Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Tiago Manuel Cardoso Sá Ferreira Simão

dezembro | 2012

PratosLimpos.NET

Tiago Simão

Responsive Web Design Aplicado à Restauração

Licenciatura – Engenharia Informática
Escola Superior de Tecnologia e Gestão
Instituto Politécnico da Guarda

Dezembro 2012 – Versão 2.0

Tiago Simão: PratosLimpos.NET, *Responsive Web Design* Aplicado à Restauração, © Dezembro 2012

Professor Orientador: Rui Pereira

Supervisores:
Professor Paulo Nunes
Professor Noel

“We have seen that computer programming is an art, because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty.” (Knuth, 1968)

RECONHECIMENTOS

O tempo dedicado a este projeto levou a que ocupasse grande parte do meu dia-a-dia nestes últimos dois meses e só consegui levá-lo avante com todo o apoio e força da minha família e amigos.

Um especial agradecimento à minha namorada não só pelo apoio, paciência e carinho que me deu, mas também por ter sido a pessoa que ajudou psicologicamente em cada etapa deste projeto.

RESUMO

Este projeto tem como objetivo desenvolver uma aplicação *Web* compatível com qualquer dispositivo móvel ou fixo. O princípio será fornecer a um espaço de restauração, a mobilidade e interação entre funcionários e clientes. O cliente ao aceder à *webpage* poderá efetuar o pedido que deseja, bem como, obter todas as informações sobre o espaço de restauração. Em relação ao funcionário, este poderá obter todos os pedidos através da mesma aplicação e geri-los consoante a sua função.

O público-alvo é o utilizador com qualquer dispositivo móvel com as tecnologias *WiFi* e *Browser*.

Independentemente do meio, a finalidade será reduzir o tempo de espera dos clientes para efetuar um pedido e fornecer todas as funções numa só aplicação, amiga do utilizador e fácil de utilizar.

Palavras-Chave:

Aplicação web, dispositivo móvel, tecnologia Wifi e Browser, amiga do utilizador.

ABSTRACT

This project aims to understand how responsive web design works on multiple personal devices. In this case I apply to the catering industry, in my opinion will increase the mobility between the employee and the client. When access to the Web Page, the client can order menus and see all the information about the restaurant. In the employee point of view, he can manage all orders in the same application.

The target is going to be all the users with a Wifi and Browser device. However, the purpose is to avoid the waiting period for costumers to make request and give all the options in one application only, user friendly and easy to use.

Key Words:

Responsive Web Design, Wifi and Browser device, catering industry, Web Application, Responsive Layouts, *user-friendly.*

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO

ENG *English*

API *Application Programming Interface*

UML *Unified Modeling Language*

XML *Extensible Markup Language*

RWD *Responsive Web Design*

ASP *Active Server Pages*

SVG *Scalable Vector Graphics*

SQL *Structured Query Language*

W3C *World Wide Web Consortium*

CSS *Cascading Style Sheets*

PT Português

APP Aplicação

BD Base de Dados

DFD Diagrama de Fluxos de Dados

1 INTRODUÇÃO

Por vezes encontramos alguns problemas quando decidimos ir a um restaurante, onde o mais ''stressante'' acaba por ser o tempo de espera para efetuar um pedido. Hoje em dia, com as novas tecnologias poderíamos contornar esses problemas se existisse um sistema informático que fornecesse aos clientes a possibilidade de efetuar o pedido através dos seus dispositivos móveis ou fixos. Com esta solução os clientes poderiam efetuar o seu pedido de imediato e poupariam trabalho ao serviço do restaurante.

Neste projeto será desenvolvida uma aplicação *Web* que visa contornar este problema, onde o princípio será fornecer a um espaço de restauração, a mobilidade e interação entre funcionários e clientes. O cliente ao aceder à *Web Page* poderá efetuar o pedido que deseja, bem como, obter todas as informações sobre o espaço de restauração. Em relação ao funcionário, este poderá obter todos os pedidos através da mesma aplicação e geri-los consoante a sua função.

O público-alvo serão todos os utilizadores com qualquer dispositivo com a tecnologia *WiFi* e *Browser*. Independentemente do meio, a finalidade será evitar o tempo de espera dos clientes para efetuar um pedido e ao longo deste trabalho irei demonstrar uma possível solução para este problema.

''The advance of technology is based on making it fit in so that you don't really even notice it, so it's part of everyday life.''
(Gates, 2001)

Nota Importante: Todo este trabalho resultou de muita pesquisa prévia e de uma matéria ainda não muito clara e concreta, por isso é possível a existência de algumas lacunas e até desatualizações dependentes da altura em que está a ser lido este relatório. Será abordada uma possível solução e de minha autoria, claro que não ficará autêntica e completa, mas deixo em aberto todo este projeto para futuros estudos.

1.1. MOTIVAÇÃO

A principal motivação para a criação deste projeto é, poder criar uma possível solução para o atendimento demorado na restauração e ao mesmo tempo "amiga da tecnologia". Esta aplicação poderá também facilitar a interação entre o funcionário e os clientes, ficando os últimos menos tempo à espera de serem atendidos.

A solução chave é o novo conceito *Responsive Web Design*, no qual ainda não está totalmente esclarecido. Para mim é gratificante participar nesta recente aposta para a *web.*

1.2. SOLUÇÃO

A solução proposta neste projeto, denominada de *Pratos Limpos*, consiste em desenvolver uma aplicação *web* com a capacidade de reunir um conjunto de recursos necessários para uma experiência dinâmica entre o utilizador e a aplicação, independentemente do dispositivo utilizado. Para pôr em prática este processo irão ser abordados, o desenvolvimento, a implementação e o teste da aplicação proposta.

O que pretendo com esta experiência dinâmica é que esta nova tecnologia, a que muitos dispositivos hoje têm acesso, permita que o utilizador tenha mais independência no dia-a-dia. Sendo o mais popular, o telemóvel, não irá desprezar todos os outros dispositivos, pois iria pôr em risco a credibilidade deste projeto, então a melhor vertente para ser amiga do utilizador (*User Friendly),* que encontrei, foi uma aplicação via *web* que se adapte ao meio onde esta está a ser executada.

Para além do *design*, a aplicação consiste em apresentar várias opções ao utilizador, de um ponto de vista geral, irá estar disponível a este, qualquer informação correspondente ao estabelecimento e seus propósitos. Mas sendo esta uma aplicação que visa a dar acesso pormenorizado a utilizadores somente registados, todo o conteúdo inicialmente apresentado será somente informativo e só após o cliente estar registado poderá ter acesso total a todos os outros conteúdos.

Conteúdos disponíveis aos clientes:

- Ver e personalizar menus do restaurante;
- Efetuar pedidos;
- Ver histórico enquanto utilizador;
- Gerir conta de utilizador;
- Efetuar pagamentos;
- Ver informação detalhada do estabelecimento e conteúdos disponibilizados via *on-line*;
- Comentar/sugerir o serviço prestado.

A aplicação também permite, pesquisar os conteúdos, informação sobre os menus da semana, bem como a disponibilidade do restaurante para receber o cliente.

Por fim é importante frisar que qualquer pedido pretendido pelo cliente só poderá ser efetuado no espaço do estabelecimento, uma explicação pormenorizada será dada mais adiante neste relatório.

1.3. CONTRIBUIÇÃO

A contribuição deste trabalho ainda é fictícia pois é um projeto independente de qualquer entidade, sendo exclusivamente preparado para a avaliação final da cadeira de Projeto do curso de Licenciatura de Engenharia Informática, mas que futuramente poderá ser aplicada a qualquer tipo de ambiente, seja ele restauração como outro meio qualquer onde o propósito seja a venda direta ao cliente. Mesmo assim será abordado de forma concisa e elucidativa para possível comercialização.

1.4. ESTRUTURA DO DOCUMENTO

Este relatório está estruturado com um total de 9 capítulos. Para além deste onde foi dado a conhecer a questão e o porque de realizar este projeto, de seguida irei falar da definição do problema e possíveis soluções. No terceiro, quarto, quinto e sexto capítulo será abordada a metodologia, diagramas UML, o estado da arte e a implementação, respetivamente. Por último serão tiradas as conclusões deste projeto e as previsões em relação a um possível futuro para esta aplicação.

2 DEFINIÇÃO DO PROBLEMA E POSSÍVEIS SOLUÇÕES

Neste capítulo irei dar uma breve descrição dos problemas e respetivos objetivos.

2.1. DEFINIÇÃO DO PROBLEMA

Ao começar com esta ideia, no início, não esperava deparar-me com tantos obstáculos pois de uma maneira generalizada parece ser muito mais fácil do que parece e quando pensamos que o mais simples é o mais fácil de implementar, muitas vezes estamos equivocados e por vezes acaba por ser o mais complicado e difícil de ultrapassar.

Este projeto tem como objetivo satisfazer todos os requisitos dos clientes de um restaurante, mas até o conseguir terei de abordar muitos problemas técnicos envolvidos nesta matéria.

2.1.1. MULTI-DEVICE WEB DESIGN

Uma das matérias chave para que este projeto tenha sucesso é e sempre será a independência do utilizador ao aceder à aplicação e que o consiga de uma maneira fácil e simples, sem grandes conceitos complexos e exigentes de algum conhecimento de vigor técnico por parte do próprio, pois essa é uma vertente que interessa a nós, programadores. Para isso e com o aparecimento dos *smartphones*, as soluções para o nosso dia-a-dia melhoraram bastante, mas como a tecnologia evolui sempre de forma contínua e pouco integral, muitos são os requisitos chave que ficam para trás e acabam por ter de se adaptar ao futuro.

O que quero dizer é que hoje em dia o conhecimento que tínhamos em relação ao conceito *WEB*, até mesmo para nós programadores ou até *designers*, mudou completamente com o surgimento destes dispositivos de pequenas dimensões e de grande aderência por parte do público. Sempre que um programador se preparava para implementar uma *web page,* normalmente recorria a um *canvas* ou ecrã para desenhar e colocar todos os conteúdos de forma estática e fixa, pois era esse o conceito que existia para um Mundo onde quem se dirigia para entrar no meio *web,* usava monitores bastante grandes comparados aos dos telemóveis de hoje em dia.

A questão do tamanho onde se iria ver os conteúdos nunca se punha em causa, pois com algum *zoom* por parte do *browser* tudo se veria de forma clara. Mas isto era o ponto de vista na era das 15-20 polegadas e hoje em dia para incluir uma página *web* num ecrã de 3 a 9 polegadas teremos de pensar de maneira diferente.

Um grande obstáculo neste novo conceito é o facto de as imagens nos formatos mais comuns (PNG, JPEG, GIF, BMP) ainda não poderem ser usadas de forma clara, pois outro grande problema é a largura de banda usada pelos telemóveis e utilizar grandes formatos de imagem para desenhar uma página *web* leva a que o acesso à Internet seja consistente, o que neste caso é o oposto.

Este tema é muito interessante e leva a muitas discussões por parte dos programadores, pois nunca se sabe que dispositivo é que o utilizador tem em mãos.

Figura 1: ***Responsive Layout's***

2.1.2. RESTRITO AO UTILIZADOR

Quando se fala em páginas *web* que têm como finalidade a venda de produtos ao cliente, o primeiro problema que nos ocorre é o tipo de controlo exercido pela aplicação. O objetivo da aplicação é que esta possa ser acedida por qualquer pessoa que possua o dispositivo necessário e pretenda efetuar uma compra, com tudo existem restrições e uma delas é, a restrição unicamente aos utilizadores registados na aplicação.

2.1.3. RESTRITO AO ESPAÇO

Uma outra restrição mais específica deste projeto é o facto de que certas opções da aplicação *web* só poderão ser acedidas no local onde é efetuado o pedido, pois não será possível efetuar pedidos fora do estabelecimento e para isso será restrito á ligação que o utilizador usa para aceder à Internet.

2.2. POSSÍVEIS SOLUÇÕES

Irei falar de algumas soluções que encontrei para dar a volta a alguns obstáculos, mas é como digo são possíveis soluções e não são definitivas, pois algumas delas ainda se encontram em estudo por parte da organização *W3C,* entre outras.

2.2.1. MULTI-DEVICE WEB DESIGN

Existem várias opções para implementar um *website* flexível e que adira a qualquer dispositivo. Pode ser uma aplicação que permita ao utilizador escolher se pretende o serviço móvel ou o serviço original para a *web,* ou uma aplicação que se adapte automaticamente ao tamanho do ecrã sem pedidos de permissão e sem serviços distintos.

A solução aqui implementada é aquela que se adapta sem o utilizador mudar de aplicação, hoje em dia conhecida como ***RESPONSIVE WEB DESIGN***. Pode ter sido recebida de forma duvidosa e cautelosa mas muitos apontam como sendo a única solução para responder a este avanço repentino na tecnologia móvel. Qualquer pessoa hoje pode aceder a uma página *web* via telemóvel e facilmente deparar-se com uma página estática, sem qualquer opção para redimensionar a vista e de difícil leitura. O utilizador será obrigado a fazer *zoom* para conseguir ler, ou escrever numa caixa de texto e efetuar *scroll* horizontal para descobrir todos os conteúdos da página. Estes são um dos grandes problemas nos *sites* antigos, mas o *Responsive Web Design* visa a contornar estes obstáculos sendo amigo do utilizador, *User Friendly*, *Mobile First* ou, na minha opinião, o termo *Experience First*, pois podemos muito bem estar a falar de telemóveis e/ou *tablet's* mas não nos podemos esquecer dos computadores pessoais que também são muito requisitados.

Então a solução encontrada, é disponibilizar ao utilizador uma só aplicação *web* mas que esta seja adaptável ao meio onde está a ser executada. Um *layout* que facilmente se adapte a um ecrã de grandes dimensões e recursivamente se vai adaptando aos outros diferentes tamanhos. Vendo bem esta é mais dependente do tamanho da janela do *browser* onde está a ser executado, pois é ele que indica qual o tamanho do ecrã do dispositivo e ninguém nos garante que o utilizador redimensione a sua janela num ecrã de grandes dimensões.

Tudo isto só seria possível através das *CSS3* que têm várias características para a *web*, onde neste caso as mais utilizadas são as *@media queries* que permitem facilmente manipular os conteúdos da página fase ao tamanho do dispositivo.

2.2.2. RESTRITO AOS UTILIZADORES

Em relação às restrições de tipos de utilizador, irá depender da implementação e das ferramentas que achei mais adequadas para este projeto. Sendo o *ASP.NET* a plataforma de eleição para desenhar grande parte da aplicação, também é um ponto forte no que toca às restrições pois permite facilmente gerir quem pode ou não aceder a certos conteúdos ou páginas *web*.

2.2.3. RESTRITO AO ESPAÇO

A aplicação *Pratos Limpos* tem como principal característica ter uma versão publicada *on-line* e outra no servidor privado do estabelecimento de restauração. O que acontece é que o utilizador poderá aceder à aplicação em qualquer ponto do mundo, mas só para obter informações do estabelecimento e gerir a sua conta de cliente. A outra versão da aplicação só é possível aceder através do serviço de internet prestado pelo estabelecimento (intranet), ou seja, qualquer pedido só poderá ser efetuado dentro do espaço de restauração.

Colocando a responsabilidade ao próprio estabelecimento em fornecer um serviço de partilha de Internet para os seus clientes, p.ex., *WI-FI*.

3 METODOLOGIA

Ao falar no desenvolvimento de *software* é importante seguir um rumo estrutural para que o estudo de todo o projeto prossiga de forma clara e com o menor número de obstáculos possível, caso ocorram, prever atempadamente para que o fator tempo não aumente substancialmente. As metodologias vêm facilitar a vida dos engenheiros de *software*, pois é composta por procedimentos, técnicas, ferramentas e documentação que ajudam a implementar um novo sistema.

Como o objetivo começa por encontrar e estudar os possíveis problemas do projeto, a primeira etapa passa por escolher uma boa metodologia para usar ao longo de todo este processo. É precisamente aqui que se encontra o primeiro obstáculo, pois existem muitas metodologias que por vezes são confusas e contraditórias, semelhantes mas muito diferentes ao mesmo tempo. O que nos leva a pensar no propósito do *software* e que tipo de cliente tem em mão.

3.1. METODOLOGIA

Neste caso, o meu projeto não tem qualquer compromisso a não ser com a data de entrega para a avaliação da disciplina de projeto, mas mesmo assim vou tentar expôr todos os problemas de forma sucinta para que o protótipo esteja em funcionamento da forma mais eficaz e no menor tempo possível.

O tipo de metodologia escolhida neste projeto é a metodologia ágil.

3.1.1. METODOLOGIA ÁGIL

Suponhamos que o cliente é uma cadeia de restaurantes que pretende um *software* o mais rápido possível e funcional, para facilitar e diminuir o tempo de espera dos seus clientes. Para este sistema ser executado, é necessário existir uma boa comunicação/interação com a entidade e a pessoa que está a desenvolver o *software*, o que requer trabalho mútuo e motivado por parte das duas entidades, pois o que está em causa é um produto que seja o mais funcional possível no menor espaço de tempo e suscetível a constantes alterações. Sem esta comunicação e confiança, o projeto caia em falhanço.

A metodologia mais adequada é a ágil pois para além de valorizar a entrega de um produto funcional e adequado ao que o cliente realmente deseja, a maior preocupação centrasse no desenvolvimento do *software*. Sendo esta equipa de um só elemento, é de certa forma mais fácil gerir toda a implementação testando regularmente cada etapa, p.ex: implementa-testa, implementa-testa e assim sucessivamente. Toda a documentação será baseada nas ferramentas usadas na produção, ou seja, no fim do desenvolvimento do *software*.

Como tal os conceitos chave são muito importantes:

- Indivíduos e interações em vez de processos e ferramentas;
- *Software* que funciona em vez de documentação abrangente;
- Colaboração do Cliente em vez de negociação de contratos;
- Resposta a modificações em vez de seguir um plano.

(Steffen, 2008)

3.2. DESCRIÇÃO DAS TAREFAS

As principais tarefas ou objetivos previstos são:

- 1. Análise de sistema dos requisitos;
- 2. Estudo dos algoritmos e cálculos a utilizar na aplicação Pratos Limpos;
- 3. Implementação da aplicação baseada nos pontos 1 e 2;
- 4. Fase de testes e análise da eficiência e fiabilidade da aplicação;
- 5. Documentação do projeto no relatório.

3.3. AGENDAMENTO DE TAREFAS

O mapa de Gantt apresentado (ver: **Figura 2**) não é preciso, mas está aproximado do tempo que foi aplicado a este projeto. Como é possível ver, o **Grupo 2** ainda permanece incompleto, pois todo o protótipo ainda se encontra em desenvolvimento devido ao pouco tempo que tive disponível para o realizar.

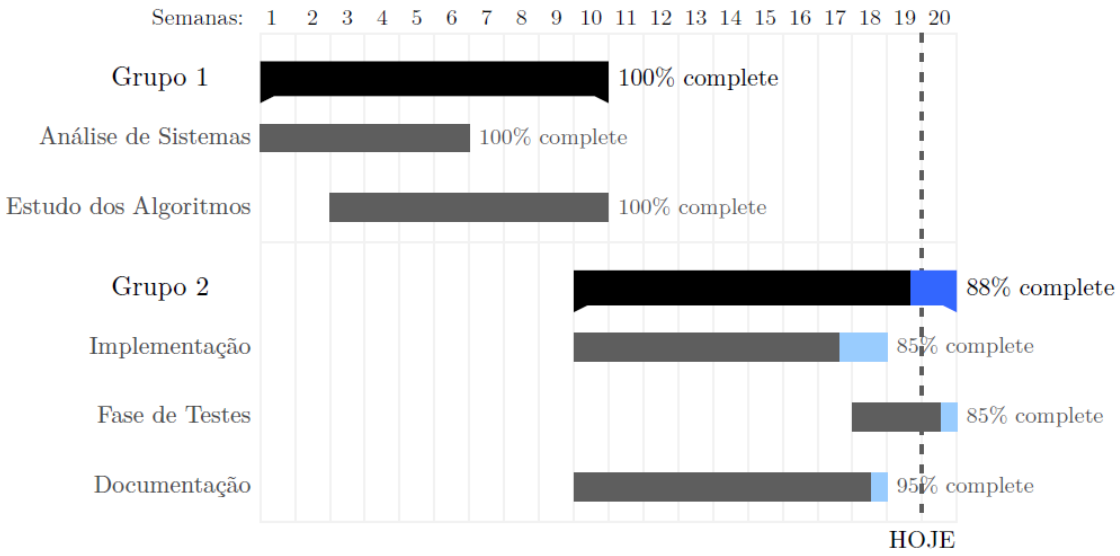


Figura 2: Mapa de Gantt

3.4. CONCLUSÃO DO TERCEIRO CAPÍTULO

Este capítulo foi essencial para o começo do projeto pois ao definir e aprender a metodologia ajudou bastante no percurso da aplicação e de que maneira a realizar.

4 DIAGRAMAS UML

Especificação OMG:

''O UML é uma linguagem ou notação de diagramas para visualizar, especificar, construir e documentar artefactos de um determinado sistema de software. Esta linguagem oferece uma forma padronizada para escrever projetos de sistemas, conceptualmente permite definir os processos de negócio e funções de sistemas e de uma maneira concreta às linguagens de programação, esquemas de base de dados e componentes reutilizáveis de software.''

(Systems, 2012)

4.1. DFD CONTEXTO

O diagrama de fluxo de dados (DFD) representa o fluxo de dados num sistema de informação, assim como as sucessivas transformações que estes sofrem. O DFD é uma ferramenta gráfica que transcreve, de forma não técnica, a lógica do procedimento do sistema em estudo, sendo usada por diferentes métodos e principalmente pelos classificados como orientados a processos.

O diagrama de fluxo de dados apresenta sempre quatro objetos de um sistema de informação: fluxo de dados, processos, ficheiros de dados e entidades externas que recorrem a métodos e símbolos diferentes para representar cada objeto.

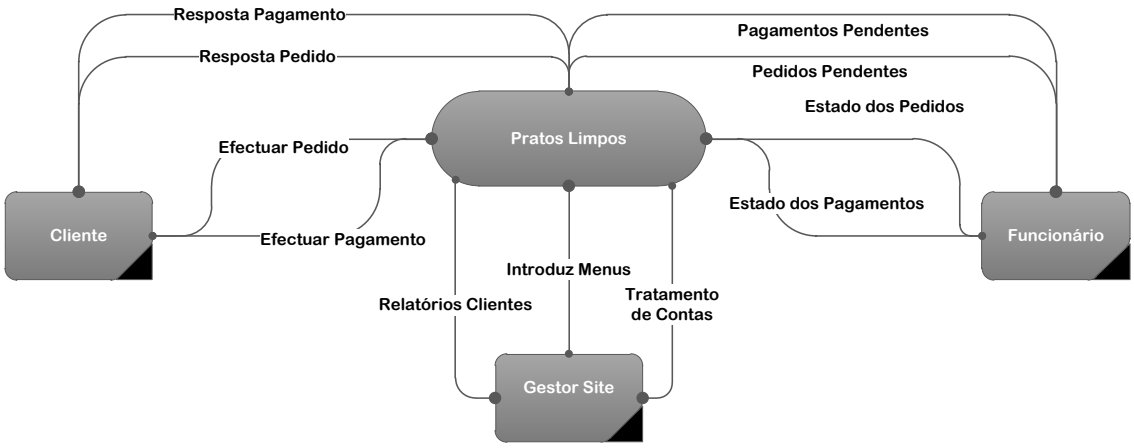


Figura 3: Diagrama Fluxo de Dados, Pratos Limpos

4.2. CASOS DE USO

Um caso de uso representa uma unidade discreta da interação entre um utilizador (humano ou máquina) e o sistema. Cada caso de uso tem uma descrição ao qual descreve a funcionalidade que irá ser construída no sistema proposto. Os casos de uso são tipicamente relacionados a "atores". Um ator é um humano ou entidade máquina que interage com o sistema para executar uma dada tarefa. É importante notar que não descreve como o *software* deverá ser construído, mas sim o seu comportamento.

4.2.1. **ATORES**

Um ator representa uma entidade externa que interage com o sistema, apesar da representação humanizada, os atores podem não ser só pessoas, mas também outros sistemas físicos ou lógicos. Os atores devem ser caracterizados através de uma pequena descrição, de forma a assegurar uma correta compreensão do significado do ator por todos os elementos envolvidos na análise do projeto, que neste caso são:

- Cliente: é o utilizador que tem como função efetuar pedidos e pagamentos através da aplicação;
- Funcionário: é a entidade responsável por efetuar o tratamento dos pedidos provenientes dos clientes e confirmar os pagamentos dos mesmos;
- Gestor: é o ator responsável pela gestão do *site*, gerir menus e respetivos preços, tratamento de contas (''banir'' e bloquear cliente, sendo estas umas das características mais fortes do gestor);

ATORES	OBJETIVOS
Cliente	Efetuar Pedido Efetuar Pagamento
Funcionário	Tratar do Estado dos Pedidos Confirmar Pagamentos
Gestor	Gerir Menus Tratar Contas

Tabela 1: Casos de Uso, Pratos Limpos

4.2.2. DIAGRAMA DE CASOS DE USO

Os diagramas de caso de uso são utilizados para a apresentação de requisitos e para assegurar que tanto o utilizador final como o perito em determinada área possuem um entendimento comum dos requisitos. O seu objetivo é mostrar o que o sistema deve efetuar e não como o vai fazer. Estes diagramas utilizam as seguintes abstrações de modelação: atores, casos de uso e relações (<<include>>,<<extend>>...).

