            while(rs.next()){
                this.id=id;
                this.forma=forma;
                this.nome=rs.getString(2);
                this.NIF=rs.getString(3);
                x++;
            }
            if(x>0){
                presente=true;
                pag.MostrarAluno(nome, NIF, id);
            }
        } catch (SQLException ex) {
            Logger.getLogger(Cartao.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
    //contrutor primario
    public Cartao(PLU plu, Pagamento pag){
        this.plu=plu;
        this.pag=pag;
        ligarBD();
    }
    //metodo para fazer o pagamento
    public boolean Pagar(String forma, float valor){

        try {
            Statement stmt;
```

```
            ligarBD();
            stmt = connAlunos.createStatement();
            //pesquisa na base de dados
            ResultSet rs = stmt.executeQuery("Select saldo,id_aluno from alunos
where n_aluno="+id);
            double saldo = 0;
            int id_aluno=0;
            while(rs.next()){
                saldo=rs.getDouble(1);
                id_aluno=rs.getInt(2);
            }
            //se o utilziador tiver saldo o seguinte codigo
            //regista o movimento e retira o saldo ao utilzador/aluno
            if(saldo>=valor){
                stmt.executeUpdate("update alunos set saldo="+(saldo-valor)+"where
id_aluno="+id_aluno);

                PreparedStatement stmtRegisto = connAlunos.prepareStatement((
                        "Insert into
movimentos(id_aluno,tipo_compra,forma_aquisicao,total,data)
Values(?,?,?,?,NOW())"));
                stmtRegisto.setInt(1,id_aluno);
                stmtRegisto.setString(2,local);
                stmtRegisto.setString(3,forma);
                stmtRegisto.setFloat(4,Float.parseFloat(String.valueOf(valor)));
                stmtRegisto.executeUpdate();
                return true;
            }else{
                return false;
            }


        } catch (SQLException ex) {
```

```
            Logger.getLogger(ThreadCartao.class.getName()).log(Level.SEVERE,
null, ex);
        }
        return false;
    }
}
```