UNIVERSIDADE DA BEIRA INTERIOR
Engenharia

Ocsi Simple
Uma Framework de Desenvolvimento WEB

(Versão Final Após Defesa)

António Pedro Rodrigues Gaspar

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Professor Doutor Simão Melo de Sousa

Covilhã, Julho de 2018

Agradecimentos

Após mais uma etapa da minha vida académica, não podia deixar de reconhecer todo o apoio, que foi fundamental para o desenvolvimento deste projeto.
Em primeiro lugar agradeço ao meu orientador de dissertação, o Professor Doutor Simão Melo de Sousa, por todo o apoio, incentivo, disponibilidade, compreensão e colaboração na realização deste projeto, assim como a todos os Professores que contribuíram com a sua transmissão de conhecimento para chegar até aqui.
Agradeço também à minha Mãe, ao meu Pai, à minha Irmã e ao meu Irmão, por todos estes anos de apoio incondicional, paciência nos momentos mais difíceis e por tudo aquilo que sou hoje.
Sendo cristão agradeço a Santo António, a São Judas Tadeu e a todas as Entidades representativas do que acredito, por me ajudarem a encontrar soluções para casos que muitas vezes pareciam irresolúveis.
Não podia deixar de agradecer a todos os colegas do laboratório *RELEASE* [rel], de forma especial ao Licínio Sousa, Nuno Pereira e ao João Reis, pelos bons momentos passados, por toda a colaboração e disponibilidade, durante a realização deste projeto.
Agradeço a todos os colegas da empresa onde exerço funções, pelo companheirismo em ambiente laboral e experiência transmitida, a qual contribuiu significativamente para o resultado alcançado.
Por fim o meu Bem Haja a todas as pessoas que me apoiaram e apoiam neste percurso de vida.

Resumo

Atualmente existem diferentes formas de produzir uma aplicação *World Wide Web* (WEB). Várias linguagens de programação dão suporte direto ou indireto a este tipo de desenvolvimento e sobre estas existem inúmeras *frameworks* já desenvolvidas. Com a fragmentação do desenvolvimento WEB distribuído por várias tecnologias, uma simples aplicação WEB, na grande maioria dos casos contém código *PHP*, *HTML* e *JavaScript*. Esta situação pode dar azo a erros sintáticos e semânticos durante o processo de produção, que vão ser refletidos no cliente. De forma a resolver grande parte destes problemas foi criada a *framework Ocsigen*. O seu *slogan* é *"develop once, safely deploy everywhere"* e tem como mais valia, a possibilidade de construir uma aplicação WEB completa, tanto parte cliente como parte servidor, exatamente com a mesma linguagem de programação, nomeadamente a linguagem *Objective Categorical Abstract Machine Language* (OCaml). Esta é também, a linguagem utilizada na construção da *framework Ocsigen*. O OCaml é uma linguagem de base funcional, multiparadigma de alto nível e fortemente tipificada. Tem por base, um poderoso compilador capaz de tornar o código produzido em algo extremamente determinístico a nível de segurança de tipos. Apesar da *framework Ocsigen* herdar características subjacentes à linguagem OCaml esta apresenta algumas limitações relativas à curva de aprendizagem, utilização devido à falta de documentação atualizada e instalação em diferentes sistemas operativos, como é o caso do *Windows*.

A presente dissertação tem como objetivo encontrar soluções para as limitações mencionadas. Para conseguir o objetivo proposto é então delineado um plano, que passa numa primeira fase, pelo estudo aprofundado da *framework Ocsigen*. Para que com base nos resultados obtidos seja possível compreender o seu funcionamento e contexto, como também as suas limitações, de forma mais detalhada, assim como os respetivos pontos críticos, nomeadamente a sua utilização e o suporte a diferentes sistemas operativos. Posteriormente, com base nestes dados é então projetada uma solução, que irá concretizar-se na forma de uma nova *framework*, a *framework Ocsi_Simple*. Esta nova *framework* irá agora permitir o uso da tecnologia *Ocsigen*, de forma mais amigável, simples e mais apetecível para a indústria do desenvolvimento WEB. É importante frisar que esta solução pretende completar a *framework Ocsigen*.