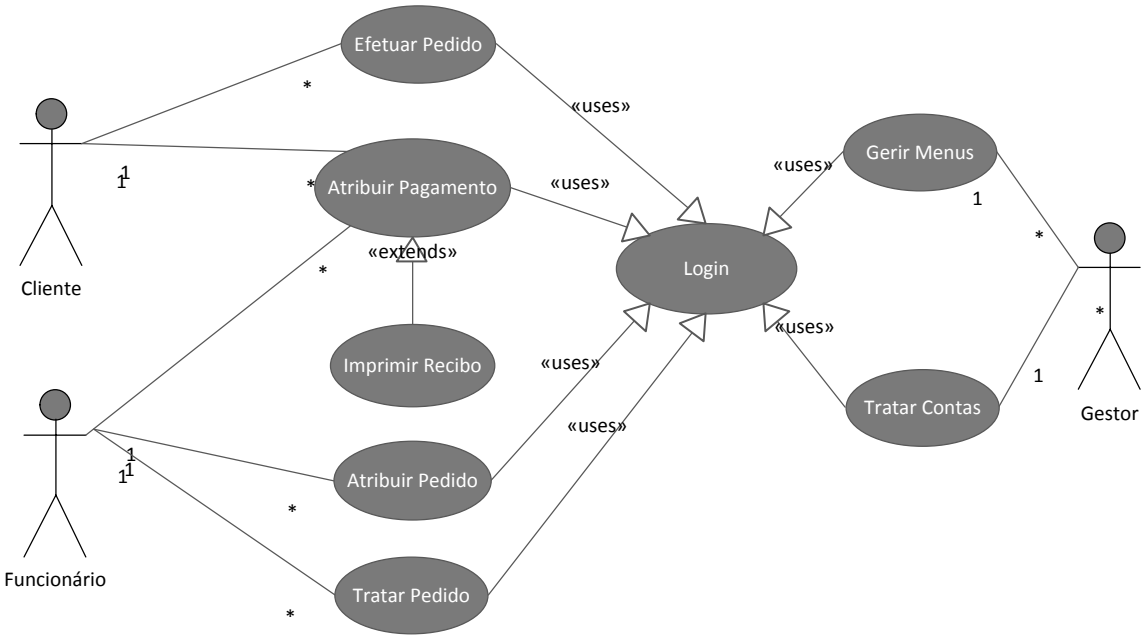


Figura 4: Diagrama Casos de Uso, Pratos Limpos

4.2.3. DESCRIÇÃO DOS CASOS DE USO

- **EFETUAR PEDIDO:**

A seguinte tabela é correspondente ao caso de uso efetuar pedido onde o ator responsável é o cliente. A função é mostrar como o cliente efetua um pedido e fica registado no sistema.

ATOR PRIMÁRIO	**CLIENTE**
Nome	Efetuar Pedido.
Descrição	O ator acede ao sistema para efetuar o pedido.
Pré-Condições	O utilizador precisa estar autenticado como cliente.
Caminho Principal	1. O caso de uso começa quando o cliente seleciona a opção efetuar pedido. 2. O sistema devolve os vários menus disponíveis para o cliente escolher. 3. O cliente adiciona os menus que pretende e respetivas quantidades. 4. O sistema pede confirmação e pergunta ao cliente se deseja efetuar o pagamento *on-line*. 5. O cliente confirma e responde à questão. 6. O sistema regista o pedido e atribui-lhe um código.
Caminhos Alternativos	3.1. Mensagem de erro apresentada porque o cliente não pode introduzir quantidades negativas e/ou carateres não numéricos. 5.1. O cliente cancelou o pedido. 5.2. O cliente decidiu efetuar o pagamento *on-line.*
Pós Condições	Após efetuada a operação o cliente recebe uma mensagem em como o pedido foi registado com sucesso.

Tabela 2: Caso de Uso: Efetuar Pedido, Pratos Limpos

- **GERIR MENUS:**

A seguinte tabela corresponde ao caso de uso gerir menus que é restrito ao administrador/gestor da aplicação. A função consiste em inserir, atualizar e apagar menus respeitando as especificações dadas pelo sistema.

ATOR PRIMÁRIO	**GESTOR**
Nome	Gerir Menu.
Descrição	O ator acede ao sistema para efetuar o pedido.
Pré-Condições	O utilizador precisa estar autenticado como administrador.
Caminho Principal	1. O caso de uso começa quando o gestor seleciona a opção gerir menus. 2. O sistema devolve os vários tipos de menus disponíveis para efetuar operações. 3. O gestor escolhe um tipo de menu. 4. O sistema devolve todos os menus disponíveis correspondentes ao tipo escolhido, bem como as várias opções que o gestor pode efetuar. 5. O gestor escolhe uma operação para efetuar. 5.1. Se o gestor escolheu a operação inserir então. 5.1.1. O sistema devolve um formulário com todas as especificações para preencher. 5.1.2. O gestor preenche e submete o formulário. 5.1.3. O sistema pede para confirmar operação. 5.1.4. O gestor confirma operação. 5.1.5. O sistema regista o menu e atribui-lhe um código. 5.2. Se o gestor escolheu a operação eliminar então. 5.2.1. O sistema pede para selecionar um menu para eliminar. 5.2.2. O gestor escolhe menu para eliminar. 5.2.3. O sistema pede para confirmar. 5.2.4. O gestor confirma operação. 5.2.5. O sistema elimina o menu. 5.3. Se o gestor escolheu a operação atualizar então. 5.3.1. O sistema pede para selecionar um menu para atualizar. 5.3.2. O gestor escolhe menu para atualizar. 5.3.3. O sistema devolve um formulário correspondente ao menu selecionado. 5.3.4. O gestor preenche e submete o formulário. 5.3.5. O sistema pede para confirmar. 5.3.6. O gestor confirma operação. 5.3.7. O sistema atualiza o menu.
Caminhos Alternativos	5.1.2.1 e 5.3.4.1 Mensagem de erro porque o gestor não pode inserir carateres diferentes ao pedido em cada parâmetro do formulário. • Em qualquer etapa da responsabilidade do gestor, este pode cancelar a operação a qualquer momento.
Pós Condições	Após efetuadas as operações dos pontos 5.1, 5.2 e 5.3 o gestor recebe uma mensagem em como a operação foi efetuada com sucesso e volta ao ponto 4.

Tabela 3: Caso de Uso: Gerir Menus, Pratos Limpos

- **TRATAR PEDIDO:**

A seguinte tabela é correspondente ao caso de uso tratar pedidos onde o ator responsável é o funcionário. Este caso de uso tem como função mostrar como se tratam todos os pedidos efetuados pelos clientes.

ATOR PRIMÁRIO	**FUNCIONÁRIO**
Nome	Tratar Pedidos.
Descrição	O ator acede ao sistema para tratar o pedido.
Pré-Condições	O utilizador precisa estar autenticado como funcionário.
Caminho Principal	1. O caso de uso começa quando o funcionário seleciona a opção tratar pedidos. 2. O sistema devolve os vários pedidos existentes na base de dados ordenados por forma de entrada com maior relevância para os pedidos ainda pendentes. 3. O funcionário seleciona o pedido que pretende atualizar. 4. O sistema pede para indicar o novo estado e observações do pedido. 5. O funcionário indica o novo estado e insere as observações necessárias. 6. O sistema pede para confirmar operação. 7. O funcionário confirma. 8. O sistema atualiza dados do pedido.
Caminhos Alternativos	3.1., 5.1. e 7.1. O funcionário pode cancelar a operação a qualquer momento. 5.1.1. Mensagem de erro porque o funcionário inseriu mal as observações.
Pós Condições	Após efetuada a operação o funcionário recebe uma mensagem em como o pedido foi atualizado com sucesso.

Tabela 4: Caso de Uso: Tratar Pedidos, Pratos Limpos

- **ATRIBUIR PAGAMENTO**

A seguinte tabela é correspondente ao caso de uso atribuir pagamento onde o ator responsável é o cliente. Este caso de uso só é requerido caso o cliente tenha escolhido a opção pré-pagamento via aplicação, pois ele poderá também pagar pessoalmente antes ou após a refeição.

ATOR PRIMÁRIO	**CLIENTE**
Nome	Atribuir Pagamento.
Descrição	O ator acede ao sistema para efetuar o pedido.
Pré-Condições	O utilizador precisa estar autenticado como cliente.
Caminho Principal	1. O caso de uso começa quando o cliente seleciona a opção efetuar pagamento. 2. O sistema calcula o preço final e devolve as várias opções de pagamento juntamente com o preço. 3. O cliente seleciona a forma de pagamento que deseja. 4. O sistema devolve formulário correspondente à forma de pagamento. 5. O cliente preenche e submete o formulário. 6. O sistema pede para confirmar operação. 7. O cliente confirma. 8. O sistema regista o pagamento e atribui-lhe um código.
Caminhos Alternativos	5.1. Mensagem de erro porque o cliente preencheu o formulário de forma incorreta. • O cliente pode cancelar a operação a qualquer momento.
Pós Condições	Após efetuada a operação o cliente recebe uma mensagem em como o pagamento foi efetuado com sucesso.

Tabela 5: Caso de Uso: Atribuir Pagamento, Pratos Limpos

4.3. DIAGRAMAS DE SEQUÊNCIA

O diagrama de sequência é um diagrama de interação que realça a ordem cronológica das mensagens entre objetos. Este diagrama é construído a partir do diagrama de casos de uso, onde se define em primeiro lugar o papel do sistema e depois como é que o *software* realiza o seu papel (sequência de operações).

- **DIAGRAMA DE SEQUÊNCIA: EFETUAR PEDIDO**

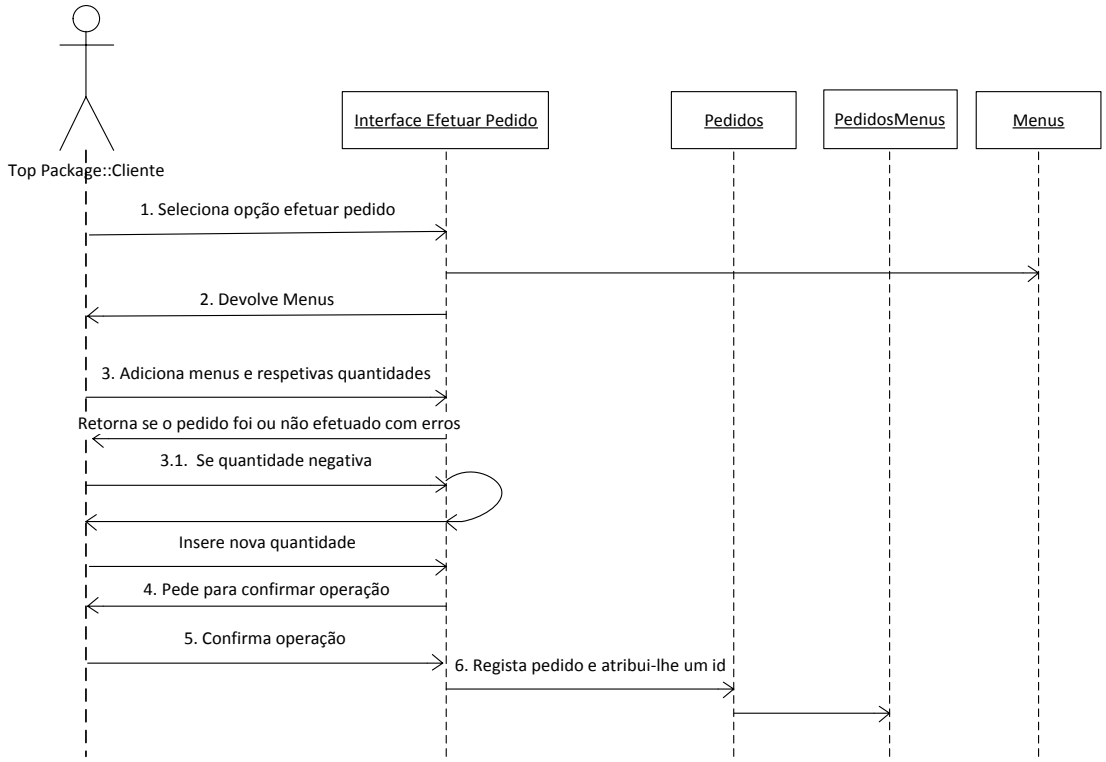


Figura 5: Diagrama de Sequência Efetuar Pedido, Pratos Limpos

- **DIAGRAMA DE SEQUÊNCIA: GERIR MENUS**

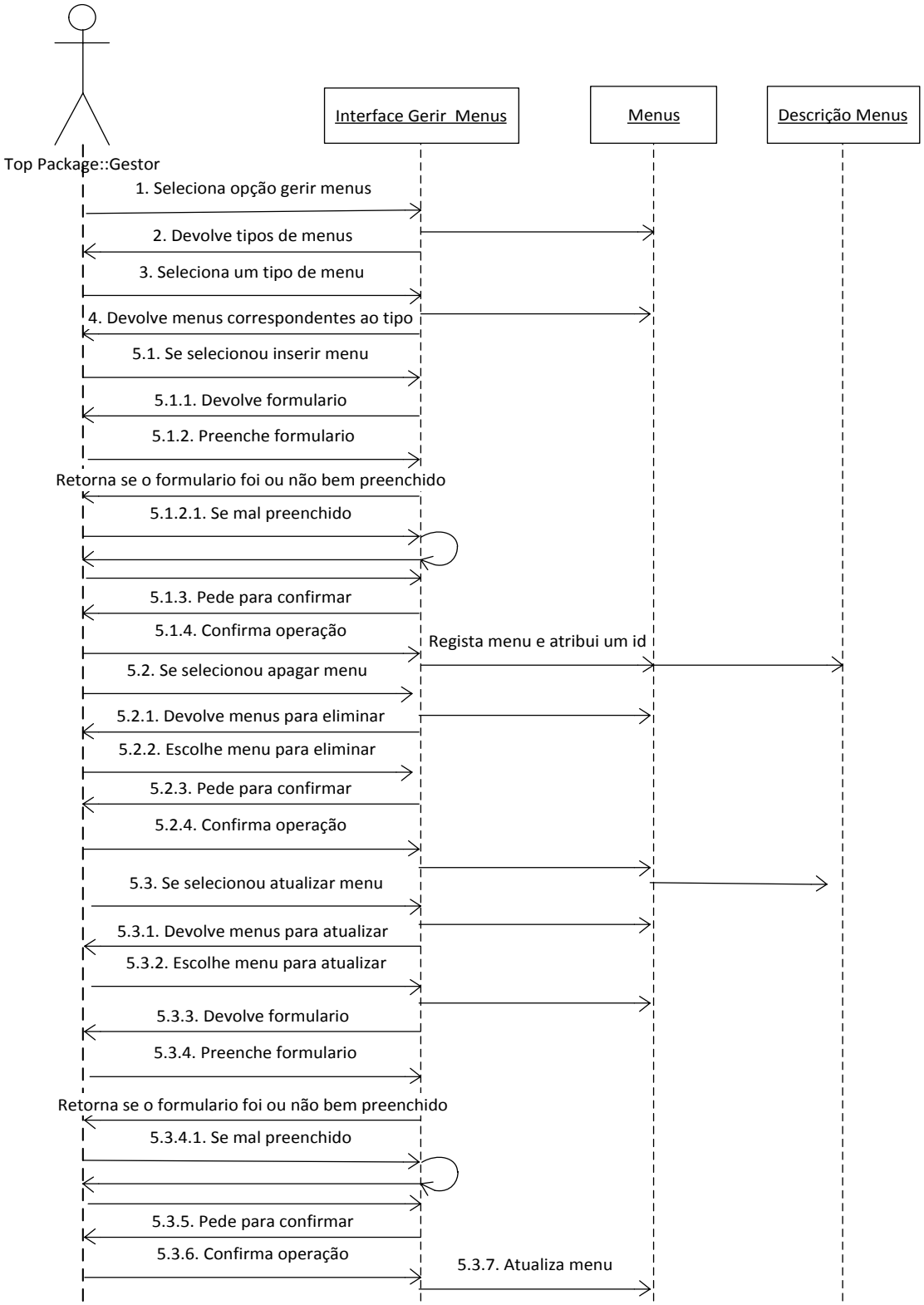


Figura 6: Diagrama de Sequência Gerir Menus, Pratos Limpos

- **DIAGRAMA DE SEQUÊNCIA: TRATAR PEDIDO**

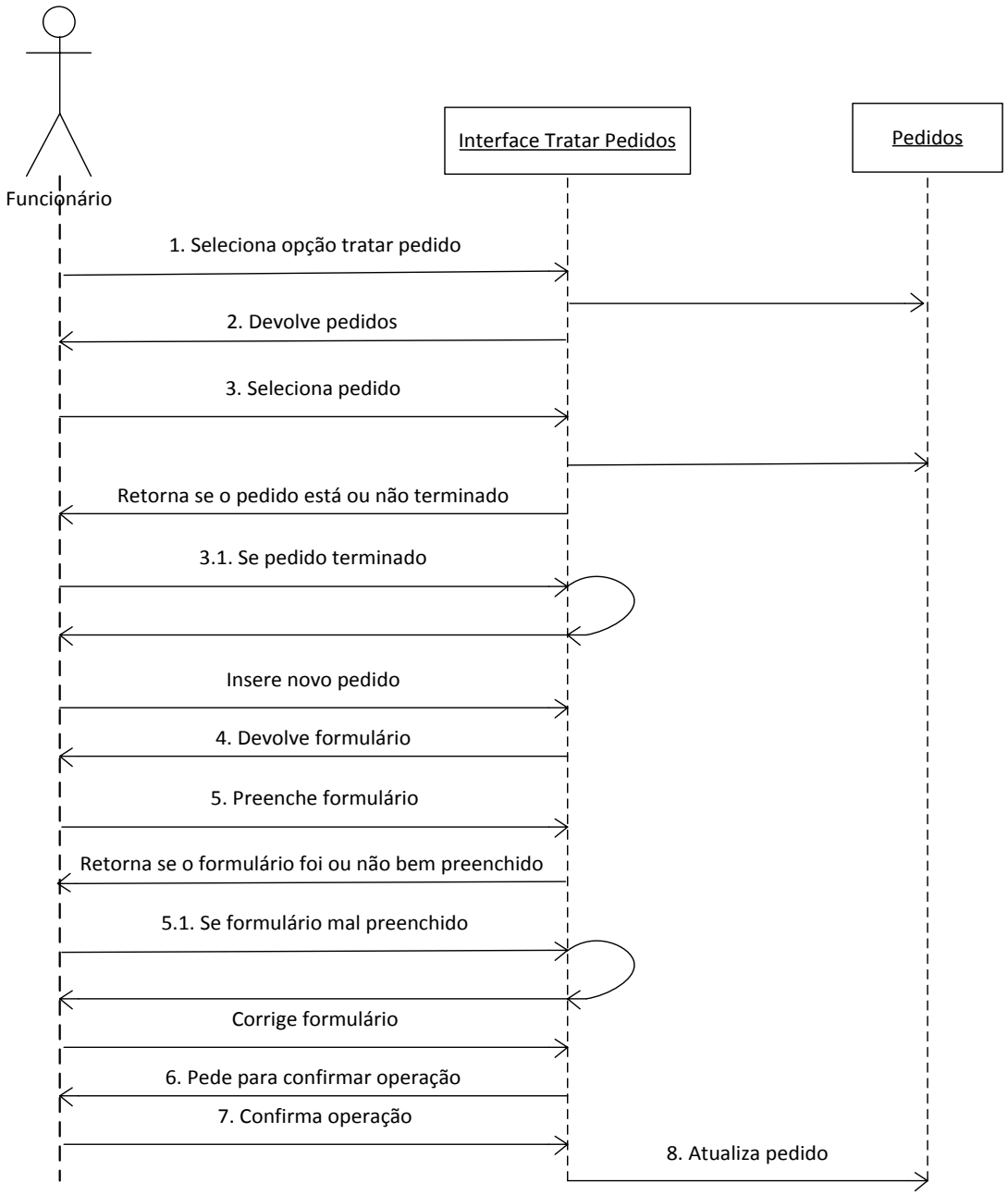


Figura 7: Diagrama de Sequência: Tratar Pedidos, Pratos Limpos

- **DIAGRAMA DE SEQUÊNCIA: ATRIBUIR PAGAMENTO**

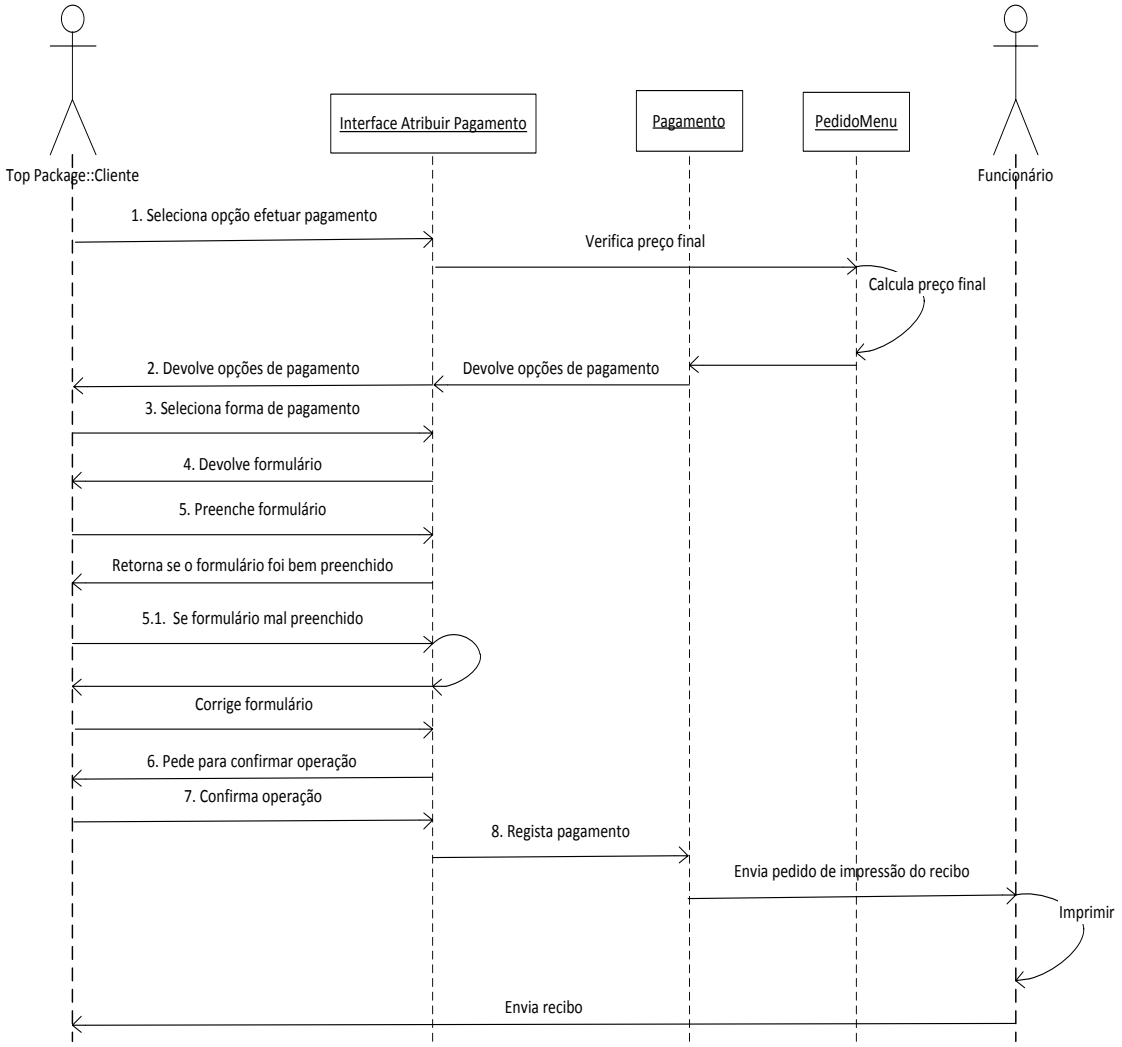


Figura 8: Diagrama de Sequência: Atribuir Pagamento, Pratos Limpos

4.4. DIAGRAMA DE CLASSES

O diagrama de classes é um dos diagramas mais importantes da linguagem *UML* pois representa o modelo geral de informação do sistema. É visto como o desenho arquitetónico da base de dados e contém todas as classes, hierarquia, relações e associações, descrevendo as operações e atributos das classes de uma forma estrutural.

Normalmente o diagrama de classes está sujeito a várias atualizações durante o desenvolvimento de um sistema e nem sempre obtemos a melhor solução para o problema, mas um diagrama de classes bem estruturado e normalizado, por vezes, diminui o tempo de implementação da nossa aplicação.

O diagrama de classes é composto pelos seguintes elementos abstratos de modelação:

- Classes de objetos;
- Relações de associação e generalização;
- Multiplicidade.

Existem várias teorias em como modelar um diagrama de classes preciso, mas normalmente é utilizado no seguinte conjunto independente de formas:

- Modelar o vocabulário de um sistema - Envolve o decidir sobre que abstrações estruturais fazem parte do sistema em estudo e quais estão fora das suas fronteiras.
- Modelar colaborações simples - Visualizar o sistema como um todo constituído por classes e suas relações que, através do trabalho em conjunto, fornecem um comportamento cooperativo.
- Modelar o esquema lógico de uma base de dados - Desenhar a estrutura de dados para uma BD relacional ou orientada por objetos, de forma a guardar a informação do sistema.

A **Figura 8** representa o diagrama de classes da aplicação Pratos Limpos.

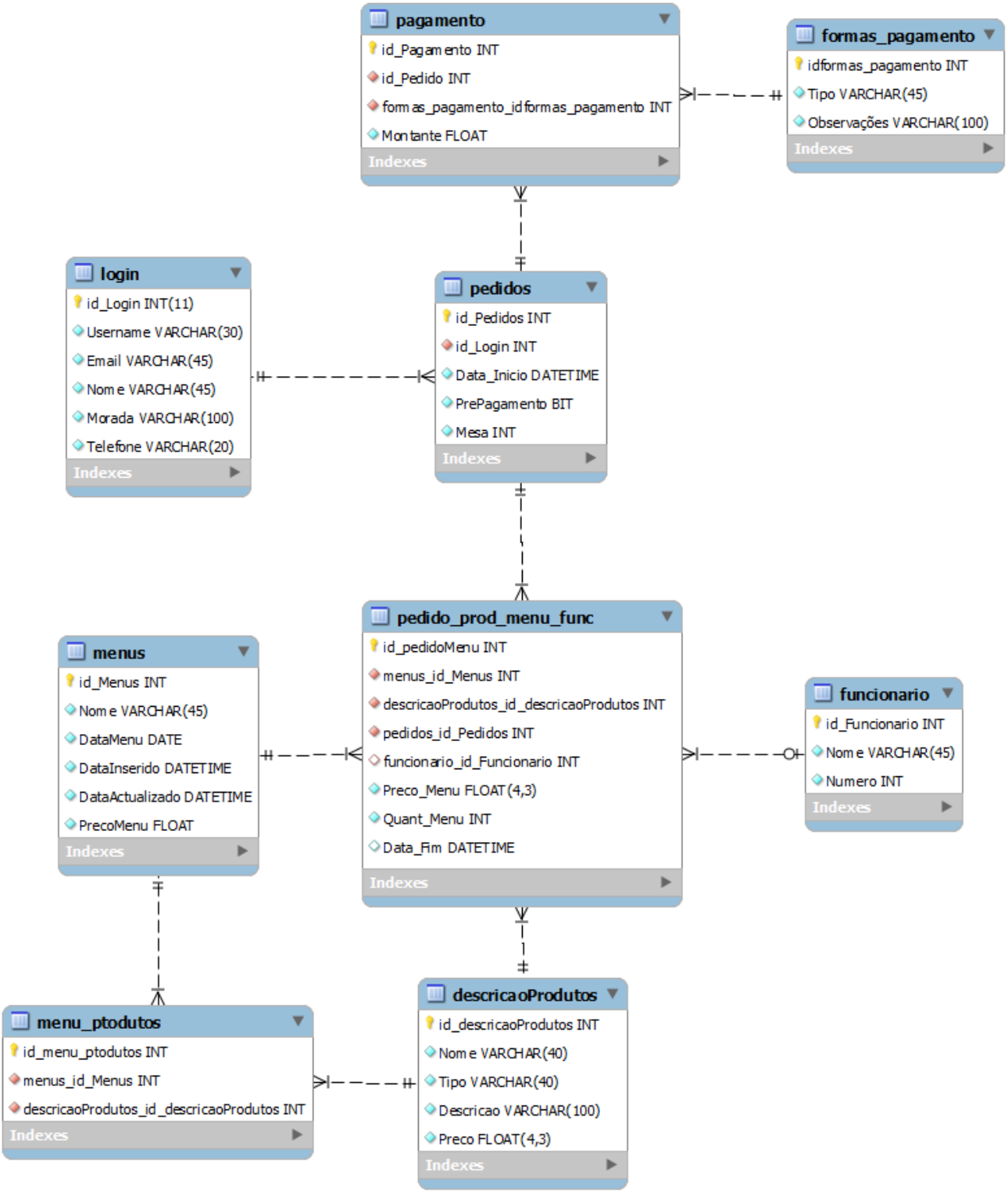


Figura 9: Diagrama de Classes, Pratos Limpos

4.4.1. SEMÂNTICA DAS CLASSES

- Dicionário de dados - O dicionário de dados consiste em descrever todos os dados que são pertinentes para o sistema. É essencial para descrever entradas, saídas, composição de depósitos de dados e alguns cálculos intermédios do modelo de dados. O dicionário de dados é uma ponte de referência de todos os elementos envolvidos na medida em que permite associar um significado a cada termo utilizado.

- **CLASSE LOGIN**

    - **DICIONÁRIO DE DADOS**

Nome do Campo	Tipo de Dados	Descrição	Valores Válidos	Formato	Restrições
id_Login	Numeração Automática	Número sequencial para identificar cada login	Maior que zero	Até 11 dígitos	Gerado pelo sistema e não alterável
Username	*Varchar*	*Username* do cliente para se autenticar na aplicação	Carateres de A a Z	Até 30 carateres	Obrigatório e não alterável
Email	*Varchar*	*Email* do cliente	Carateres de A a Z	Até 45 carateres	Obrigatório e alterável
Nome	*Varchar*	Nome do cliente	Carateres de A a Z	Até 45 carateres	Obrigatório e alterável
Morada	*Varchar*	Morada do cliente	Carateres de A a Z	Até 100 carateres	Facultativo e alterável
Telefone	*Varchar*	Telefone do cliente	Carateres alfanuméricos mais carateres especiais [‘+’, ‘(’, ‘)’]	Até 20 carateres	Facultativo e alterável

Tabela 6: Dicionário de Dados: Classe Login, Pratos Limpos

- **OPERAÇÕES**

NOME	DESCRIÇÃO
Inserir()	Operação que permite inserir o Login para um cliente. 1. Sistema gera o idLogin (incrementa uma unidade ao último idLogin). 2. Introduzir *Username*. 3. Introduzir *Email*. 4. Introduzir Nome. 5. Introduzir Morada. 6. Introduzir Telefone. 7. Criar novo Login para cliente.
Atualizar()	Operação que permite atualizar a conta de um cliente. 1. Selecionar o cliente através do idLogin. 2. Introduzir novo *Email.* 3. Introduzir novo Morada. 4. Introduzir novo Telefone. 5. Atualizar dados do cliente.
Consultar()	O sistema consulta a base de dados antes de introduzir um novo parâmetro. 1. O sistema indica qual o campo que pretende consultar. 2. Se (campo incorreto ou inexistente) 2.1. Introduzir novo campo. 3. Senão mostra resultado.
Eliminar()	Operação que permite eliminar uma determinada conta. 1. O gestor indica qual o *username* que pretende eliminar. 2. O sistema seleciona o idLogin correspondente a esse *username* na tabela login. 3. Se (login.idLogin = NULL) 3.1. Indica que não existe conta. 4. Senão 4.1. Seleciona pedido.idLogin 5. Se (pedido.idLogin = login.idLogin) 5.1. Elimina pedido.idLogin. 6. Elimina login.idLogin.

Tabela 7: Operações: Classe Login, Pratos Limpos

- **CLASSE PEDIDOS**

  - **DICIONÁRIO DE DADOS**

Nome do Campo	Tipo de Dados	Descrição	Valores Válidos	Formato	Restrições
id_Pedidos	Numeração Automática	Número sequencial para identificar cada pedido.	Maior que zero	Até 11 dígitos	Gerado pelo sistema e não alterável
id_Login	Numeração Automática *(Chave Estrangeira)*	Número sequencial para identificar cada login.	Maior que zero	Até 11 dígitos	Facultado pelo sistema e não alterável
DataInicio	*DateTime*	Altura em que o pedido foi efetuado	Algarismos de 0 a 9 mais carateres especiais '/' e ':'.	DD/MM/YYYY HH:MM:SS	Data do sistema e não alterável
PrePagamento	*Bit*	Indica se o cliente quer ou não pré-pagamento	Algarismo 0 ou 1	1 Carater	Obrigatório e alterável
Mesa	Inteiro	Mesa correspondente ao pedido	Algarismos de 0 a 9	Até 3 carateres	Obrigatório e alterável

Tabela 8: Dicionário de Dados: Classe Pedido, Pratos Limpos

  - **OPERAÇÕES**

NOME	DESCRIÇÃO
Inserir()	Operação que permite inserir um pedido. 1. Sistema gera o idPedido (incrementa uma unidade ao último idPedido). 2. Selecionar o *Username* do cliente da tabela login para ser atribuído ao idLogin. 3. O sistema introduz a data de início do pedido. 4. Introduzir o valor de Pré-pagamento. 5. Introduzir o número da mesa. 6. Cria novo registo de pedido.
Atualizar()	Operação que permite atualizar o pedido. 1. Selecionar o pedido através do idPedido. 2. Introduzir novo valor de Pré-pagamento. 3. Introduzir novo número de mesa. 4. Atualizar dados do pedido.
Consultar()	O sistema consulta a base de dados antes de introduzir um novo parâmetro. 1. O sistema indica qual o campo que pretende consultar. 2. Se (campo incorreto ou inexistente). 2.1. Introduzir novo campo. 3. Senão mostra resultado.

Tabela 9: Operações: Classe Pedido, Pratos Limpos

- **CLASSE PEDIDO_PROD_MENU_FUNC**

  - **DICIONÁRIO DE DADOS**

Nome do Campo	Tipo de Dados	Descrição	Valores Válidos	Formato	Restrições
id_pedidoMenu	Numeração Automática	Número sequencial para identificar cada pedido	Maior que zero	Até 11 dígitos	Gerado pelo sistema e não alterável
menus_id_Menus	Numeração Automática *(Chave Estrangeira)*	Número sequencial para identificar cada menu	Maior que zero	Até 11 dígitos	Facultado pelo sistema e não alterável
descricaoProdutos_ id_ descricaoProdutos	Numeração Automática *(Chave Estrangeira)*	Número sequencial para identificar cada produto (descrição)	Maior que zero	Até 11 dígitos	Facultado pelo sistema e não alterável
pedidos_id_Pedidos	Numeração Automática *(Chave Estrangeira)*	Número sequencial para identificar cada pedido	Maior que zero	Até 11 dígitos	Facultado pelo sistema e não alterável
funcionário_id_ Funcionário	Numeração Automática *(Chave Estrangeira)*	Número sequencial para identificar o funcionário	Maior ou igual a zero. Pode ser '*null*'	Até 11 dígitos	Facultado pelo sistema e alterável.
Preco_Menu	*Float*	Preço total de um determinado pedido consoante a quantidade pedida	Algarismos de 0 a 9 mais carater especial ','	Até 4 números inteiros e 3 números decimais	Facultado pelo sistema e alterável
Quant_Menu	Inteiro	Quantidade de um determinado menu para um determinado pedido	Algarismos de 0 a 9	Até 3 carateres	Obrigatório e alterável
Data_Fim	*DateTime*	Altura em que o pedido foi terminado	Algarismos de 0 a 9 mais carateres especiais '/' e ':'. Pode ser '*null*'.	DD/MM/YYYY HH:MM:SS	Data do sistema e alterável

Tabela 10: Dicionário de Dados: Classe Pedido_Prod_Menu_Func, Pratos Limpos

- **OPERAÇÕES**

NOME	DESCRIÇÃO
Inserir()	Operação que permite inserir um pedido relacionado com os produtos, menus e funcionário. 1. Sistema gera o id_PedidoMenu (incrementa uma unidade ao último id_PedidoMenu). 2. Selecionar o menu da tabela Menus para ser atribuído ao menus_id_Menus. 3. Selecionar o produto da tabela descricaoProdutos para ser atribuído ao descricaoProdutos_id_descricaoProdutos. 4. Selecionar o pedido da tabela pedidos para ser atribuído ao pedidos_id_Pedidos. 5. Inserir *'null'* no campo funcionário_id_Funcionário. 6. Introduzir a quantidade pretendida do menu. 6.1. Se (quantidade < 0) então 6.1.1. Voltar ao ponto 6 6.2. Senão introduzir quantidade 7. Selecionar o Preco_Menu dos menus 8. Calcular o preço total do pedido (pedido_prod_menu_func.Quant_Menu X Menus.Preco_Menu) 9. Introduzir Preco_Menu 10. Inserir *'null'* no campo Data_Fim. 11. Criar novo registo pedido_prod_menu_func.
Atualizar()	Operação que permite atualizar o pedido_prod_menu_func. 1. Selecionar o pedido através do id_PedidoMenu. ***SE CLIENTE*** 2. Introduzir a nova quantidade pretendida do menu. 2.1. Se (quantidade < 0) então 2.1.1. Voltar ao ponto 2 2.2. Senão introduzir quantidade 3. Selecionar o Preco_Menu dos menus 4. Calcular o preço total do pedido (pedido_prod_menu_func.Quant_Menu X Menus.Preco_Menu) 5. Introduzir novo Preco_Menu 6. Atualizar dados do pedido_prod_menu_func. ***SE FUNCIONÁRIO*** 2. Se (funcionário_id_Funcionário == NULL) então 2.1. Selecionar o funcionário para ser atribuído ao funcionário_id_Funcionário. 3. Senão 3.1. Introduzir data do sistema Data_Fim. 4. Atualizar dados do pedido_prod_menu_func.
Consultar()	O sistema consulta a base de dados antes de introduzir um novo parâmetro. 1. O sistema indica qual o campo que pretende consultar. 2. Se (campo incorreto ou inexistente). 2.1. Introduzir novo campo. 3. Senão mostra resultado.

Tabela 11: Operações: Classe pedido_prod_menu_func, Pratos Limpos

- **CLASSE MENUS**

  - **DICIONÁRIO DE DADOS**

Nome do Campo	Tipo de Dados	Descrição	Valores Válidos	Formato	Restrições
id_Menus	Numeração Automática	Número sequencial para identificar cada menu	Maior que zero	Até 11 dígitos	Gerado pelo sistema e não alterável
Nome	*Varchar*	Nome do menu	Carateres Alfanuméricos	Até 45 carateres	Obrigatório e não alterável
DataMenu	*DateTime*	Data do menu.	Algarismos de 0 a 9 mais carateres especiais '/' e ':'. Pode ser '*null*'	DD/MM/YYYY HH:MM:SS	Data do sistema e alterável
DataInserido	*DateTime*	Altura em que o menu foi inserido	Algarismos de 0 a 9 mais carateres especiais '/' e ':'. Pode ser '*null*'	DD/MM/YYYY HH:MM:SS	Data do sistema e não alterável
DataAtualizado	*DateTime*	Altura em que o menu foi atualizado	Algarismos de 0 a 9 mais carateres especiais '/' e ':'. Pode ser '*null*'	DD/MM/YYYY HH:MM:SS	Data do sistema e alterável
PrecoMenu	*Float*	Preço total do menu	Algarismos de 0 a 9 mais carater especial ','	Até 4 números inteiros e 3 números decimais	Obrigatório e alterável

Tabela 12: Dicionário de Dados: Classe Menus, Pratos Limpos

- **OPERAÇÕES**

NOME	DESCRIÇÃO
Inserir()	Operação que permite inserir um menu. 1. Sistema gera o id_Menus (incrementa uma unidade ao último id_Menus). 2. Introduzir o nome do menu. 3. Introduzir a data do menu. 4. Introduzir a data do sistema em DataInserido. 5. Introduzir a data do sistema em DataAtualizado. 6. Introduzir o preço do menu. 7. Criar novo registo menus.
Atualizar()	Operação que permite atualizar um menu. 1. Introduzir a nova data do menu. 2. Introduzir a data do sistema em DataAtualizado. 3. Introduzir o novo preço do menu. 4. Atualizar dados de menus.
Consultar()	O sistema consulta a base de dados antes de introduzir um novo parâmetro. 1. O sistema indica qual o campo que pretende consultar. 2. Se (campo incorreto ou inexistente). 2.1. Introduzir novo campo. 3. Senão mostra resultado.

Tabela 13: Operações: Classe Menus, Pratos Limpos

- **CLASSE DESCRICAOPRODUTOS**

  - **DICIONÁRIO DE DADOS**

Nome do Campo	Tipo de Dados	Descrição	Valores Válidos	Formato	Restrições
id_descricaoProdutos	Numeração Automática	Número sequencial para identificar cada produto	Maior que zero	Até 11 dígitos	Gerado pelo sistema e não alterável
Nome	*Varchar*	Nome do produto	Carateres Alfanuméricos	Até 40 carateres	Obrigatório e não alterável
Tipo	*Varchar*	Tipo do produto	Carateres Alfanuméricos	Até 40 carateres	Obrigatório e não alterável
Descricao	*Varchar*	Descrição do produto	Carateres Alfanuméricos	Até 100 carateres	Facultativo e alterável
Preco	*Float*	Preço do produto	Algarismos de 0 a 9 mais carater especial ‘,’	Até 4 números inteiros e 3 números decimais	Obrigatório e alterável

Tabela 14: Dicionário de Dados: Classe descricaoProdutos, Pratos Limpos

  - **OPERAÇÕES**

NOME	DESCRIÇÃO
Inserir()	Operação que permite inserir um produto. 1. Sistema gera o id_descricaoProdutos (incrementa uma unidade ao último id_descricaoProdutos). 2. Introduzir o nome do produto. 3. Introduzir o tipo do produto. 4. Introduzir a descrição do produto. 5. Introduzir o preço do produto. 6. Criar novo registo descricaoProdutos.
Atualizar()	Operação que permite atualizar um produto. 1. Introduzir a nova descrição do produto. 2. Introduzir o novo preço do produto. 3. Atualizar dados de descricaoProdutos.
Consultar()	O sistema consulta a base de dados antes de introduzir um novo parâmetro. 1. O sistema indica qual o campo que pretende consultar. 2. Se (campo incorreto ou inexistente). 2.1. Introduzir novo campo. 3. Senão mostra resultado.

Tabela 15: Operações: Classe descricaoProdutos, Pratos Limpos

- **CLASSE MENU_PRODUTOS**

  - **DICIONÁRIO DE DADOS**

Nome do Campo	Tipo de Dados	Descrição	Valores Válidos	Formato	Restrições
id_menu_produtos	Numeração Automática	Número sequencial para identificar cada menu relacionado com produtos	Maior que zero	Até 11 dígitos	Gerado pelo sistema e não alterável
Menus_id_Menus	Numeração Automática *(Chave Estrangeira)*	Número sequencial para identificar cada menu	Maior que zero	Até 11 dígitos	Facultado pelo sistema e não alterável
descricaoProdutos_id_descricaoProdutos	Numeração Automática *(Chave Estrangeira)*	Número sequencial para identificar cada produto	Maior que zero	Até 11 dígitos	Facultado pelo sistema e alterável

Tabela 16: Dicionário de Dados: Classe Menu_Produtos, Pratos Limpos

  - **OPERAÇÕES**

NOME	DESCRIÇÃO
Inserir()	Operação que permite inserir produtos relacionados com menus. 1. Sistema gera o id_menu_produtos (incrementa uma unidade ao último id_menu_produtos). 2. Selecionar o id_Menus da tabela Menus para ser atribuído a menus_id_Menus. 3. Selecionar o id_descricaoProdutos da tabela descricaProdutos para ser atribuído a descricaoProdutos_id_descricaoProdutos. 4. Criar novo registo menu_produtos.
Atualizar()	Operação que permite atualizar produtos relacionados com menus. 1. Selecionar menus_id_menus para introduzir novos produtos. 2. Selecionar o id_descricaoProdutos da tabela descricaProdutos para ser atribuído a descricaoProdutos_id_descricaoProdutos. 3. Atualizar dados de descricaoProdutos.
Consultar()	O sistema consulta a base de dados antes de introduzir um novo parâmetro. 1. O sistema indica qual o campo que pretende consultar. 2. Se (campo incorreto ou inexistente). 2.1. Introduzir novo campo. 3. Senão mostra resultado.

Tabela 17: Operações: Classe menu_produtos, Pratos Limpos

4.5. DIAGRAMA DE ATIVIDADES

O diagrama de atividades constitui um elemento de modelação simples, mas eficaz, para descrever fluxos de trabalho numa organização ou para detalhar operações de uma classe, incluindo comportamentos que possuam processamento paralelo.

4.5.1. *DIAGRAMA DE ATIVIDADES: GERAL*

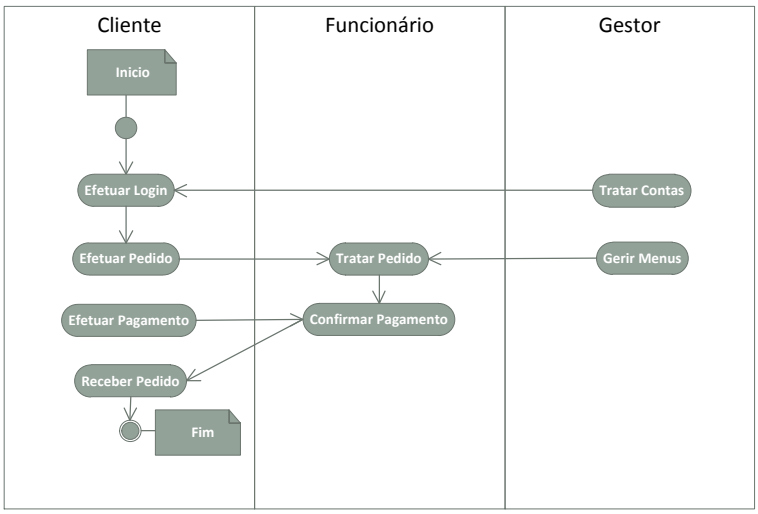


Figura 10: Diagrama de Atividades: Geral, Pratos Limpos

4.5.2. *DIAGRAMA DE ATIVIDADES: DISTRIBUIR PEDIDOS*

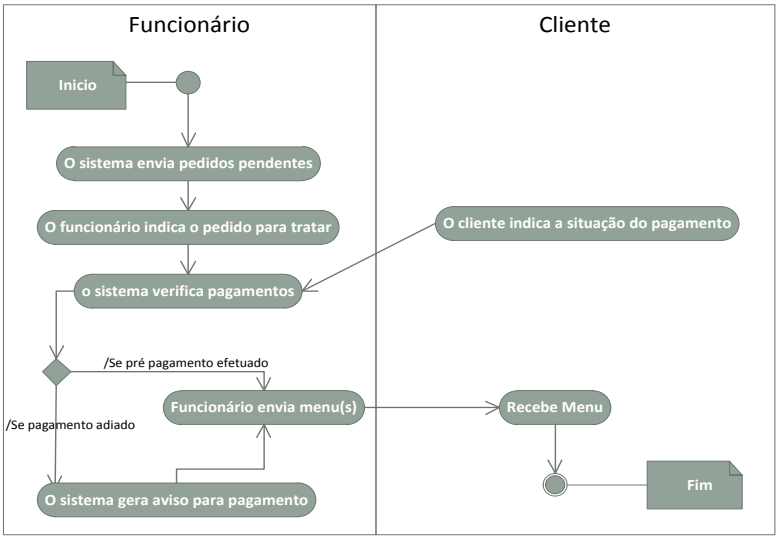


Figura 11: Diagrama de Atividades: Distribuir Pedidos, Pratos Limpos

4.6. DIAGRAMA DE ESTADOS

O diagrama de estados é utilizado para descrever o comportamento de um objeto. Na modelação de um sistema de informação devesse criar um diagrama de estados somente para cada classe de objeto que tenha um comportamento dinâmico relevante como, por exemplo, os objetos de controlo ou objetos de interface. Estes diagramas têm uma certa semelhança aos diagramas de atividades. A principal diferença entre ambos é que o diagrama de estados é centrado no objeto, enquanto o diagrama de atividades é centrado no processo.