Palavras-chave

OCaml, *Ocsigen*, *Lwt*, *Tyxml*, *WEB*, *WEB Design*, *WEB Framework*, *Security*, *Ocsi_Simple*.

Abstract

Currently there are different ways to produce a WEB application. There are several programing languages backed by many frameworks that either directly or indirectly support WEB programming. With the fragmentation of WEB development, distributed by various technologies, a simple WEB application, in most cases contains *PHP*, *HTML* and *JavaScript* code. This situation can give rise to syntactic and semantic errors during the production process, which will be reflected on the client side. In order to solve a large part of these problems, the *Ocsigen framework* was created. The slogan is *"develop once, safely deploy everywhere"* and its added value, the possibility of building a complete WEB application, both client side and server side, exactly with the same programming language, namely OCaml. This is also the language used in the construction of the *Ocsigen framework*. OCaml is a high-level, multi-paradigm, strongly typed language with emphasis on functional programming, which is based on a powerful compiler, able to make very type safe programs. Though *Ocsigen* naturally shares common properties to the *OCaml* language, the former presents some limitations, such as, learning curve, lack of susbtantial documentation and installation issues with some operating systems like *Windows*.

The present dissertation aims to find solutions to the mentioned limitations. In order to achieve the proposed objective, a plan is outlined, of which, the first phase consists of an in-depth study of the *Ocsigen framework*. So that, based on the results obtained, it is possible to understand its operation and context, as well as its limitations in a more detailed way as well as the respective critical points, namely its use and support for different operating systems. Subsequently, based on this data, a solution is then designed, which will materialize, in the form of a new framework, the *Ocsi_Simple framework*. This new framework will now allow the use of *Ocsigen* in a friendlier, simpler and more appealing way for the WEB development industry. It is important to stress out that this solution intends to complete the *Ocsigen framework*.

Keywords

OCaml, *Ocsigen*, *Lwt*, *Tyxml*, *WEB*, *WEB Design*, *WEB Framework*, *Security*, *Ocsi_Simple*.

Conteúdo

Lista de Figuras

Listings

Acrónimos

ARM *Advanced RISC Machine*

CAM *Categorical Abstract Machine*

CAML *Categorical Abstract Machine Language*

CPS *Continuation Passing Style*

CSS *Cascading Style Sheets*

HTML *Hyper Text Markup Language*

HTTP *Hypertext Transfer Protocol*

JS *JavaScript*

LCF *Logic for Computable Functions*

Lwt *Lightweight Cooperative Threading*

ML *Meta Language*

OCaml *Objective Categorical Abstract Machine Language*

OPAM *OCaml Package Manager*

REST *Representational State Transfer*

SGBD Sistema de Gestão de Base de Dados

SO Sistema Operativo

URL *Uniform Resource Locator*

WEB *World Wide Web*

WSL *Windows Subsystem for Linux*

Capítulo 1

Introdução

1.1 Contexto da Dissertação

O ecossistema OCaml voltado para a criação de aplicações WEB, tem assistido a grandes evoluções. Exemplo disso é a *framework Ocsigen*, que foi criada especificamente para dar resposta a este propósito. Contudo existem outras *frameworks* com um objetivo semelhante. É o caso da *framework Opium*, que facilita a criação de aplicações *Representational State Transfer* (REST), assim como a *framework OCaml-Webmachine* [OCac], que complementa a *Opium* [OCab] ao nível dos protocolos suportados. Recentemente surgiu um novo dialeto de OCaml direcionado para o desenvolvimento WEB, o **ReasonML** [rea] que tem tido grande notoriedade. No entanto a sua utilização em conjunto com a *framework Ocsigen*, ainda está em desenvolvimento. Por essa razão a direção deste projeto vai de encontro à utilização da *framework Ocsigen* nativa. Apesar da relevância das *frameworks* citadas a *framework Ocsigen*, propõe um paradigma de desenvolvimento particular que lhe tem permitido uma afirmação única, no panorama tecnológico e industrial em que se insere. Construída totalmente em OCaml é composta por vários módulos. Entre eles distinguem-se, o módulo *Lwt*, cujo propósito é desenvolver mecanismos assíncronos, essenciais no desenvolvimento de aplicações WEB e o módulo *Tyxml* usado para tipificar o HTML produzido evitando desta forma possíveis erros sintáticos. Usando esta *framework* é possível aplicar o slogan *"develop once, safely deploy everywhere"*, devido à análise que o compilador faz ao código produzido garantindo este objetivo. No entanto a *framework Ocsigen* apresenta alguns problemas, expostos na secção seguinte (1.2) e desenvolvidos ao longo deste documento.