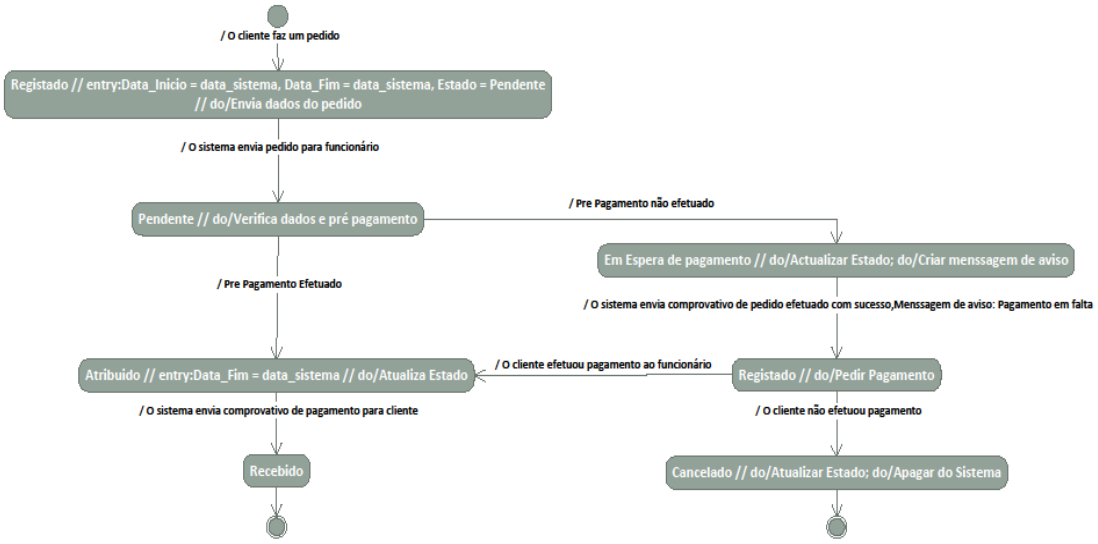


Figura 12: Diagrama de Estados: Pedido, Pratos Limpos

4.7. DIAGRAMA DE COMPONENTES

Um diagrama de componentes mostra um conjunto de componentes e as suas relações. Os candidatos a serem componentes do sistema são os itens que desempenham uma funcionalidade que é utilizada constantemente pelo sistema. Um componente representa um empacotamento físico de elementos relacionados logicamente.

Os diagramas de componentes têm como objetivo:

- Organizar o código fonte (ambiente de desenvolvimento).
- Construir uma *release* executável (ambiente de produção).
- Especificar componentes como base de dados e outros.

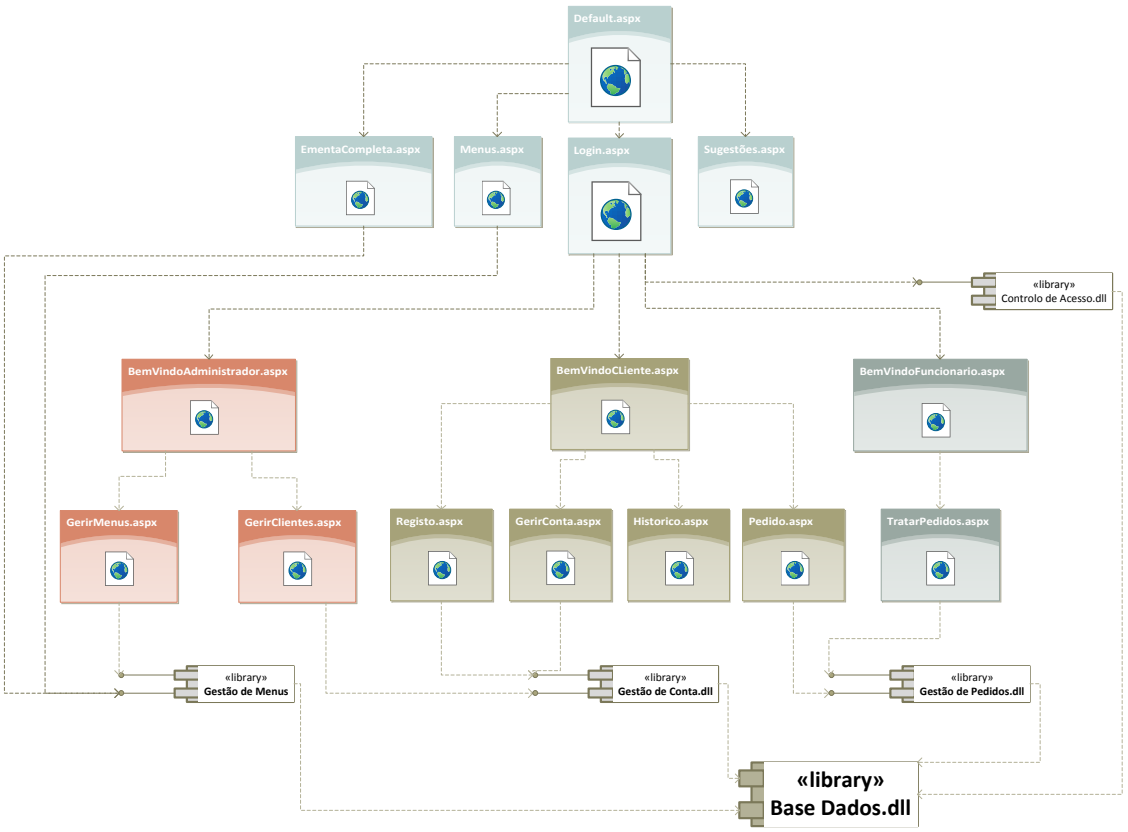


Figura 13: Diagrama de Componentes, Pratos Limpos

4.8. DIAGRAMA DE INSTALAÇÃO

O diagrama de instalação ilustra a arquitetura do sistema em termos de nós (*nodes*) que efetuam o processamento de componentes. Na prática, permite demonstrar como o *hardware* estará organizado e como os componentes (software) estarão distribuídos, estabelecendo assim a sua relação física.

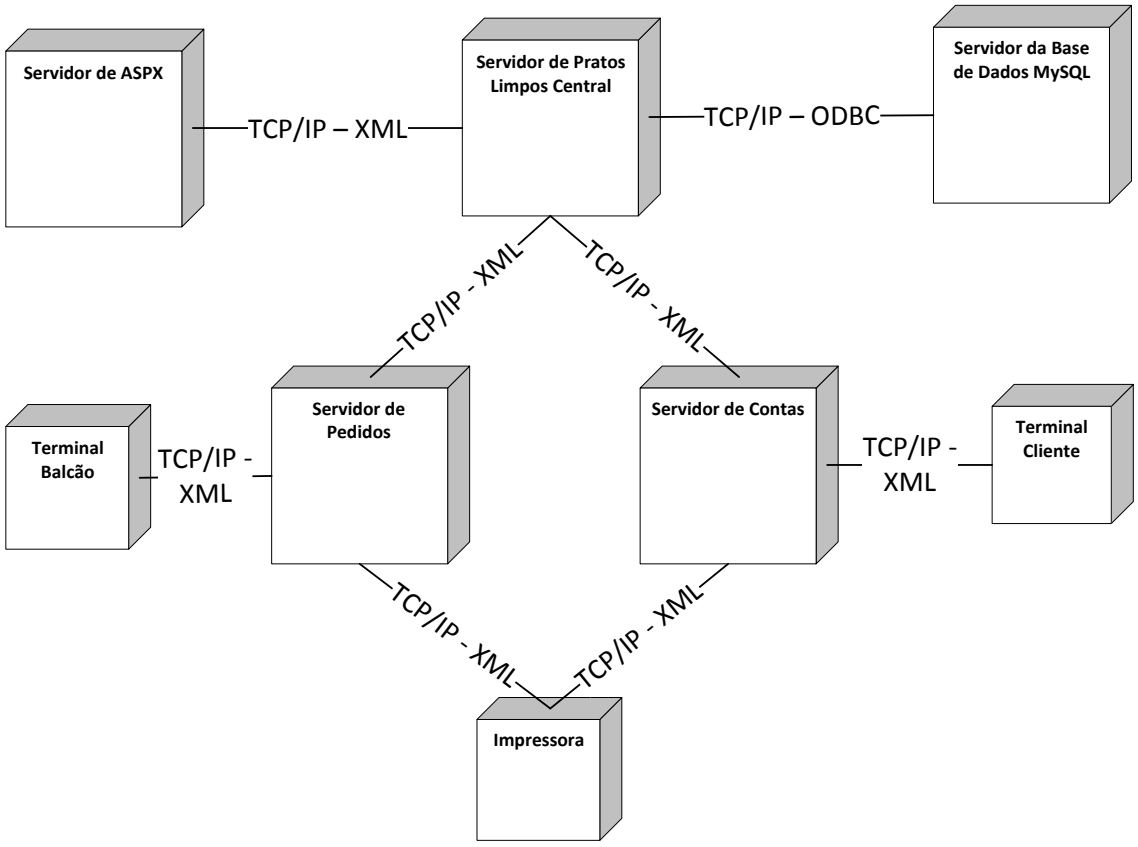


Figura 14: Diagrama de Instalação, Pratos Limpos

5 ESTADO DA ARTE

Neste capítulo irei expôr algumas aplicações já implementadas e semelhantes à aplicação ***Pratos Limpos.*** No final do capítulo são apresentadas as diferenças e vantagens da aplicação aqui falada, a todas as outras.

5.1. RESPONSIVE WEB DESIGN: *EXEMPLOS*

Os seguintes exemplos dizem respeito a aplicações que implementem uma ideia de *Responsive Web Design* (teórica detalhada no capitulo 2 e 6 deste relatório).

Hoje em dia é habitual encontrar inúmeras aplicações *web* que implementem esta ideia, pois uma grande percentagem de utilizadores acede a páginas *web* via *smartphone*, *tablet*, computador pessoal, etc. e devido a esta variedade de dispositivos, os *designers/*programadores foram obrigados a criar um método de desenho *web* que respondesse de forma "flexível" a estes diferentes tamanhos de ecrãs.

Em Portugal já se encontram algumas páginas *web* deste tipo, como é o caso do jellycode.pt que fornece um *design* muito interativo com o utilizador e sem restrições com o dispositivo com que acedemos, totalmente "*responsive"* e "*user-friendly*".

Figura 15: *Website* JellyCode.pt

"O crescente número de utilizadores de dispositivos móveis como o iPhone, iPad e outros smartphones e tablets, criaram uma nova exigência no desenvolvimento de websites: a adaptação do web site ao formato destes dispositivos de tamanho mais reduzido, com navegadores internet mais simples. A JellyCode desenvolve websites que se adaptam a dispositivos móveis, promovendo a leitura fácil de conteúdos e facilidade de navegação." (JellyCode, 2012)

O *JellyCode* é uma empresa que desenvolve *websites* flexíveis e fornece todo o serviço on-line. Mas se o interesse é aprender a desenhar o nosso próprio *site* do zero, sem ter de os comprar a alguém, a própria internet fornece variadíssimos guias e/ou tutoriais.

Tutorial de como fazer um *website* flexível (by NETMAGAZINE):

URL: http://www.netmagazine.com/tutorials/build-responsive-site-week-designing-responsively-part-1

Em relação a este tutorial feito pela *netmagazine*, serviu de base para o projeto *PratosLimpos*, pois algumas ideias e conhecimentos foram obtidos através deste guia.

Figura 16: *Website* netmagazine.com: Tutorial RWD

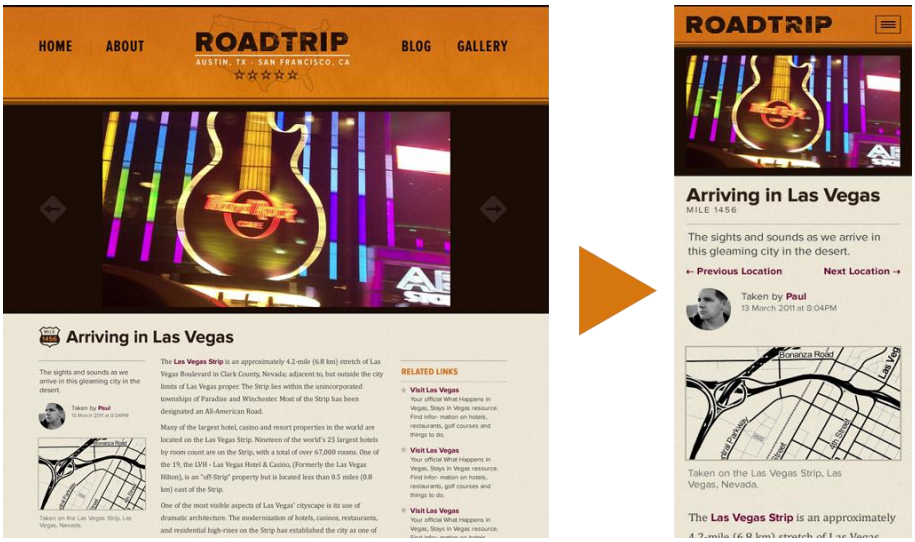


Figura 17: Website roadtrip.paulrobertlloyd.com: Tutorial

Achei este tutorial muito elucidativo e simples para quem ainda é principiante nesta área, como é o meu caso. Seguindo este tutorial podemos aprender como tornar toda a tipografia flexível e bem ilegível nos *sites,* como desenhar um layout simples e *responsive* e como conjugar todos os métodos das CSS3 de maneira a obter um *website* totalmente flexível às janelas dos diferentes tipos de *browsers*. O resultado encontra-se publicado em: http://roadtrip.paulrobertlloyd.com, apenas informativo, este *website* é um bom exemplo de *Responsive Web Design.*

Como este exemplo acima referido, existem muitos outros na internet, onde exemplificam esta nova abordagem com breves explicações e como começar a desenvolve-la.

5.2. INAMO RESTAURANT: PIONEIRO EM RESTAURANTES INTERATIVOS

Inamo é um restaurante Oriental situado em Londres, pioneiro no uso de mesas interativas expostas aos seus clientes. Estas mesas são ecrãs táteis, com o sistema operativo Windows e o *software E-Table.* Elas proporcionam ao cliente uma experiência única ao efetuar os seus pedidos, ao ver os menus, ao fazer os seus próprios menus e até jogar, ver a *webcam* da cozinha, ver mapas da cidade, entre outras aplicações.

Figura 18: *Inamo*: Restaurante com experiência E-Table

Como foi referido acima, utiliza o sistema *E-Table*, que é o cérebro destas mesas, pois é ele que permite esta interação entre cliente e mesa. Foi criado em 2005 e lançado inicialmente para o Windows XP. Hoje em dia encontra-se implementado em variadíssimos restaurantes espalhados por todo o Mundo.

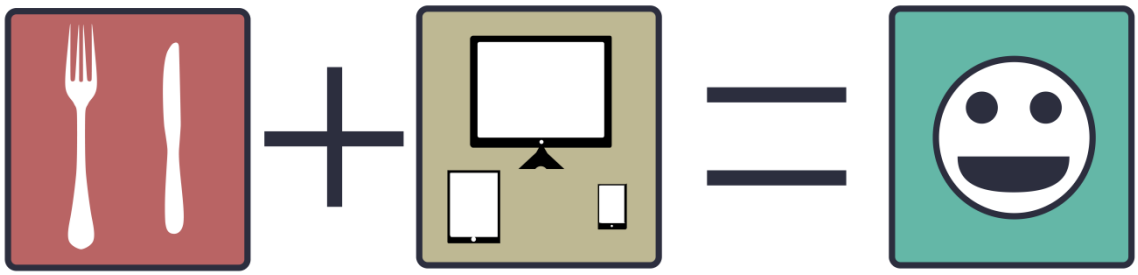
5.3. TELEPIZZA.PT

O *website* da *Telepizza* Portugal, fornece um serviço ao cliente onde este pode efetuar o pedido desejado nesta aplicação sem dirigir-se ao estabelecimento, respeitando todos os dados pedidos e a sua encomenda será entregue ao domicílio de imediato. O utilizador tem de estar autenticado e para isso terá de fornecer a morada válida do local para onde a encomenda será efetuada. Pode ver menus, fazer o seu próprio menu, verificar o tempo que falta para receber a encomenda, entre outras funções.

Figura 19: Website telepizza.pt

Este *website* é uma forma de evitar telefonemas ou idas desnecessárias ao estabelecimento da *telepizza* para efetuar uma encomenda, pois assim é concluído através do *site* sem ter de sair de casa.

5.4. PRATOSLIMPOS: *EXPERIENCE*



Nos capítulos anteriores foram mencionadas aplicações já implementadas e comercializadas, onde usufruem das novas tecnologias para dar aos utilizadores uma nova experiência dinâmica com os seus serviços.

A aplicação *Pratos Limpos* acaba por ser mais uma forma de interagir com o utilizador via tecnologia e pretende aplicar algumas das técnicas acima referidas, dando novos atributos a quem a utiliza.

PRATOSLIMPOS VS. RESPONSIVE WEB DESIGN

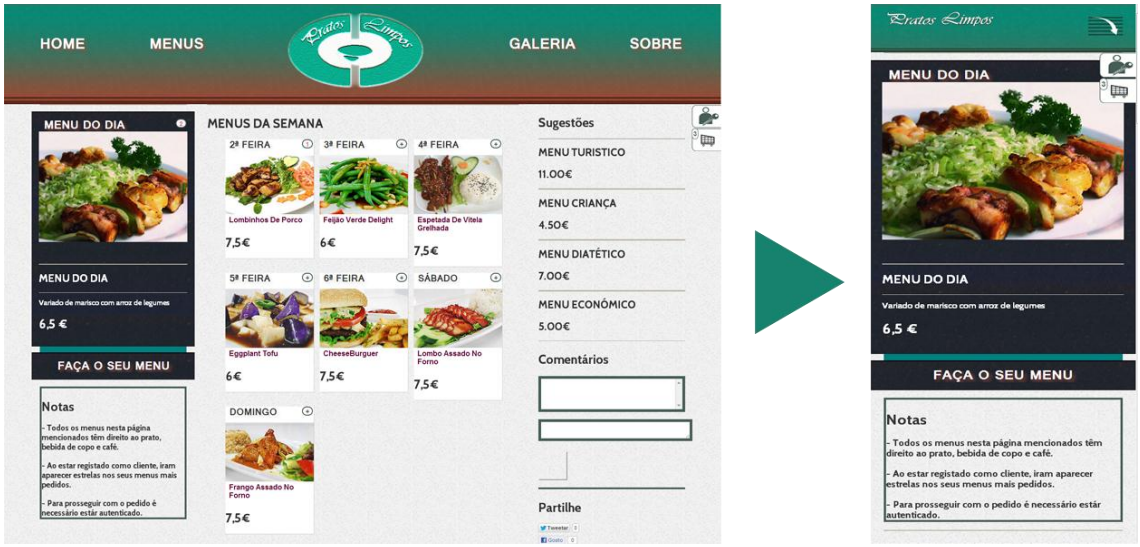


Figura 20: Website PratosLimpos

Utiliza a técnica *Responsive Web Design* para tornar o *website* flexível ao tamanho da janela do *browser*, com isto o utilizador não tem qualquer dificuldade ao navegar em *smartphone, tablet*, computador pessoal, etc.. Contém uma tipografia cuidada e um *layout* simples e *responsive*.

PRATOSLIMPOS VS. INAMO RESTAURANT

Em comparação com o *Inamo Restaurant,* evita custos elevados ao restaurante para implementar uma tecnologia avançada, proporcionando as mesmas funções. O único requisito ao restaurante é que disponibilize um serviço de internet aos seus clientes e estes têm de aceder através de um dispositivo com tecnologia *browser* e *wireless*.

Figura 21: Pratos Limpos: PC vs. *Smartphone*

PRATOSLIMPOS VS. TELEPIZZA.PT

Em relação ao serviço *Telepizza.pt*, com algumas modificações, *PratosLimpos* pode muito bem vir a ser implementado com o mesmo objetivo e o cliente poder efetuar encomendas para o domicílio, até mesmo através do seu *smartphone*. O conceito é parecido visto que o cliente tem de estar autenticado e é ele que tem controlo sobre os pedidos que deseja, sem ter de entrar em contacto com terceiros. A diferença passa por ser elegível em qualquer dispositivo (móvel, outros) e o acesso é feito dentro do estabelecimento e não fora como o da *Telepizza*.

Como desconheço os meios e métodos usados, do ponto de vista do funcionário e manutenção de conteúdos nos exemplos dados, não refiro que *PratosLimpos* seja diferente ou semelhante neste aspeto, mas aqui, todas as funções estão incorporadas no próprio *website* e assim tornando uma só aplicação única e eficaz.

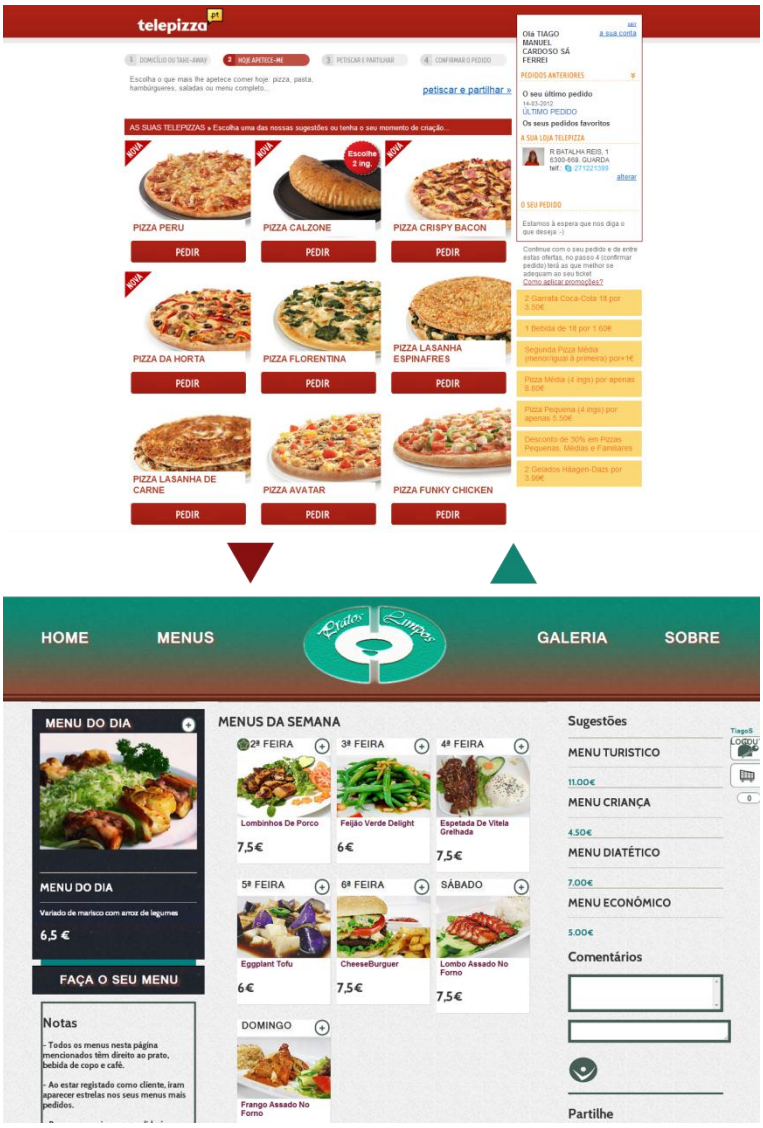


Figura 22: *PratosLimpos* vs. *Telepizza*

6 IMPLEMENTAÇÃO

6.1. INTRODUÇÃO

A implementação da solução passa basicamente por mostrar o detalhe e os conhecimentos práticos obtidos após uma vasta pesquisa e estudos efetuados nestes últimos anos na área da informática. Aqui apresento uma possível solução para um problema já estudado mas pouco esclarecido ainda, como é o caso do *Responsive Web Design* e pretendo com isto criar uma única aplicação direcionada à restauração.

Para ser mais óbvio e concreto, as soluções aqui estudadas foram ao nível do *design* da aplicação, da interação com a linguagem ASP.NET, ligações a base de dados e restrições através do servidor. Todos estes tópicos serão aqui descritos e exemplificados de acordo com a aplicação.

Nota: Antes de apresentar a aplicação e soluções encontradas é importante salientar que esta aplicação ainda é um simples protótipo, mas já representa os tópicos mais importantes aqui falados, mesmo assim alguns detalhes ainda não estão bem definidos e por isso não poderei aprofundar algumas questões particulares.

6.2. IMPLEMENTAR EM *RESPONSIVE WEB DESIGN*

Hoje em dia os dispositivos móveis já são uma questão bem esclarecida e usual no dia-a-dia das pessoas, pois praticamente este novo "micro ecrã tátil" é recorrente nas nossas mãos. Com um simples toque dos dedos se pode navegar, jogar, consultar, organizar, etc.. Mas ainda se encontram alguns problemas neste meio. Uma solução para a navegação na *web* nestes dispositivos é mesmo este *Responsive Web Design,* onde basicamente este conceito vem tentar mudar as ideias dos *web designers,* pois até ao dia de hoje desenhavam *websites* inteiramente estáticos, recorrendo a documentos já definidos ou criando *templates* com uma largura e altura, dimensão e forma pré-definidas, desprezando por completo o tipo de dispositivo que o iria executar.

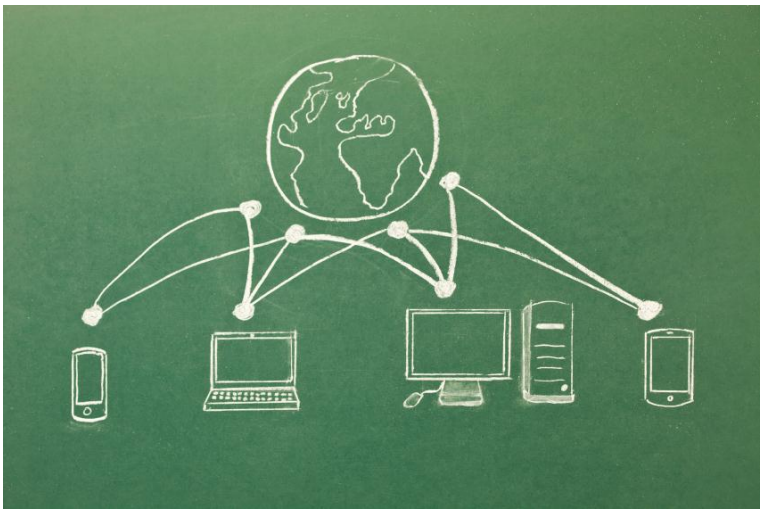


Figura 23: Responsive Web Design

"… we create a "canvas" in our favorite image editor, a blank document with a width and height, with dimension and shape. The problem with this approach is that we're one step removed from our actual canvas: the browser window, and all of its inconsistencies and imperfections. Because let's face it: once they're published online, our designs are immediately at the mercy of the people who view them—to their font settings, to the color of their display, to the shape and size of their browser window…"

(Marcotte, 2011)

Sendo mais concreto, o problema não se baseia só por estes dispositivos terem ecrãs de dimensões pequenas, pois anteriormente, já nos deparávamos com a dificuldade de navegação quando reduzíamos o tamanho das páginas ou janelas dos *browsers*. Num *site* de filosofia estática, onde o conteúdo por página é extenso, ao reduzirmos o tamanho de visualização o resultado é o aparecimento da barra horizontal para efetuarmos *scroll* para esquerda ou direita á busca do conteúdo desejado, o que torna a aplicação pouco amigável e pouco funcional.

Esta ideia centra-se em criar *websites* flexíveis para combater estes problemas, ser o mais *user experience* possível e adaptável aos ambientes futuros que possam surgir. Não importa o quanto podemos apresentar numa única página, mas sim que esse conteúdo seja bem visível e de fácil interação.

A aplicação *PratosLimpos* segue esta filosofia, pois só assim seria possível implementar uma aplicação que entrega ao utilizador a liberdade de escolha do dispositivo que pretende usar, pois eu como programador não sei se a aplicação será executada num computador pessoal ou num telemóvel onde o ecrã é numa escala reduzida, mas sei que o utilizador será fiel à aplicação se esta fornecer os conteúdos claros, de fácil acesso e interativos.

Depois de ter apresentado o meu ponto de vista para a implementação usada nesta aplicação, passo a explicar a prática aqui usada.

A fórmula usada para tornar os conteúdos o mais flexíveis possível foi a seguinte:

- *Layout* flexível
- Imagens flexíveis
- Uso de *Media Queries*, módulo das especificações CSS3

6.2.1. ***LAYOUT* FLEXÍVEL**

Ao implementar um *website*, o *layout* é uma das primeiras tarefas a realizar e normalmente recorre-se a vários auxílios de desenho para o concluir (desenhar no papel -> *software* de edição -> implementar estilos). Por vezes é um dos problemas em que se demora menos tempo, pois ''estaticamente'' falando, o *pixel* é uma medida fácil de se aplicar e alterar de acordo com o desejado.

Mas neste caso os termos em que falamos são relativos e adaptáveis ao ambiente onde se encontram a ser executados, para isso já não se pode falar tanto em *pixel* mas sim noutras medidas como a **percentagem** ou o **'em'**. Estas medidas relativas são muito habituais em estilos flexíveis e ambas muito parecidas, onde a medida **'em'** é uma medida conhecida da tipografia, representando '1em' a altura da letra 'M' no tamanho padrão da fonte. Em CSS o **'em'** representa o tamanho padrão da fonte do utilizador ou o tamanho da fonte do elemento ''pai'', se disponível. Assim se utilizarmos esta medida no tamanho das fontes do *website*, estas irão se redimensionar de acordo com o tamanho padrão da fonte definida no ***browser*** que as executa. A percentagem **(%)** é parecida à medida **'em',** no que toca ao tamanho da fonte ou à largura do sublinhado, onde **'100%'** representa **'1em'** e fora deste contexto o seu valor é sempre relacionado com o bloco ''pai'' desse elemento. (GuiStuff, 2007)

Bem-Vindo a *pratoslimpos* (h1{font-size: 26px})

Bem-Vindo a *pratoslimpos* (h2{font-size: 20px})

Bem-Vindo a *pratoslimpos* (h3{font-size: 16px})

Ao definir os estilos CSS do *website*, a tipografia é dos primeiros elementos a implementar, mas para relacionar estas novas medidas é necessário recorrer à seguinte fórmula:

$$\text{'objetivo'} \div \text{'contexto'} = \text{'resultado'}$$

, por outras palavras, se dividirmos o tamanho da fonte que desejamos pelo tamanho da fonte do seu elemento (contexto), o resultado expresso, é o tamanho relativo da fonte que pretendemos. Assumindo que o tamanho padrão da fonte num *browser* é 16pixels e que o elemento "*body*" da nossa aplicação tem como tamanho padrão da fonte **100%**, podemos então aplicar diretamente os valores na fórmula e obter os títulos em tamanhos relativos.

$$\text{ex.: } 20px \div 16px = 1.25em$$

, ou seja, 20px é 1.25 vezes maior que 16px o que quer dizer que o tamanho relativo da fonte é 1.25em.

A ***Figura 24*** mostra como foi implementado o elemento h2 e respetivos resultados em '**px**' e '**em**', aberto num *browser* em *fullscreen*.

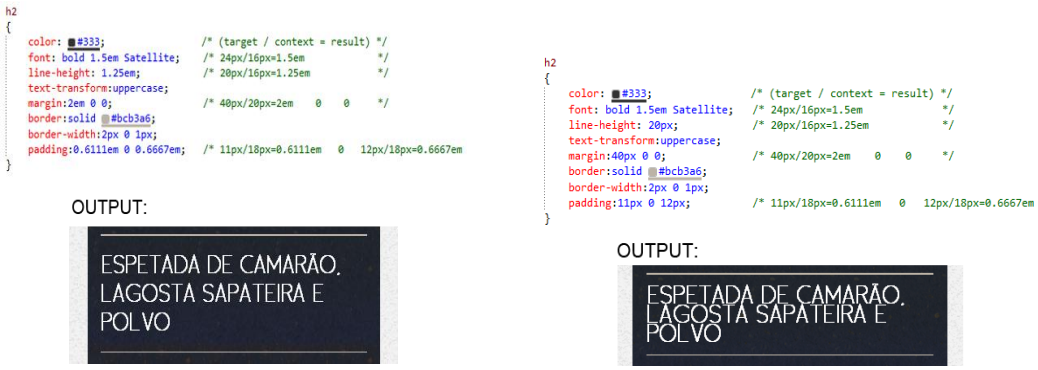


Figura 24: Tamanho da fonte em 'em' e 'px', respetivamente.

A partir deste ponto é possível definir os diferentes elementos de uma forma flexível e adaptável, mas tendo em conta que os valores definidos serão sempre influenciáveis para os "filhos" dos respetivos elementos, isto é, se quisermos aplicar um *link* a este elemento h2 com as mesmas características mas tamanho de letra inferior, o nosso contexto muda, pois agora não são os 16px que nos guiam mas sim o valor definido para h2,

$$\text{ex.: } 14px \div 20px = 0.7em$$

, ou seja, não dividimos o nosso objetivo por 16px definidos no "*body*" mas sim pelos 20px definidos no "*headline*" 2.

Assim, com esta fórmula é possível tornar as fontes da aplicação adaptáveis à janela do *browser*, do modo mais preciso possível, sem ser necessário muita implementação e cálculo.

Agora, a questão será implementar estas fontes num "*layout*" definido por uma grelha flexível, por isso as tabelas e colunas/linhas estáticas terão de ser postas de parte, por enquanto. A melhor maneira de criar uma grelha flexível é dar uso ás ** <\div>**, pois são fáceis de alterar e simples de implementar.

A aplicação *pratoslimpos* tem como principal *layout* a ***Figura 25***.

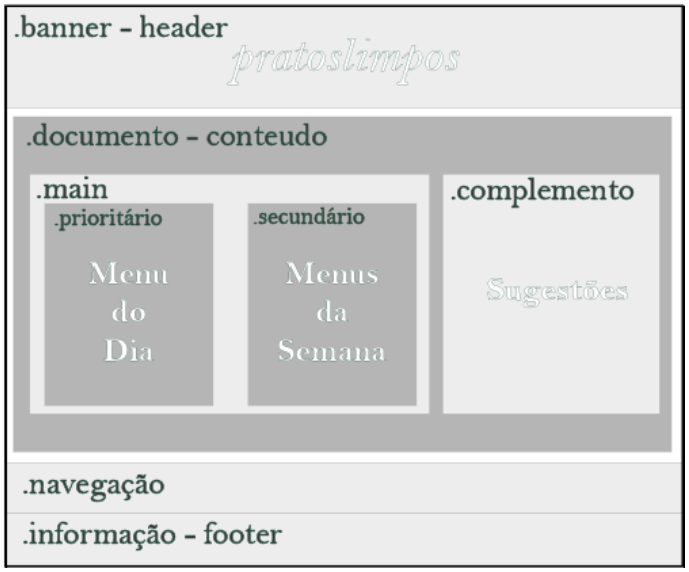


Figura 25: Layout: Pratos Limpos

```
<div class=''page''>
    <div class=''header''>
        <div class=''banner''>
            /*(...)*/
        </div>
    </div>
    <div class=''documento''>
        <div class=''main''>
            <div class=''prioritario''>
                /*(...)*/
            </div>
            <div class=''secundario''>
                /*(...)*/
            </div>
        </div>
        <div class=''complemento''>
            /*(...)*/
        </div>
    </div>
    <div class=''navegacao''>
        /*(...)*/
    </div>
    <div class=''informacao''>
        /*(...)*/
    </div>
</div>
```

Figura 26: Código HTML: *Layout* Pratos Limpos

O objetivo agora, é criar um modelo HTML que desenhe este *layout* recorrendo às ***div's*** e respetivos estilos ou classes, o que é simples, pois cada parte do *layout* irá corresponder a um bloco ** ** (ver**: Figura *26***). Utilizando uma hierarquia entre conteúdos pode-se definir em concreto o tipo de *layout* pretendido sem recorrer a tabelas, onde a página principal irá ser definida por um "*.header*", com um "*.banner*" associado, um "*.documento"* com dois sub-conteúdos, "*.main*" e "*.complemento*", onde o "*.main*" irá obter mais dois sub-conteúdos, o "*.prioritário*" que se irá sobrepor sempre em relação aos restantes módulos e ao "*.secundário*", por fim o "*.navegação*" e o "*.informação*" serão conteúdos do tipo "*footer*", pois o seu posicionamento será sempre no fim de cada página. Especial atenção para a navegação pois esta irá ter um papel importante no "*layout*" flexível, pois em janelas superiores a 1024px de comprimento, encontra-se no topo da página e em janelas inferiores a esta medida encontra-se no fim da página sobre o "*footer"*.

Agora cada ** ** irá assumir valores relativos para que se possa moldar de acordo com o tamanho da janela onde está a ser executado. Do ponto de vista teórico pode parecer simples mas quando posto em prática, terão de ser aplicados alguns truques. Assumindo que os estilos CSS da tipografia, os *backgrounds* e todos os elementos não associados ao *layout* já se encontrem implementados, vou dar agora mais importância em como tornar as ** ** em módulos definidos por colunas e linhas, pois neste momento não passam de ''linhas dentro de linhas''.

Antes de mais é importante usar vários estilos para implementar um *website*, principalmente quando este contém características flexíveis. Neste caso os estilos básicos ("*base.css*") da tipografia irão estar separados de dois outros documentos CSS. Um que é o próprio *layout* ("*layout.css*") e outro que serão os padrões ("*padroes.css"*) onde contêm os itens usados particularmente em certos conteúdos.

Passando novamente ao *layout* e mais concretamente à página ("*.page*"), esta será definida com um ***width: 100%***, pois eu pretendo que a página ocupe, relativamente, sempre toda a janela do "*browser*". Passando ao módulo "*.main*", este já não toma toda a página mas sim uma parte dela e tem de se adaptar às diferentes dimensões. Então considerei que o tamanho horizontal do módulo "*.main*" será aproximadamente 800px (***width: 800px***), mas este valor está expresso em pixels e como foi dito antes, o pixel não é "amigo" da flexibilidade, então recorrendo à mesma fórmula já usada anteriormente pela tipografia (**'objetivo' ÷ 'contexto' = 'resultado'**) e assumindo que o contexto é aproximadamente 1060px, valor definido para o documento ("*.documento*"), obtém-se então a medida relativa do que é pretendido.

$$800px \div 1060px = 0{,}754716981em$$

Mas este valor aproximado (0,754716981) ainda não indica um valor muito concreto, por isso considerando que o pretendido é expresso em percentagem, basta multiplicar o resultado por 100.

$$100 \text{ X } 0{,}754716981 \simeq 75{,}47\%$$

Agora já se pode utilizar este valor, que é relativo, ao tamanho do módulo "*.main"* ficando da seguinte maneira,

Estilo *.main* da folha de estilos *base.CSS*:

```
.main{
    width:75.47%;
    float: left;
}
```

Agora o conteúdo do módulo "*.main*" irá corresponder a **75.47%** do conteúdo de "*.documento"* e será alinhado à esquerda (**float: left**;) para corresponder ao que foi delineado no *layout* anteriormente. Novamente é importante salientar que o nosso contexto sofre constantes mudanças e é preciso ter isso em conta, p.ex., se agora implementar o módulo "*.secundario",* o contexto corresponde ao defino em "*.main*", pois é o módulo "pai" deste elemento, assim ao efetuar o cálculo não se divide o objetivo por 1060px mas sim por 800px,

$$240\text{px} \div 800\text{px} = 0{,}3 \rightarrow 0{,}3 \text{ X } 100 = 30\%$$

Assim o conteúdo de "*.secundario*" irá corresponder a **30%** do conteúdo de *".main*", o restante será atribuído a "*.prioritario*" e respetivos *paddings*.

Estilo *.main.secundario* da folha de estilos *base.CSS*:

```
.main.secundario{
    width: 30%;
    float: right;
    padding: 0.25em 0 0;  /*(4px 0 0)*/
}
```

Definindo os valores estáticos para os respetivos valores relativos, usando unicamente uma simples fórmula, obtém-se um "*layout*" puramente flexível e capaz de se adaptar ao ambiente onde se encontra a ser executado, simples e de fácil compreensão. É importante dar atenção aos valores relativos dos espaços entre conteúdos ou *paddings* pois estes também são proporcionais a todo o ambiente.

Apenas demonstrei uma parte dos estilos usados em *PratosLimpos,* pois as restantes atribuições seguem o mesmo esquema de ideias e como disse esta matéria é muito extensa e iria retirar tempo precioso para poder explicar os outros tópicos que também são relevantes para este projeto.

6.2.2. IMAGENS FLEXÍVEIS

Em HTML colocar uma imagem em um determinado elemento é extremamente simples pois basta definir um bloco ** ** e colocar no interior a imagem desejada, neste caso irei exemplificar uma imagem que por sua vez também servirá de *link* dentro de um módulo definido pela classe figura:

Declarar imagem tipo *link:*

```
<div class="figura">
    <a href="MenuDoDia.aspx">
        <img src="menuDia.png" alt=""/>
    </a>
</div>
```

, assim facilmente teremos a imagem dentro de um elemento do *layout*, mas este elemento está definido com parâmetros relativos e a imagem com dimensões estáticas, então para que a imagem não ultrapasse as dimensões do elemento figura, basta atribuir a seguinte característica ao estilo CSS da imagem:

Estilo "*.img*" da folha de estilos "*padroes.CSS*":

```
.img{
    max-width: 100%;
}
```

, logo a imagem irá corresponder ao tamanho máximo do elemento "figura" e irá se "auto-redimensionar" para as dimensões do mesmo elemento. Fácil e rápido de implementar, mas como tudo, não é assim tão linear, pois esta solução é executada na perfeição nos novos *web browsers* mas não nos antigos, como é o caso das versões anteriores ao Internet Explorer 6. Para estes, o parâmetro *max-width* não é reconhecível, mas o simples parâmetro *width* ainda o é.

Uma solução seria implementar um estilo em que se fosse executado em *browsers* modernos utilizaria o parâmetro *max-width* e em *browsers* antigos o parâmetro *width*, isto é possível recorrendo a algum código *javascript*, mas de momento não me vou prenunciar nesta matéria e vou cingir-me ao que é possível nos tempos modernos, por o menos por enquanto.

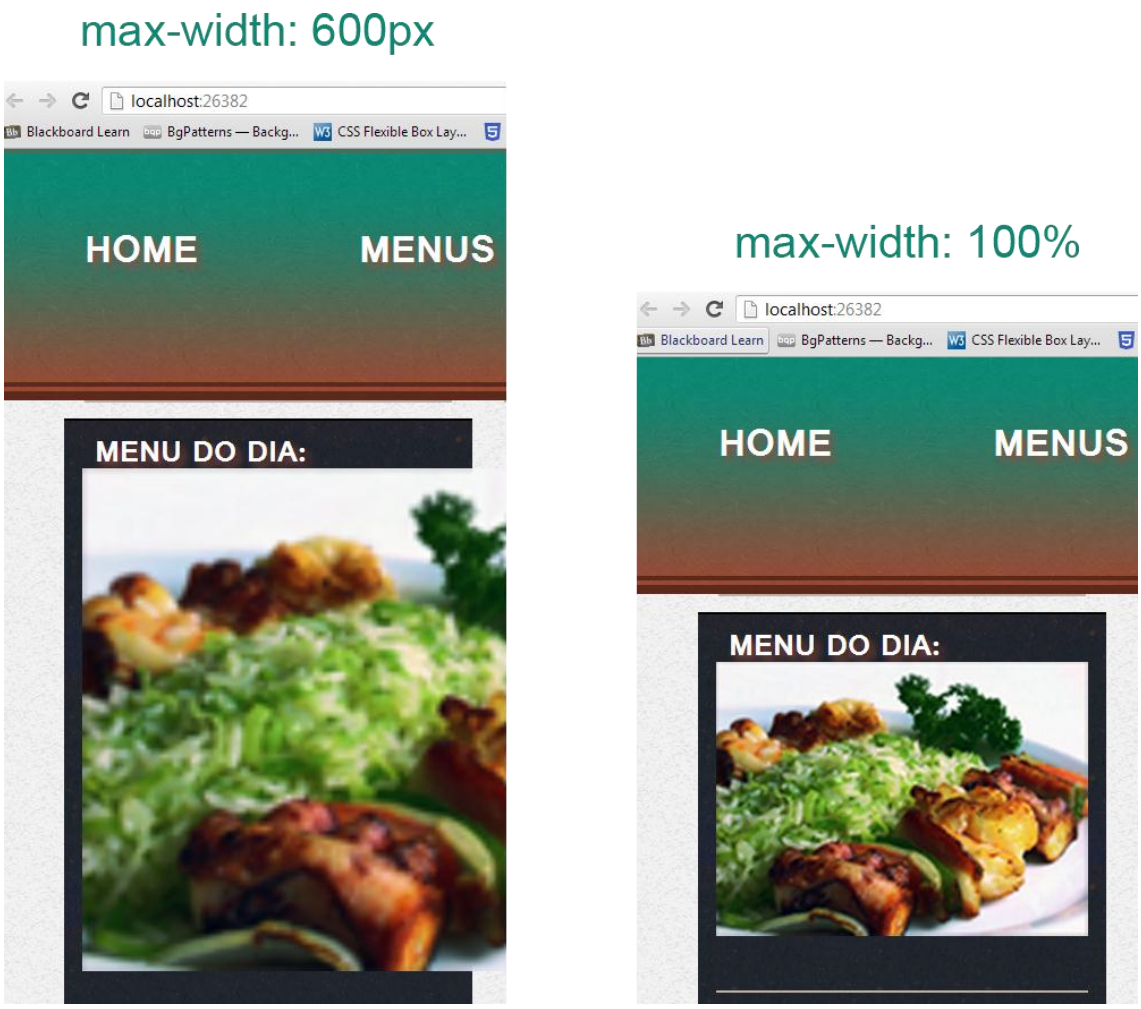


Figura 27: Imagem Flexível, Pratos Limpos

De momento já é possível fazer uma imagem flexível sem grandes alterações nos estilos CSS da mesma, esta técnica também foi usada para quase todas as imagens de maneira a elas serem o mais flexíveis possível. Mas o fundo e os *icons* foram implementados de outra maneira, pois também são eles imagens mas de propósito diferente. Em relação ao fundo achei por bem utilizar imagens tipo padrão e aplica-las com uma técnica já antiga mas muito usual até porque assenta bem num *layout* flexível. Não é mais nem menos, que outra propriedade do HTML, neste caso o parâmetro *background*, que permite repetir vezes sem conta o mesmo objeto dando assim a ilusão que é uma única imagem, pois não passa de um padrão repetido várias vezes num determinado sentido. Sendo mais concreto, como já é possível visualizar na última imagem apresentada, estes padrões foram usados no conteúdo "*.pagina*" e "*.banner*", onde para o *banner* é usada uma imagem de dimensão 224px/224px e para o *background* da página, uma imagem de 100px/100px, repetidas de maneira a preencher todo o conteúdo.

A técnica usa-se da seguinte maneira, se pretendermos que o padrão se repita num só sentido é necessário cinco características no parâmetro background:

- 1. A cor do *background,* alternativa à imagem, se possível de cor semelhante aos tons da mesma;
- 2. O caminho físico (url) da imagem ou ficheiro;
- 3. Para onde o *background* se irá repetir;
- 4. A posição do *background* na página;
- 5. A posição inicial do *background.*

Em relação ao "*.banner",* o *background* irá se repetir em relação à coordenada x e a posição é no topo do conteúdo onde está inserido, inicialmente estará alinhada à esquerda desse mesmo módulo.

Estilo "*.img"* da folha de estilos "*padroes.CSS*":

```
.banner{
    background: #396246 url('bgd\_Banner.png') repeat-x top left;
}
```

Como o fundo da página se repete até ao fim da mesma, então basta simplesmente colocar a característica *repeat* a seguir ao *url* da imagem para que o *background* se repita pelo conteúdo todo.

Já demonstrei as técnicas utilizadas nas imagens que necessitam de melhor definição, as imagens tipo padrão para o fundo e agora só faltam os *icons,* que têm muita definição mas pouco ''peso'' para a aplicação. Um bom método é o uso das imagens tipo SVG ("*Scalable Vector Graphics*'') que entraram em desuso, mas que agora com o aparecimento de uma filosofia mais flexível e que ao mesmo tempo pretende ser o mais amiga do utilizador, o uso deste tipo de ficheiros oferece uma técnica independente da resolução para apresentar imagens na *web*. É um formato de gráficos vetoriais usados em XML e que define propriedades complexas para desenhar toda uma imagem vetorial. Não me vou pronunciar muito mais neste tema pois é possível criar uma imagem SVG rapidamente e personalizada recorrendo ao uso da ferramenta *Adobe: Illustrator* que é própria para desenhar imagens vetoriais (ver: **Figura 28**).

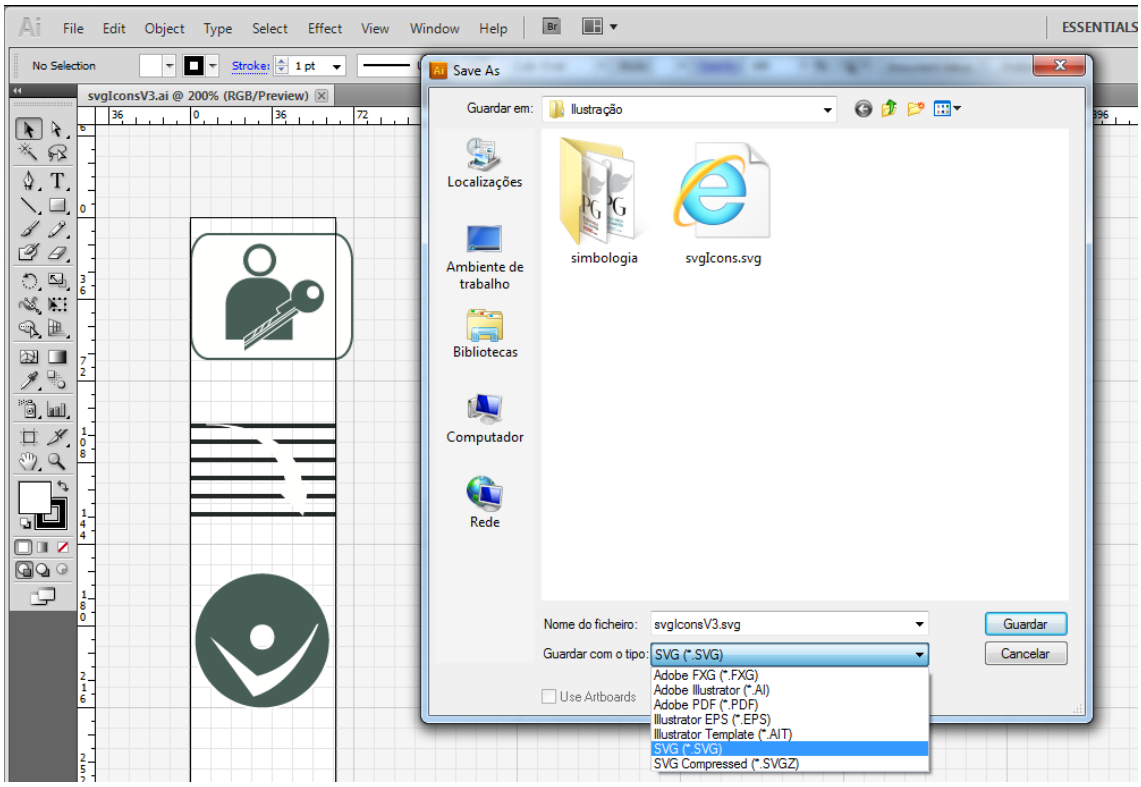


Figura 28: Illustrator: Como criar imagens SVG, Pratos Limpos

Após incluir este ficheiro "*.svg*" no projeto e respetiva imagem em formato "*.png*", tem de se alterar a folha de estilos CSS ("*Layout.css*") e definir os parâmetros para obter as imagens vetoriais "*.svg*" da seguinte maneira:

Definir imagens *SVG*:

```
.icons\_Menu a.skipIcons {
  background:url('../Imagens/svgIcons.png') no-repeat 0 0;
  width:3.6em;
  height:3.2em;
  float:right;
  }
  .svg .icons\_Menu a.skipIcons{
            background-image:url('../Imagens/svgIconsV3.svg');
}
```

Como já foi referido pode-se facilmente manipular os *backgrounds* dos elementos. Neste caso os parâmetros definidos apontam para a primeira figura da imagem *"svgIcons.png"*, mas alterando o ultimo parâmetro de 0 para um valor negativo aproximado, obtemos a segunda figura e assim sucessivamente. Este exemplo foi implementado para fazer um botão de *login* para o utilizador:

Definir botão *icon SVG*:

```
<div class=''icon\_Menu''>
  <a class="skipIcons" href="Login.aspx"></a>
</div>
```

Na **Figura 29** o elemento sublinhado a vermelho corresponde à navegação que está posicionada com o último parâmetro do *background* a -4 e o elemento sublinhado a amarelo corresponde ao botão de *login* que é representado pelo código acima descrito.

Figura 29: Icons: Navegação e Login, Pratos Limpos

Até agora as imagens foram tratadas de forma livre e sem dar grande importância à forma em como são descarregadas pela Internet, mas como se sabe em conteúdos *web* o tamanho de uma imagem é muito importante e normalmente se adequa recorrendo a programas de edição para as tornar com as dimensões desejadas e com um "peso" o mais diminuto possível. Mas isto é muito relativo pois não estamos a falar de um *website* que é só visualizado em computadores pessoais que normalmente têm acesso à Internet com uma boa largura de banda, estamos também a falar de dispositivos de várias dimensões que têm ecrãs não muito grandes e que estão constantemente sujeitos a oscilações nas ligações à rede. Assim sendo, as imagens definidas para computadores com ecrãs de dimensões consideravelmente grandes e um acesso bom à rede, seria um enorme problema para os dispositivos móveis e seus requisitos muito abaixo do aconselhável para este tipo de situações.

P.ex.: A imagem usada no menu do dia da página principal da aplicação *pratoslimpos* tem uma dimensão de 500/300px, com tamanho aproximado de 300KB, não se pode dizer que é um ''peso'' elevado para uma rede normal, mas agora vejamos se esta imagem for aberta num dispositivo que está sujeito a uma rede de velocidade inconstante (ex.: 3G) e que por vezes não passa os 100KB/seg, juntando todo o conteúdo da mesma página iria sobrecarregar demasiado a rede e o próprio dispositivo.

Com alguma pesquisa encontrei várias soluções para este problema, mas nenhuma "*100%*" eficaz, pois este é um problema que ainda nenhum *browser* é capaz de resolver e seria uma enorme ajuda se existe-se algo para enviar dados com o tamanho ideal, respeitando os requisitos dos dispositivos. A própria comunidade W3C já se encontra de volta de uma solução para o HTML que resolva esta situação, mas por enquanto irei-me cingir às soluções de alguns *web designers*.

Muitas envolvem o dito *JavaScript* e outras através do servidor para detetar a versão do *browser* que está a carregar os conteúdos.

- ***Responsive Enhance***: Josh Emerson foi o primeiro a abordar esta técnica. Basicamente é efetuada uma referência para uma imagem de tamanho pequeno e que será **sempre** descarregada, só depois é que será reposta por uma de maior dimensão, se for o caso de existir uma referência para tal. Para quem estiver familiarizado com a história do HTML é muito parecido com o antigo atributo *lowsrc*, que foi extinto da linguagem.
- ***Adaptive Images***: Matt Wilcox propôs uma solução para o PHP que envolve o carregamento de várias imagens e de várias dimensões consoante o tamanho do ecrã que as executa.
- ***Responsive Images***: Por fim, Scott Jehl combinou o *JavaScript*, com os *cookies* e o lado do servidor, para conseguir detetar o tamanho da *viewport* antes de qualquer imagem ser descarregada. Se a *viewport* for pequena então a imagem de menor tamanho será descarregada, se a *viewport* for grande o suficiente, as imagens de grande dimensão serão requisitadas. Mas nem sempre grandes ecrãs suportam grandes imagens.

Das três soluções acima descritas todas têm vantagens e desvantagens para este problema e segui aquela que achei mais fácil de implementar e que também acaba por ser a que se apropria mais a este projeto. Basicamente a aplicação irá ter poucas imagens que requerem grandes resoluções, por página não passará de uma ou duas imagens, pois as restantes são de pequenas dimensões e por isso podem muito bem ser carregadas com um tamanho pequeno, sejam visualizadas em grandes ou pequenas "janelas".

A proposta de Josh Emerson, ***Responsive Enhance*** é uma técnica que encaixa na perfeição nestes requisitos e ele próprio disponibilizou os conteúdos necessários para implementar este método:

https://github.com/joshje/Responsive-Enhance

O ficheiro mais importante é o "*resonsive-enchance.js*", que terá de ser incluído no projeto.

Implementar *Responsive-Enchance*:

- 1º É necessário incluir o ficheiro *"resonsive-enchance.js"* no elemento **<head>** da página que irá usar esta técnica:

  Incluir "*responsive-enhance.js*":

  ```
  <head>
      <script src="../JavaScript/responsive-enhance.js"></script>
  </head>
  ```

- 2º O elemento **<img>** terá de ter o atributo "*data-fullsrc*", para dar referência à imagem de grande dimensão:

  Definir <img> para receber duas imagens:

  ```
  <img class="media-object" id="media-object" src="../Imagens/Menus/MenuDia\_Small.png" data-fullsrc="../Imagens/Menus/MenuDia\_Large.png" alt="Menu do Dia"/>
  ```

- 3º É necessário a seguinte linha de código *JavaScript* a seguir ao elemento **<img>**:

  Codigo javascript relativo ao "*responsiveEnhance*":

  ```
  <script>responsiveEnhance(document.getElementById('media-object'), 320);</script>
  ```

  Nota: Esta linha de código indica ao *browser* qual a imagem que deve repor com base no tamanho definido (320), caso o *Javascript* esteja desativado o utilizador irá receber a imagem de menor dimensão pois é aquela que está definida em *src*.

- 4º Para finalizar, é necessário alterar alguns atributos na folha de estilos CSS correspondentes ao objeto da imagem para que esta tome sempre o tamanho do elemento onde está inserida, independentemente de ser a de pequena ou grande dimensão:

  Definir estilo de *img.media-object:*

  ```
  img.media-object{
    margin:0 auto;
    width:100%;
    max-width:50em;   /*800px/16px = 50em*/
  }
  ```

Como foi referido acima esta solução não é perfeita, mas assenta bem quando utilizada só para uma imagem, pois esta técnica não permite mais imagens por página, mas sempre é melhor que dispor imagens de grandes dimensões.

Na **Figura 30** nota-se alguma perda de qualidade quando o *JavaScript* é desativado:

Figura 30: Aplicação prática da técnica *Responsive Enchance*, Pratos Limpos

6.2.3. MEDIA QUERIES

Por fim para conseguir com que o *layout* seja o mais "*responsive*" possível, ou seja, adaptável a qualquer dispositivo, seja ele, computador portátil, *tablet* ou *smartphone,* o que importa é que todo o conteúdo exposto ao utilizador esteja visível, fácil de interagir, ler e que anule dois grandes problemas:

- *Scroll* horizontal
- *Zoom* manual

Principalmente quando se fala de dispositivos móveis que têm pequenas dimensões e nem sempre o conteúdo é visto de forma clara.

Até agora foi definido um *layout* flexível e que se acomoda automaticamente à "janela", por o menos por enquanto o problema do *scroll* horizontal foi eliminado, mas ainda falta resolver a questão do zoom e é aqui que as *media queries* entram. Antes de falar nesta solução irei demonstrar como o *website* se irá moldar seguindo o primeiro *layout* que foi apresentado (ver: **Figura 31**).

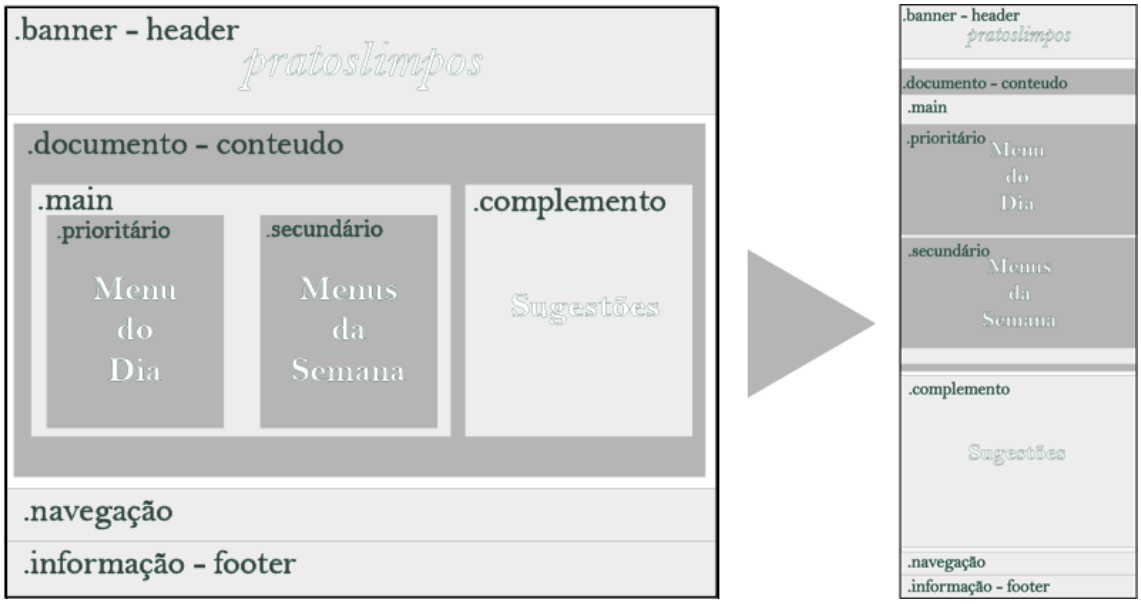


Figura 31: Comportamento do *layout* em diferentes dimensões, Pratos Limpos

Após refletir sobre as modificações que vão ser feitas ao *layout* devido às mudanças constantes da *viewport,* pode-se partir para os tipos de *media* que puderam ser utilizados para definir se a página é do tipo *print, screen, projection,* entre outros, o que importa é que estes tipos de *media* são ótimos para desenhar um *website* mais compatível com os diferentes tipos de *browsers* ou dispositivos.

> *''Using the viewport tag allows us to control the size of the canvas, and override the default behavior: we can dictate how wide the browser's viewport should be.''*
>
> (Marcotte, 2011)

Antes de mais é preciso definir a *viewport* da aplicação no principio *(<head>)* do bloco de HTML da seguinte maneira:

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

A propriedade *initial-scale=1* basicamente coloca o *zoom* da página a 1.0 ou 100% de modo a garantir alguma consistência em ecrãs pequenos, a propriedade *content = "width= device-width"* indica que o comprimento da *viewport* do *browser* irá ser igual ao comprimento do ecrã do dispositivo, o que é uma grande ajuda pois o *browser* não irá ter mais o tamanho predefinido mas sim corresponder exatamente ao tamanho do ecrã do dispositivo que o executa.

Agora, recorrendo novamente à folha de estilos CSS, esta agora tomará pequenos blocos de código do tipo **@media** para efetuar as diferentes alterações do conteúdo ao *layout*. As *media queries* são compostas por um tipo, neste caso *"screen"* e a respetiva *query* entre parênteses. Se definirmos uma *query* da seguinte maneira, **@media screen (min-width: 1024px),** com um conjunto de instruções dentro deste bloco, só serão executadas se o tipo de *media* for "*screen*" e caso seja, verifica se o comprimento mínimo da *viewport* é por o menos 1024px. Assim facilmente se obriga o *layout* a responder às medidas onde está a ser executado.

O browser "Mobile Safari" tem como predefinição 980px de área mesmo quando o dispositivo que o executa tem um ecrã de 320px, apenas se acomoda ao ecrã efetuando "zoom-in".

(Marcotte, 2011)

Figura 32: Predefinição do Browser num iPhone

A **Figura 33** mostra uma simples *media query* implementada no *site pratoslimpos*, que irá efetuar alterações nos elementos do "*.banner*", caso este seja executado num tipo *screen* e com uma *viewport* de comprimento mínimo 55em (aproximadamente 1024px) e de altura mínima de 40em (aproximadamente 768px), esta é a definição do *layout* para monitores de grandes dimensões.

```
@media screen and (min-width:55em) and (min-height:40em) {
    .banner {
        border-width:4px 0 8px;
    }
    .banner .masthead {
        border-bottom-width:4px;
        margin-bottom:4px;
    }
    .banner h1 {
        margin:0 auto;
        width:21em;
        float:none;
    }
    .banner h1 a {
        background:url('../Imagens/Logo_Big.png') no-repeat;
        width:21em;
        height:11em;
    }
}
@media screen
and (min-width:55em)
and (min-height:40em) {
    .banner a.skip {
        display:none;
    }
}
```

Figura 33: Aplicação prática das *media queries*, Pratos Limpos

Sem as *media queries* toda esta ideia de "*Responsive Web Design*" não faria sentido, elas permitem servir condicionalmente as CSS a comandar as capacidades dos dispositivos, o que os leva a obedecer às características do *website*. Assim os utilizadores já não têm que se preocupar com o ambiente da página e sim cingir-se a explora-la de uma maneira mais rápida e não tanto cansativa.

Este tema daria para um só único projeto mas o que aqui está em causa é a integração desta nova ideia que ainda se encontra em estudo e a linguagem ASP.NET de forma a implementar uma só aplicação única, completa e flexível às necessidades do utilizador. O que me leva a terminar este tema, por enquanto, dei uma breve explicação em como facilmente se torna um *website responsive*, mais amigo do utilizador e adaptável ao futuro da *web.*

6.3. INTEGRAÇÃO DO ASP.NET

Após concluir a primeira parte da implementação que por sua vez também é a mais extensa em termos de matéria, irei explicar agora como é possível passar de um simples *website* estruturado em HTML e integra-lo na plataforma ASP.NET, que também ela é própria para desenvolver aplicações *Web*.

O porquê de ter escolhido esta plataforma, é que para além de ter obtido conhecimentos através das aulas lecionadas na cadeira de Programação para a Internet, também ganhei algum interesse e conforto ao implementar nesta plataforma. A plataforma .NET suporta a linguagem **C#** que acaba por ser uma mais-valia, pois abrange o poder e versatilidade do Visual Basic, a força e a criatividade do C++ e a inteligência do *Javascript*. Para mim o uso de C# só me facilitou em termos de trabalho pois é muito parecido com a linguagem *Java,* que me identifico muito.

A integração do HTML é extremamente simples pois a própria tecnologia .Net suporta esta mesma linguagem, que incluindo C#, é possível obter um único bloco multifacetado.

Após criar um projeto ASP.NET recorrendo à ferramenta *Visual Studio*, uma das primeiras tarefas a fazer é definir a *MasterPage,* que é basicamente a página responsável pelos conteúdos comuns em todas as páginas da mesma aplicação. Neste caso é utilizada para definir todo o *layout* da aplicação *pratoslimpos* de modo a que em todas as páginas criadas, este seja flexível como foi mencionado anteriormente.

Como em HTML a *masterpage* também será definida recorrendo à mesma linguagem (ver: **Figura 34**).

Percorrendo o código mencionado na **Figura 34** sabe-se que, o cabeçalho da *masterpage* tem sempre de começar por referir qual a linguagem que irá incorporar dentro da página ASP.NET e a classe que se encontra separada, mas que pertence ao mesmo arquivo (forma de separar o código HTML das classes em C). Agora tudo o que será implementado estará dentro de um bloco HTML. Dentro da *tag* **<head>** definem-se todos os métodos principais, como a *viewport*, o código "*javascript*", as várias folhas de estilo utilizadas (neste caso: *base, layout e patterns*) e por fim o ficheiro "*responsive-enchance.js",* ou outros ficheiros de suplemento à aplicação. A *tag* **<body>** irá conter uma *form* que indica que todo o seu conteúdo irá ser processado pelo servidor (***runat=''server''***).

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Site.master.cs" Inherits="Pratos_Limpos.SiteMaster" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head runat="server">
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <!--detect js+svg support -->
    <script>
        document.getElementsByTagName('html')[0].className += ' js';

        (function flagSVG() {
            var ns = { 'svg': 'http://www.w3.org/TR/SVG11/' };
            if (!!document.createElementNS && !!document.createElementNS(ns.svg, 'svg').createSVGRect) {
                document.getElementsByTagName('html')[0].className += ' svg';
            }
        })();
    </script>
    <title></title>
    <!--css-->
    <link rel="stylesheet" href="Styles/base.css" type="text/css" media="all"/>

    <link rel="stylesheet" href="Styles/layout.css" type="text/css" media="all"/>

    <link rel="stylesheet" href="Styles/patterns.css" type="text/css" media="all"/>

    <asp:ContentPlaceHolder ID="HeadContent" runat="server">
    <script src="~/JavaScript/responsive-enhance.js" type="text/javascript"></script>
    </asp:ContentPlaceHolder>
</head>
<body>
    <form runat="server">
        <div class="page">
            <div class="header">
                <div role="banner" class="banner">
                    <div class="masthead">
                        <h1><a href="/" rel="home"></a></h1>
                        <a class="skip" href="#navigation"></a>
                    </div>
                </div><!--/.banner-->
                <div role="icons_Menu" class="icons_Menu">
                    <a class="skipIcons" href="Login.aspx"></a>

                </div>
            </div><!--/.header-->
```

Figura 34: Definição da *masterpage*, Pratos Limpos

Agora basta definir o "corpo" em HTML para tornar o *website* flexível, com uma única particularidade, a *masterpage* também tem a sua própria forma de processar os conteúdos das páginas (**ContentPlaceHolder**), que passa por definir cada área do *layout*, (ver: **Figura 35**).

```
<div role="document" class="document">
    <div role="main" class="main">

        <div class="prioritario">
            <asp:ContentPlaceHolder ID="ConteudoPrioritario" runat="server">
            </asp:ContentPlaceHolder>
        </div><!--/.prioritario-->

        <div class="secundario">
            <asp:ContentPlaceHolder ID="ConteudoSecundario" runat="server">
            </asp:ContentPlaceHolder>
        </div><!--/.secundario-->

    </div><!--/.main-->

    <div class="complemento">
            <asp:ContentPlaceHolder ID="ConteudoComplemento" runat="server">
            </asp:ContentPlaceHolder>
    </div><!--/.complemento-->
</div><!--/.document-->
```

Figura 35: Interligação dos blocos em HTML com os ContentPlaceHolder, Pratos Limpos

Definida a *masterpage, agora* é mais fácil implementar cada página (.aspx) individualmente sem ter que redefinir todo o *layout* novamente, para isso basta incluir o seguinte cabeçalho em cada página:

<\%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Pratos_Limpos._Default" \%>

Neste caso a página "*Default.aspx*" irá utilizar o ficheiro "*Site.master*" como sua *masterpage,* em todas as outras páginas basta alterar os atributos "*CodeBehind"* e "*Inherits"* para os respetivos nomes.

Agora basta colocar devidamente cada conteúdo dentro dos *ContentPlaceHolders* definidos com um ID individual e um ID correspondente ao *ContentPlaceHolder* da *masterpage* (ver: **Figura 36**}.

```
<asp:Content ID="ConteudoPrioritario" runat="server" ContentPlaceHolderID="MainContent">
    <div>
    <figure class="media-frame">
        <a id="btoMenuDia" runat="server">
                <h7>Menu do dia:</h7>
                <img class="media-object" id="media-object" src="../Imagens/Menus/MenuDia_Small.png"
                    data-fullsrc="../Imagens/Menus/MenuDia_Large.png" alt="Menu do Dia"/>
                <h2 style="color: #ffffff"> <asp:Label ID="LabelNomeMDia" runat="server"></asp:Label> </h2>
                <h4 style="color: #ffffff"> <asp:Label ID="LabelAcompMDia" runat="server"></asp:Label> </h4>
                <h1 style="color: #ffffff"> <asp:Label ID="LabelPrecoMDia" runat="server"></asp:Label> €</h1>
            <asp:SqlDataSource ID="SqlDataSourceMenuDia" runat="server"
            ConnectionString="<%$ ConnectionStrings:pratoslimposConnectionString %>"
            ProviderName="<%$ ConnectionStrings:pratoslimposConnectionString.ProviderName %>"
            SelectCommand="SELECT menus.PrecoMenu, comida.Acompanhamento, descricaoprodutos.Nome, descricaoprodutos.Preco,
            descricaoprodutos.Descricao
            FROM menus INNER JOIN comida
            ON menus.id_Comida = comida.id_Comida
            INNER JOIN descricaoprodutos
            ON comida.id_Descricao = descricaoprodutos.id_descricaoProdutos"></asp:SqlDataSource>
        </a>

    </figure><!--/.media-frame-->
    <script>responsiveEnhance(document.getElementById('media-object'), 320);</script>
    </div>
```

Figura 36: Interligação do HTML com os objetos ASP.NET, Pratos Limpos

A **Figura 36** também já revela como interagir facilmente com o HTML e o ASP.NET, mais propriamente o que leva a inserir objetos do tipo *label* dentro de *tag's,* com as classes CSS já definidas anteriormente. Cada objeto irá sofrer as alterações pela *tag*, sem ser necessário estar a redefinir toda a folha de estilos. Em relação ao objeto *SqlDataSource* este irá influenciar cada conteúdo das *label's.*

Com esta breve explicação é possível integrar o código implementado em HTML na plataforma ASP.NET. Mas isto leva a algumas lacunas no projeto, pois grande parte dos objetos que esta plataforma fornece terão de ser utilizados de forma cautelosa, como é o caso das tabelas (ex.: *GridViews*) que não são nada flexíveis, as *textbox* que também elas têm de obter dimensões relativas e ao mesmo tempo fáceis de interagir ou ler os seus conteúdos, os botões que não convêm ter menos de 24px de área (área mínima recomendada para interagir com os dedos em *smartphones*), etc.. O que irá levar a um trabalho extenso para que a aplicação se torne 100% fiável. Algumas alternativas já foram implementadas, mas não irei cingir-me a muitos pormenores pois a aplicação ainda se encontra em desenvolvimento.

6.3.1. BASE DE DADOS

Vista a integração principal com a tecnologia ASP.NET, agora já é possível interagir de forma mais peculiar com certos objetos da plataforma, neste caso o *SqlDataSource*, que permite efetuar uma ligação com uma determinada base de dados de forma rápida e fácil de implementar. Como foi visto na **Figura 36**, é possível criar este objeto e facilmente escrever as *query's* necessárias para manipular os dados das tabelas já presentes na base de dados. Antes de falar deste objeto irei apresentar o sistema *MySql*.

O *MySql* é um sistema que gera bases de dados e utiliza como linguagem o SQL. É um dos mais utilizados por todo o mundo, não só por ser *Open Source*, mas também pela sua alta performance, segurança/confiança e fácil de usar.

Neste caso usei o MySql 5.5 e a ferramenta MySql WorkBench para manipular os primeiros dados e tabelas da base de dados de Pratos Limpos.

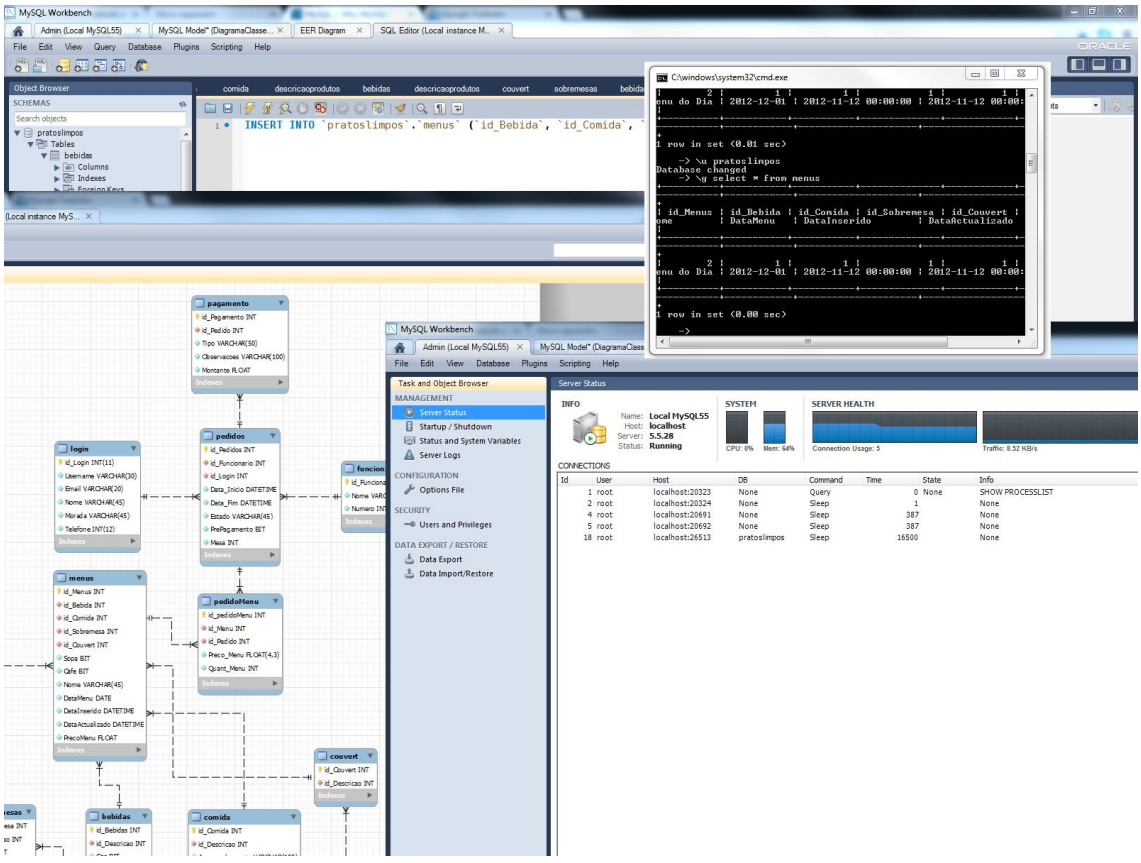


Figura 37: Ferramenta MySql Workbench

Integrar o MySql no projeto criado em Visual Studio é extremamente simples, pois existe uma ligação (*connector*) própria para a plataforma .Net, neste caso o MySQL Connector Net 6.5.4, que foi devidamente instalado e referido no projeto. Esta base de dados é remota e este *connector* utiliza a tecnologia ADO.Net para aceder aos seus dados (ver: **Figura 38**).

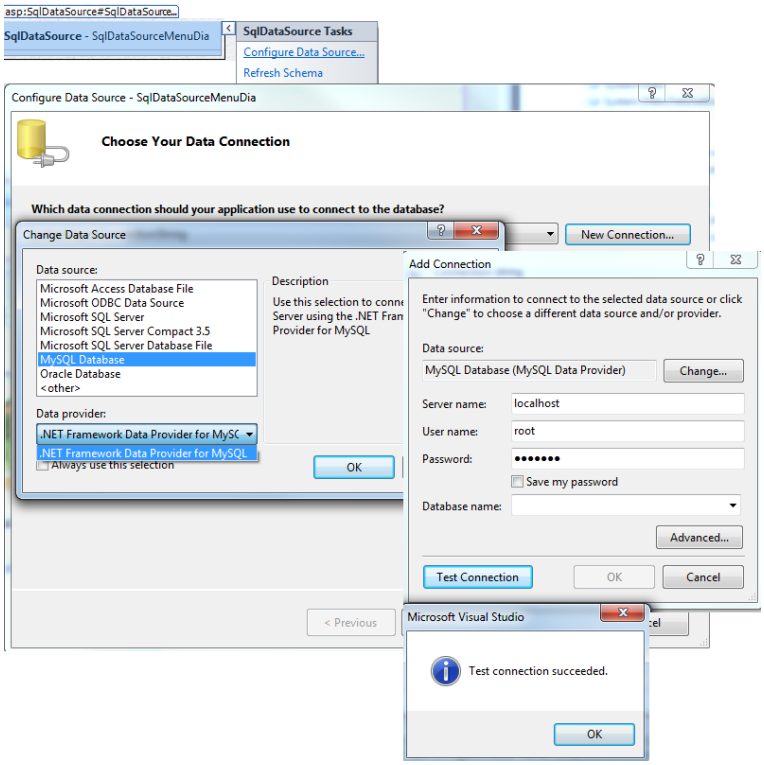


Figura 38: Ligação do SqlDataSource ao MySql, Pratos Limpos

A partir de agora é fácil ligar o objeto SqlDataSource ao MySql e construir as *query's* necessárias.

Claro que este processo não é assim tão linear, mas já é uma grande ajuda na manipulação e integração de dados na aplicação.

A seguinte situação mostra como na classe "*Default.aspx.cs*" foi implementada uma forma de aceder à base de dados através de uma **DataView,** que passa por utilizar *query's* to tipo "*Select*" e uma **DataRowView**, de modo a que cada *label* (ex.: Nome, Acompanhamento e Preço) tomem os valores da tabela "*Menus*". Esta operação é efetuada quando é carregada a página.

Outra operação também efetuada nesta mesma classe, é a passagem de parâmetros pela *url*. É um simples exemplo usando o método *Server.Transfer,* que permite passar os parâmetros para uma determinada página sem expôr os conteúdos *url,* pois é efetuado através do servidor. É de evitar esta situação pois é mais uma operação que o servidor irá executar, porém no subcapítulo seguinte irei apresentar uma alternativa recorrendo a uma característica do ASP.NET.

Neste penúltimo subcapítulo foi apresentada uma forma eficaz de a tecnologia ASP.NET interagir com uma base de dados remota MySql e, como outras matérias apresentadas também não foi muito aprofundada pois também ela é muito extensa.

6.3.2. RESTRITO E SEGURO

Uma das primeiras razões pelo qual escolhi utilizar a tecnologia ASP.NET foi a própria segurança e as restrições que facilmente se aplicam aos utilizadores. Esta plataforma fornece uma ferramenta capaz de gerir as contas dos utilizadores e restringi-los aos seus devidos conteúdos sem que as suas funções colidam. As configurações são guardadas num ficheiro XML denominado de "*Web.config*", que se encontra na raiz do projeto.

Neste caso terá que ser fornecida a ligação à base de dados MySql (*connectionString)* no ficheiro "*Web.config*":

```
<add name="MySQLConn" connectionString="Server=localhost; \\
Port=3306;Database=pratoslimpos;Uid=root;password=******;" />
```

Após efetuadas as configurações anteriores, já se pode entrar na página de configuração do ASP.NET:

Project → ASP.NET Configuration

Acedendo à página *security* pode-se definir os diferentes utilizadores da aplicação, as roles ou funções de cada utilizador e a que conteúdos podem aceder. Para dar permissões aos utilizadores é aconselhável definir pastas para cada tipo e incluir as suas páginas particulares (nota: Incluir um *web.config* a cada pasta para distinguir as *roles* de acesso a cada uma).

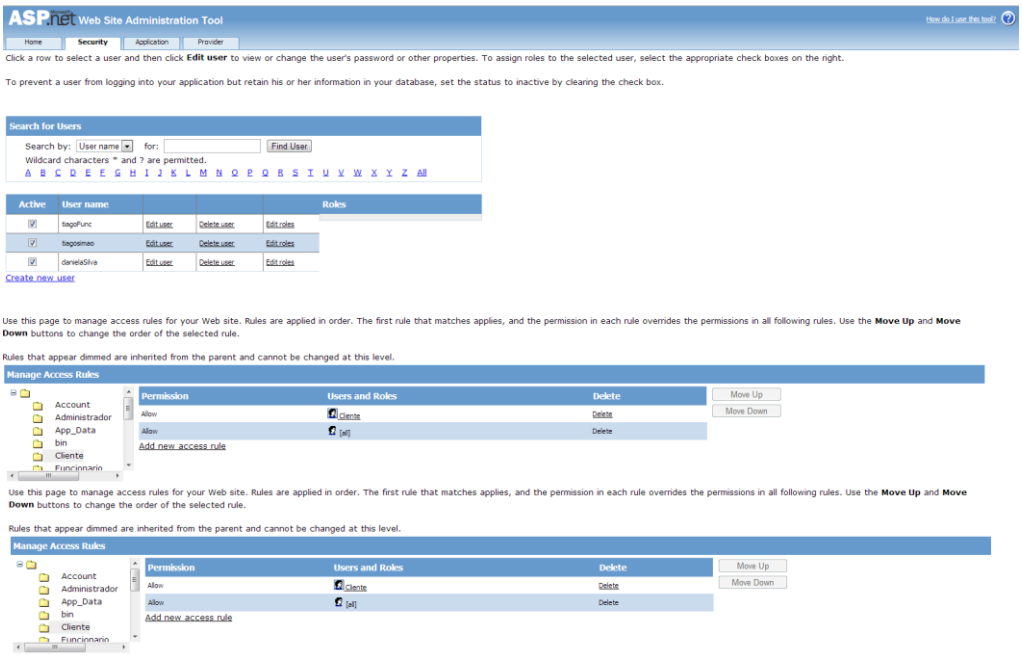


Figura 39: Várias opções da ferramenta ASP.NET WebSite Administration, Pratos Limpos

Com a segurança devidamente configurada pode-se agora criar os formulários devidos para registar e efetuar *login* dos utilizadores. Estes formulários têm de obedecer ao primeiro problema inicialmente falado, nesta situação particularmente, porque a interação do utilizador e da aplicação é crucial, pois este irá introduzir dados e ler ao mesmo tempo.

Por falar ficou o caso particular de passagem de parâmetros de uma página para outra. Ora como esta é uma aplicação que pertence ao mesmo projeto ou *website* aqui não se coloca o problema da portabilidade, logo pode-se usufruir da tecnologia ASP.NET para resolver esta questão. Facilmente se indica na página que irá obter os parâmetros partilhados, ou seja, como estão a ser usados objetos da mesma tecnologia e estes são públicos, assim qualquer página poderá obter a informação de outra.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (PreviousPage != null)
    {
        Control placeHolder = PreviousPage.Controls[0].FindControl("MainContent");
        Label SourceLabelNomeMDia = (Label)placeHolder.FindControl("LabelNomeMDia");
        Label SourceLabelAcompMDia = (Label)placeHolder.FindControl("LabelAcompMDia");
        Label SourceLabelPrecoMDia = (Label)placeHolder.FindControl("LabelPrecoMDia");
        if (SourceLabelNomeMDia != null && SourceLabelAcompMDia != null && SourceLabelPrecoMDia != null)
        {
            lbPedidoNMenu.Text = SourceLabelNomeMDia.Text;
            lbPedidoAMenu.Text = SourceLabelAcompMDia.Text;
            lbPedidoPMenu.Text = SourceLabelPrecoMDia.Text;
        }
    }
}
```

Figura 40: Passar parâmetros via página anterior, Pratos Limpos

6.4. FUNCIONALIDADES

A aplicação *PratosLimpos* ainda se encontra em desenvolvimento e como tal certas características permanecem incompletas. Mas após a implementação de todos os métodos acima mencionados, já se pode visualizar uma aplicação funcional e "flexível". De seguida, irei demonstrar como os utilizadores navegam na aplicação *PratosLimpos.*

6.4.1. COMO ACEDER A PRATOS LIMPOS

Uma particularidade nesta aplicação é que o cliente irá poder indicar a mesa através de um código QR específico, isto é, cada mesa terá o seu respetivo número e código QR, onde este contém encriptado o *link* para a aplicação, juntamente com o parâmetro "mesa = [*número*] ". Assim, o cliente pode aceder à aplicação sem digitar o *url* e sem se preocupar em que mesa se encontra.

O *link* "interno" (intranet) da aplicação será sempre do tipo:

***url* = http://[*ip servidor com a aplicação*]**

Se pretender colocar o número da mesa como parâmetro, basta adicionar ao *link* acima "?mesa=[*número*]". Este *link* também corresponde ao que se encontra nos códigos QR:

***url(com mesa)* = http://[*ip servidor com a aplicação*]?mesa=[*número*]**

Exemplo:

Figura 41: Acesso à aplicação Pratos Limpos via QRCode

Nota: Estes códigos QR foram criados através do *site* **http://goqr.me** e as aplicações de leitura testadas foram: o **QR Droid** para sistemas *Android* e o **QReader** da *Adobe* para qualquer sistema operativo. Este último tem a particularidade de funcionar nos computadores pessoais através da *webcam.* Estas aplicações são gratuitas e 100% funcionais.

6.4.2. VISTA GERAL DA *HOMEPAGE*

Vistas as possibilidades de acesso à aplicação, *PratosLimpos* inicialmente apresenta ao utilizador uma página de boas vindas juntamente com uma breve introdução dos principais conteúdos de toda a aplicação. Como é o caso dos menus do dia e da semana, os acessos aos vários tipos de utilizadores, as novidades, parceiros e algumas informações em como navegar. A **Figura 42** apresenta toda a interface apresentada ao utilizador, onde as imagens correspondem aos diferentes tipos de dispositivos ou diferentes tamanhos de "janelas":

- PC – Superior a 1024px;
- TABLET – Superior a 640px e inferior a 1024px;
- SMARTPHONE – Superior a 320px e inferior a 640px, onde valores inferiores a 320px, também será visível o *layout*, mas irá perder qualidade ao diminuir de tamanho.

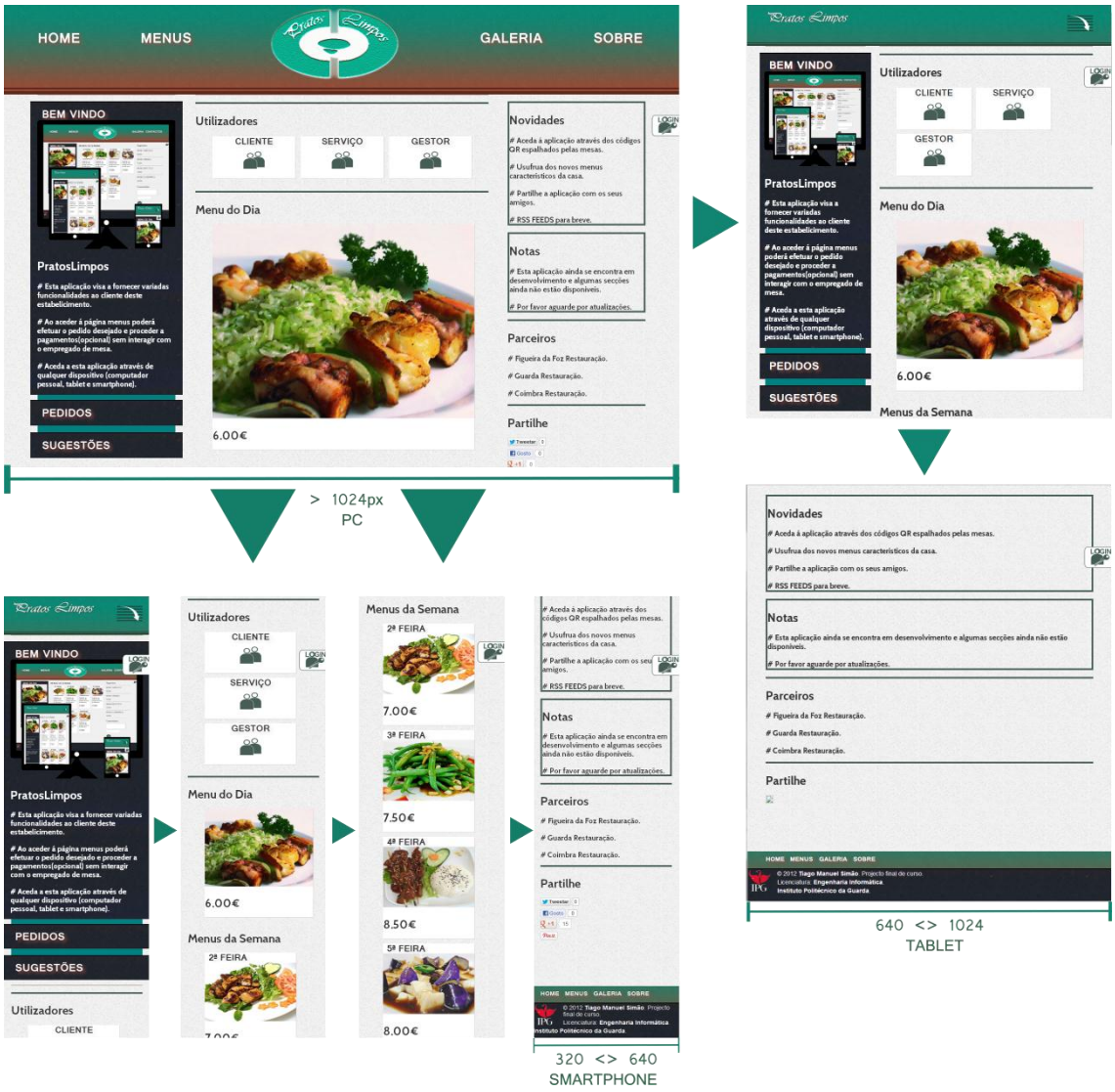


Figura 42: *HomePage*: PC, TABLET E SMARTPHONE, Pratos Limpos

6.4.3. REGISTO E LOGIN DE CLIENTE

O utilizador após navegar na página de boas vindas poderá efetuar login para usufruir de toda a aplicação. Para isso, existe o botão *login* que o irá remeter para a área de login do utilizador. Se já estiver registado basta inserir as credenciais do utilizador, senão, pode ser efetuado registo de novo cliente através do botão "*Registe-se Aqui*"**.** Na área de registo o utilizador terá de preencher todo o formulário e submetê-lo sem erros. Em ambas situações, seja após login ou registo, o cliente será redirecionado para a área de cliente. Caso seja administrador ou funcionário só será possível efetuar login, pois o registo é interno. Também serão redirecionados para as áreas de utilizador devidas. (ver **Figura 43**)

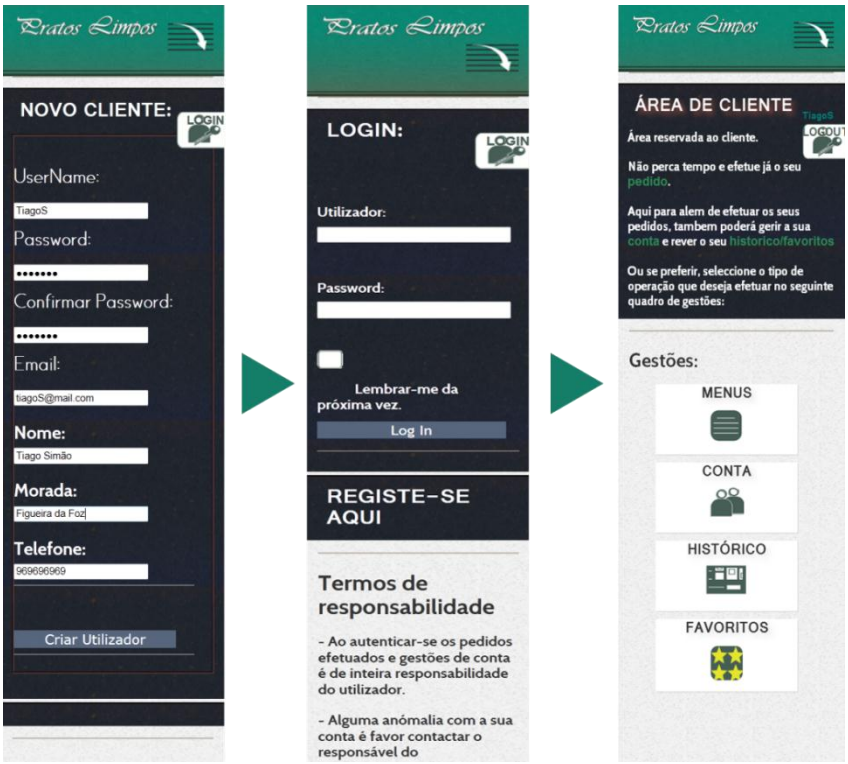


Figura 43: Registo e Login de Clientes (vista *Smartphone*), Pratos Limpos

6.4.4. EFETUAR PEDIDO (*cliente*)

Uma das operações mais importantes para o cliente é efetuar pedidos, onde a primeira área de visualização é a página respetiva aos menus (**Figura 44**). Cada tipo de menu contém toda a informação detalhada (nome, prato, acompanhamento e preço) e botões para adicionar os respetivos menus. Também é possível visualizar comentários de todos os utilizadores da aplicação. Para efetuar um pedido, o cliente pode pressionar o botão de compras (carrinho) ou o botão "*Faça o Seu Menu*". Em ambos será redirecionado para a mesma página de pedidos, com a exceção de que através do carrinho o cliente já tem adicionado na sua lista de compras os menus escolhidos anteriormente. Na área de pedido o cliente visualiza todos os menus já adicionados anteriormente e se pretender poderá adicionar novos produtos a essa mesma lista. Pode modificar, apagar e adicionar produtos. Caso tenha entrado via código QR, o número da mesa já se encontra preenchido, senão terá de fornecer o respetivo número. Pode também efetuar pré-pagamento (em desenvolvimento) e por fim confirmar o pedido através do botão "*Confirmar*". Em qualquer situação pode cancelar a operação. (ver **Figura 45**)

Figura 44: Área de Menus (vista *PC*), Pratos Limpos

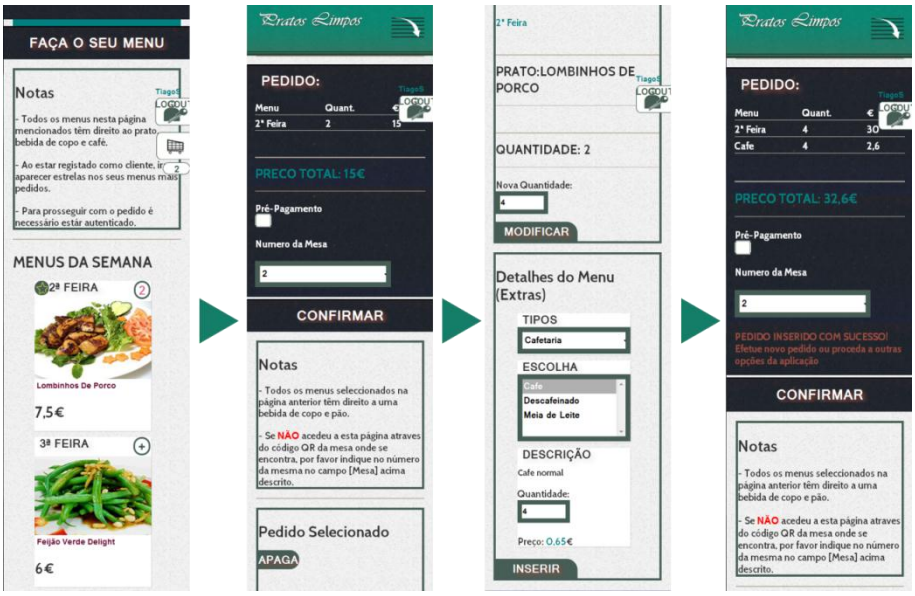


Figura 45: Efetuar Pedido (vista *Smartphone*), Pratos Limpos

6.4.5. FAVORITOS E HISTÓRICO DE PEDIDOS (*cliente*)

Na área de cliente, o utilizador também pode ver os seus favoritos e histórico de todos os pedidos já efetuados. No caso do histórico, este apresenta todos os pedidos efetuados pelo cliente, ordenados por tempo, onde prevalecem aqueles que ainda se encontram por tratar. Em relação aos favoritos, são apresentados os cinco menus mais pedidos. O cliente ao navegar na página de menus pode também verificar qual o menu que mais pediu até à data, este é sinalizado com uma estrela no canto superior esquerdo de cada bloco de menus. (ver **Figura 46**)

Figura 46: Favoritos e Histórico de Pedidos, Pratos Limpos

6.4.6. TRATAR PEDIDOS (*funcionário*)

Após efetuados os pedidos por parte do cliente, estes serão tratados pelo funcionário na respetiva área de funcionário. Aqui o utilizador pode verificar todos os pedidos por ordem de chegada, divididos em duas categorias: os pedidos pendentes de menus e de produtos individuais e os pedidos já tratados ou despachados. Em ambos, os pedidos são sinalizados com "cores", onde os vermelhos são pedidos pendentes sendo do tipo botão para o utilizador poder interagir e a verde os pedidos já tratados sem qualquer interação. Ao pressionar os botões vermelhos, os pedidos correspondentes passam para a tabela das mesas tratadas. Esta página encontra-se em atualização automática (*refresh*) de 15 em 15 segundos para dar entrada de novos pedidos. (ver **Figura 47**)

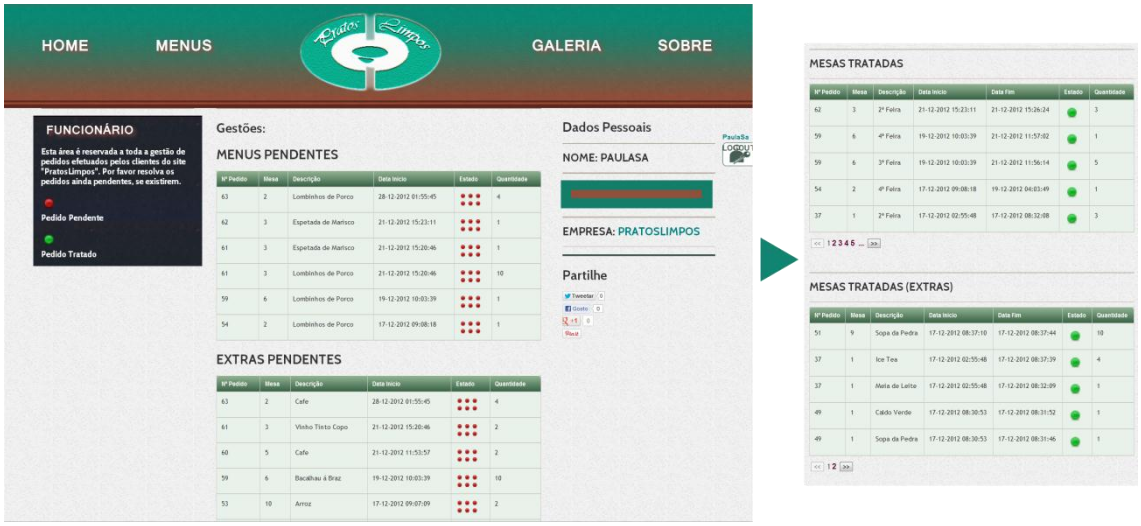


Figura 47: Área do Funcionário (vista *PC*): Tratar Pedidos, Pratos Limpos

6.4.7. GERIR CONTAS (*gestor*)

O gestor/administrador é um utilizador que também pode usufruir da mesma aplicação, pois é este que tem a função de inserir menus/produtos, gerir contas e enviar *Emails* para os clientes. Para gerir menus basta pressionar o botão "*Menus*" e será automaticamente redirecionado para a área de gestão. Nesta página os campos das tabelas: "*menus*" e "*descricaoProdutos*" são inseridos, atualizados ou apagados consoante a operação desejada. O gestor também pode bloquear e apagar os utilizadores do tipo "*cliente*". (ver **Figuras 48 e 49**)

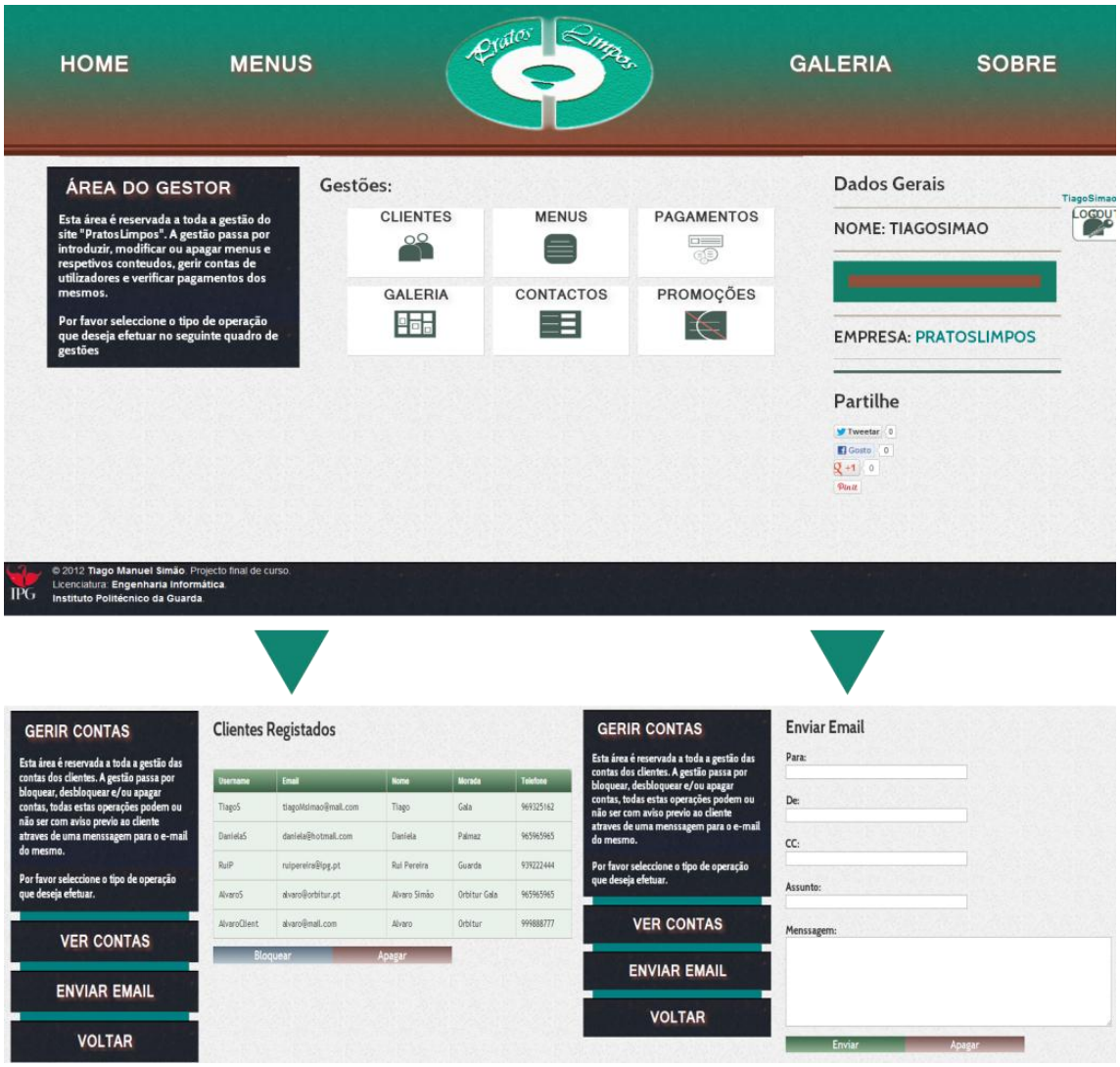


Figura 48: Área do Gestor, Pratos Limpos

Como foi mencionado o gestor pode bloquear a conta dos clientes registados na aplicação *Pratos Limpos* e ao fazê-lo estes serão impedidos de efetuar *login*, como demonstra a **Figura 49.**

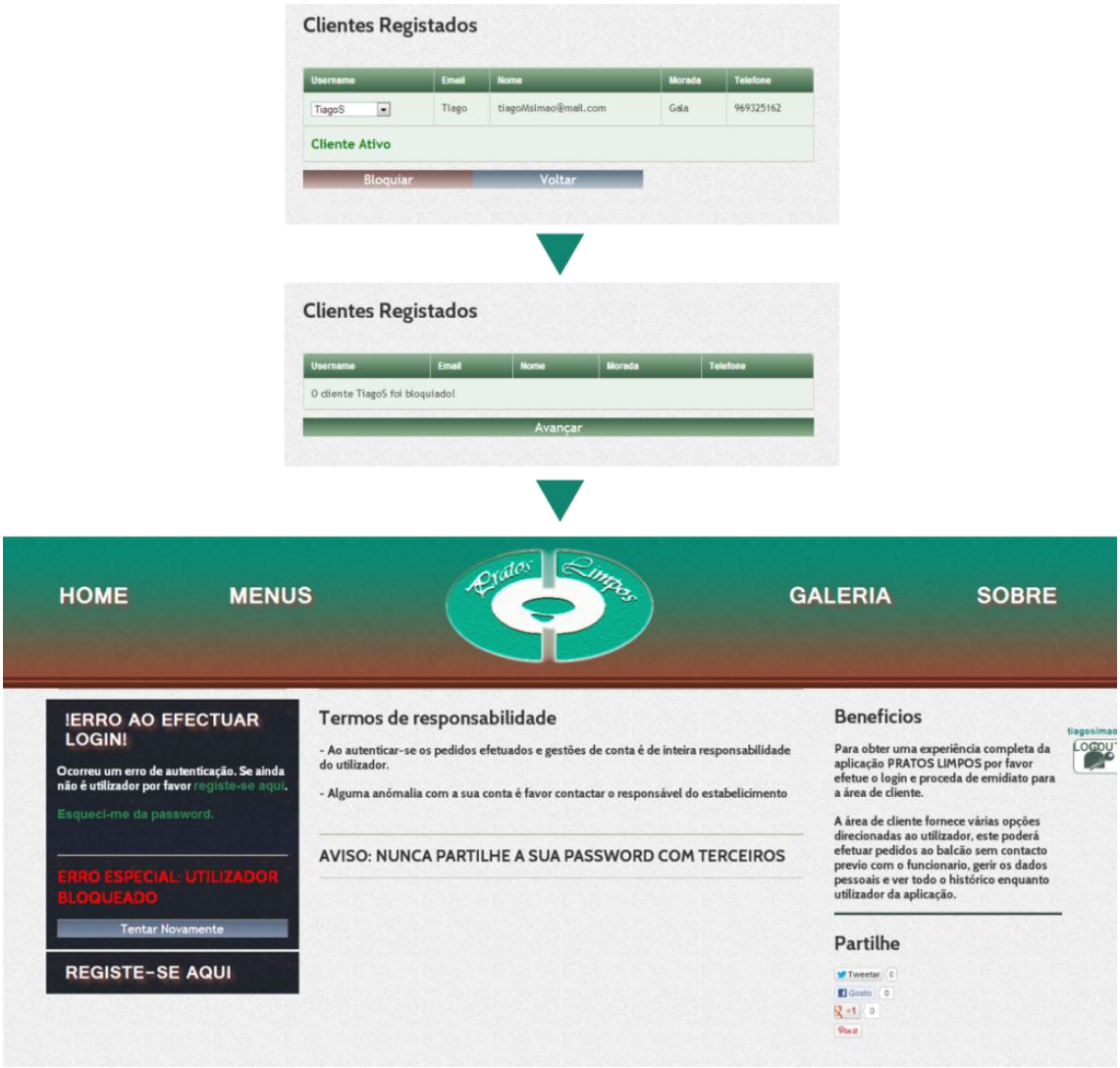


Figura 49: Bloquear Cliente: Função Gestor, Pratos Limpos

6.4.8. FOOTABLE (by themergency.com)

Um extra ou "*add-on*" adicionado a esta aplicação foi a tabela flexível "fooTable", criada por uma comunidade de *webdesigners* (*themergency*). Com esta tabela um dos grandes problemas do *Responsive Web Design* é resolvido, a questão das *tables*, onde aqui são substituídas por métodos implementados em *javascript* e manipuladas de maneira a corresponderem a um certo tamanho flexível. Em algumas figuras anteriores pode-se observar estas tabelas no tamanho original e ao ser redimensionado o conteúdo onde elas estão envolvidas, passam a respeitar o tamanho máximo onde se encontram (ver **Figura 50**). Existem muitas soluções para este problema, mas achei esta a mais completa, pois é interativa com o utilizador, é percetível e não esconde os dados menos relevantes como muitos outros métodos. Neste caso o que acontece é que são escondidos os campos mais à direita, mas serão visualizados ao pressionar o botão '+' concebido pelo *javascript*.

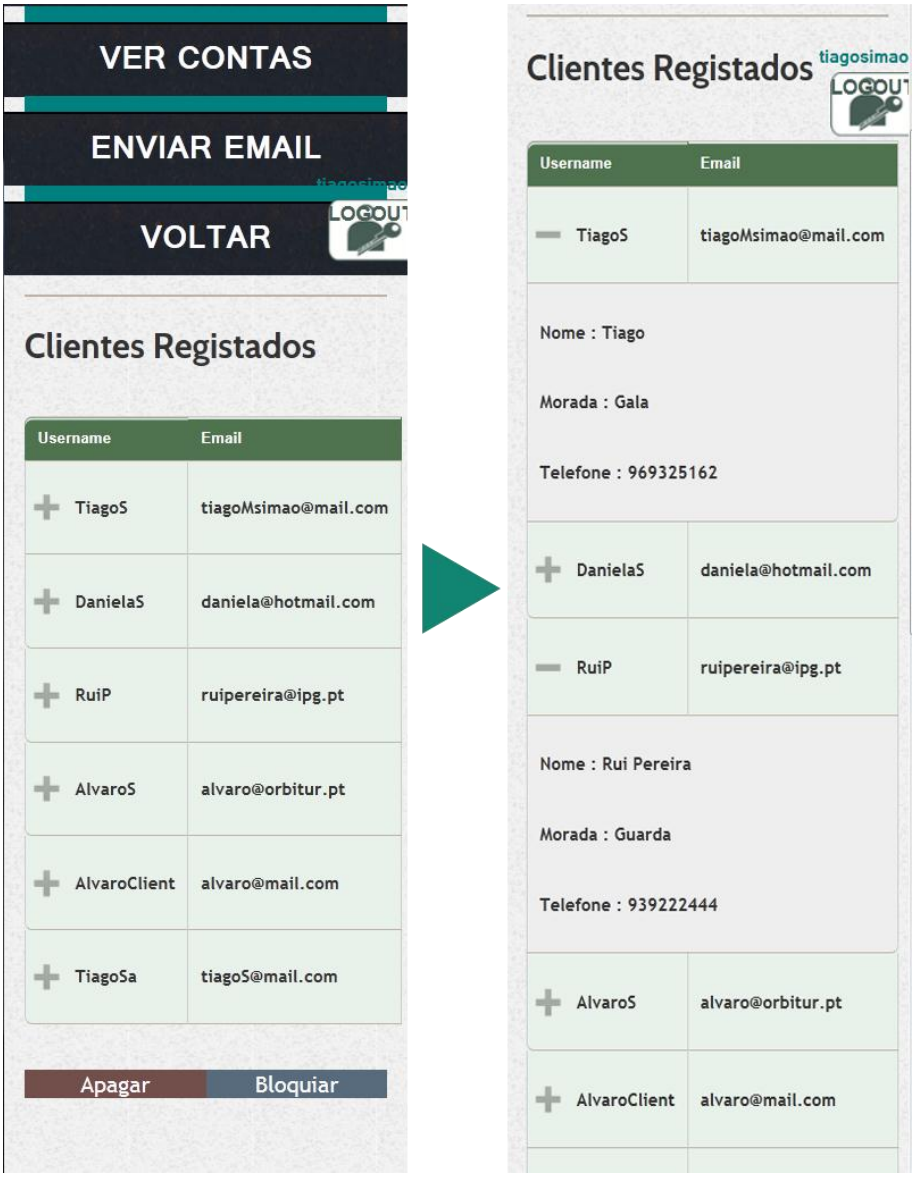


Figura 50: Tabelas Flexíveis (by themergency.com), Pratos Limpos

7 CONCLUSÃO E PREVISÕES

7.1. CONCLUSÃO

A cadeira de projeto foi uma das minhas preferidas nesta licenciatura de Informática, pois deu-me a liberdade de desenvolver uma aplicação pensada por mim, mas claro, aplicando todos os conhecimentos obtidos ao longo dos anos. Devido a alguma incompatibilidade de tempo para fazer o projeto este ainda se encontra em desenvolvimento, daí o relatório não ser o mais detalhado possível. No início devido à ferramenta Latex houve alguns contratempos com a sua configuração e formatação mas no fim o resultado do relatório foi satisfatório.

Em relação à aplicação *PratosLimpos*, ao desenvolvê-la obtive muitos conhecimentos que para mim eram incógnitos. O facto do *Responsive Web Design* ser complexo foi um grande obstáculo, o maior de todo o projeto. A tecnologia ASP.NET, apesar de já ter conhecimentos concretos, ao desenvolver a aplicação, aprendi muito mais ao pormenor da linguagem utilizada (C#) e todas as suas características. Por fim a utilização do MySql também foi satisfatória e produtiva para o meu conhecimento ao nível da linguagem SQL.

Posso dizer que este projeto necessitou de muito trabalho, de longas horas de pesquisa e de sucessivas tentativas para escolher a melhor forma de abordar o problema proposto. Contudo, posso afirmar que essas mesmas horas e essa mesma pesquisa me trouxeram um novo conhecimento que nunca pensei adquirir. Foi um trabalho duro mas satisfatório mas a cima de tudo enriquecedor para o meu futuro.

7.2. PREVISÕES

A aplicação *PratosLimpos* abrangeu várias matérias e mostrou-se multifacetada aos requisitos dos utilizadores. Apesar de ter sido desenvolvida para o ramo da restauração, facilmente se aplica a outro tipo de negócios desde que o objetivo seja fornecer ao cliente um produto completo e numa só aplicação, sem que exista qualquer tipo de contacto com o funcionário antes do pagamento. Também pode ser facilmente adaptado a um *website* de compras *on-line* com entrega ao domicílio. Entre outras aplicações variadas, ele próprio está adequado ao futuro da web, pois esse foi o principal objetivo, tornar com que fosse possível executá-lo de forma livre sem quaisquer problemas ao nível dos dispositivos.

O foco agora está em acabar a aplicação, onde as tarefas passam por concluir os *Web Services,* que não foram aqui falados, devido a estarem em fase de testes. Tornar a aplicação ainda mais flexível e compatível com os vários tamanhos de ecrãs dos diferentes dispositivos, fazendo assim o *layout* ainda mais *responsive*. Aplicar um bom sistema de segurança para evitar problemas com a base de dados, no caso de intrusos menos desejados. Por fim, colocar uma parte da aplicação *on-line* e a versão completa num meio de restauração.

Visto o futuro promissor da aplicação não poderei desistir dela. Como tal, acho importante referir a minha vontade de apostar num desenvolvimento desta ideia inovadora. Para alguém como eu, esta ideia representa uma nova era no atendimento, acabando pelo cliente ter um maior poder de decisão.

> *"If we're willing to research the needs of our users, and apply those ingredients carefully, then responsive web design is a powerful approach indeed.*
> *I can't wait to see the stories you'll tell with it."*
>
> (Marcotte, 2011)

8 BIBLIOGRAFIA

Emerson, J. (1 de 8 de 2012). *Using Svg Graphics Today*. Obtido em 20 de 10 de 2012, de Joshemerson: http://joshemerson.co.uk/blog/using-svg-graphics-today/

Gates, B. (2001). Interview: Bill Gates. *BBC News*, 2.

GuiStuff. (2007). *CSS: Units of Measurement*. Obtido em 20 de 10 de 2012, de guistuff.com: http://guistuff.com/css/css_units.html

JellyCode. (1 de 1 de 2012). *O seu Web Site em dispositivos móveis*. Obtido em 15 de 10 de 2012, de www.jellycode.pt: http://www.jellycode.pt/web-design/web-design-mobile/

Knuth, D. E. (1968). *The Art of Computer Programming.* USA: Addison-Wesley.

Lloyd, P. R. (23 de 4 de 2012). *Build Responsive Site Week Designing Responsively*. Obtido em 10 de 10 de 2012, de NetMagazine: http://www.netmagazine.com/tutorials/build-responsive-site-week-designing-responsively-part-1

Marcotte, E. (2011). *Responsive Web Design.* New York: A Book Apart.

Steffen, J. B. (23 de 1 de 2008). *O que são essas tais de metodologias Ágeis ?* Obtido em 20 de 10 de 2012, de www.ibm.com: https://www.ibm.com/developerworks/mydeveloperworks/blogs/rationalbrasil/entry/mas_o_que_s_c3_a3o_essas_tais_de_metodologias__c3_a1geis?lang=en

Systems, S. (1 de 1 de 2012). *UML Tutorial.* Obtido em 1 de 11 de 2012, de www.sparxsystems.com.au: http://sparxsystems.com.au/resources/tutorial_home/index.html

Wilcox, M. (1 de 1 de 2012). *adaptive-images*. Obtido em 15 de 11 de 2012, de Adaptive Images: http://adaptive-images.com/

9 ANEXOS

9.1. EXEMPLOS

A aplicação *pratoslimpos* foi desenvolvida com o intuito de fornecer ao utilizador várias experiências:

- Aceder de forma livre e não depender do dispositivo que utiliza
- Efetuar pedidos de comida numa zona de restauração
- Efetuar pagamentos de pedidos

O *website* está alojado em **http://www.pratoslimpos.co.nf**, mas poderá estar sujeita a futuras mudanças.

9.2. INTERFACE

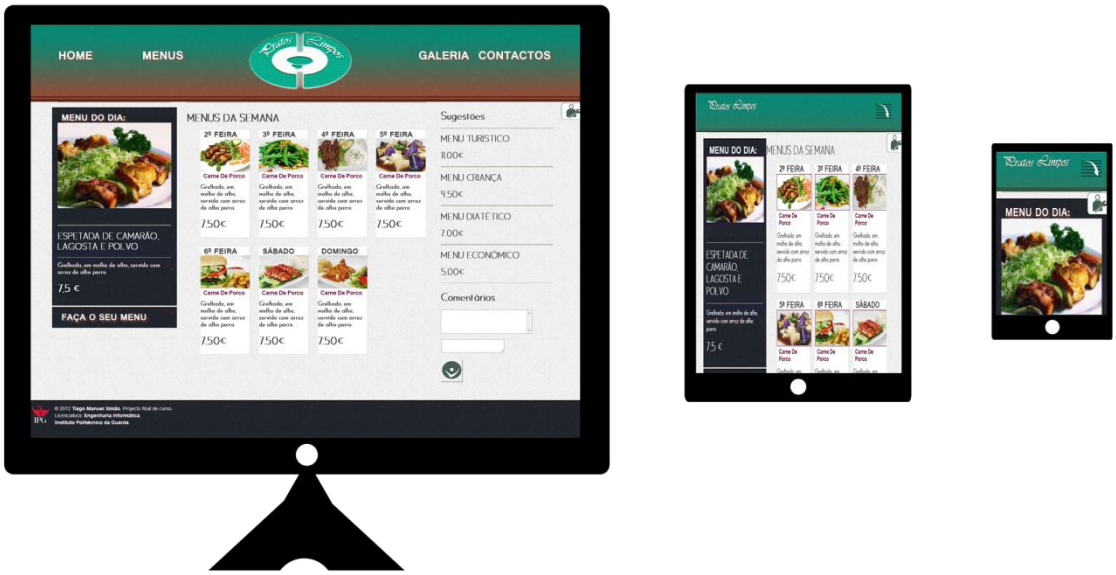


Figura 51: Diferentes *layouts* da aplicação *PratosLimpos*