1.2 Definição do Problema

A *framework Ocsigen* está em franca evolução existindo no entanto alguns problemas associados à sua utilização. Entre eles está a elevada curva de aprendizagem, que requer tanto um conhecimento prévio do paradigma de programação funcional, como um conhecimento das metodologias de desenvolvimento WEB. A própria mecânica, da *framework Ocsigen* é complexa. Existe também um problema de compatibilidade com o sistema operativo *Windows*, o que compromete a sua utilização a uma escala industrial. Esta afirmação é comprovada pela experiência adquirida em contexto laboral, a qual envolveu várias tentativas de uso da tecnologia em análise. Porém os problemas referidos estavam sempre presentes. O âmbito da presente dissertação pretende resolver estes problemas. Baseadas numa experiência real, as situações referidas levam ás seguintes perguntas:

1. O que se pode fazer para tornar a *framework Ocsigen* mais simples de utilizar?
2. O que se pode fazer para tornar a *framework Ocsigen* utilizável em *Windows* e por conseguinte em ambiente empresarial, qualquer que seja o seu ecossistema tecnológico (*Windows*, *Linux*, etc.)?

Na próxima secção (1.3) é apresentado um plano de trabalhos. O qual tem como objetivo delinear uma estratégia para dar resposta a estas perguntas.

1.3 Plano de trabalho

Este projeto tem como objetivo principal implementar uma solução que permita uma maior facilidade e transparência na utilização da tecnologia *Ocsigen* levando-a a uma maior aceitação, em particular, em meio empresarial alheio ao ecossistema OCaml. A solução irá ser na prática uma nova *framework* designada por *Ocsi_Simple framework*. Esta é composta por mecanismos refinados, que no seu todo irão solucionar os problemas citados. Para chegar a esta solução serão contextualizadas as tecnologias envolvidas e implementado um exemplo de teste. Posteriormente será analisado, de forma a verificar os pontos críticos que apresentem um comportamento de difícil implementação. Com base neste último passo parte-se então para a análise da arquitetura da solução, que irá culminar numa nova *framework*. Com o mesmo exemplo de teste, implementado anteriormente, este será reconvertido, utilizando a nova *framework* originando assim a fase de implementação e validação, a qual tem como base nos resultados obtidos. Paralelamente serão estudas formas de compatibilizar, a solução com o sistema operativo *Windows*. De forma a resumir o encadeamento do processo, é delineada a seguinte lista de tarefas:

- Contextualização do problema, com as tecnologias envolvidas;
- Implementação do exemplo de teste;
- Análise e desenho de uma arquitetura de solução;
- Implementação e validação da nova *framework*;
- Documentação do processo e resultados.

1.4 Resultados e Contribuições

Com base na definição do problema (1.2) e da experiência adquirida em contexto industrial, foi originada uma análise profunda sobre as tecnologias envolvidas e respetivas metodologias. Esta culminou num plano de trabalhos (1.3), que responde diretamente às questões levantadas na secção **Definição do Problema** (1.2). Desta forma à pergunta, 1 - O que se pode fazer para tornar a *framework Ocsigen* mais simples de utilizar?. É apresentada como resposta a criação de uma nova *framework*, denominada por *Ocsi_simple Framework* e que é composta por três módulos:

- ***Ocsi_simple_persist*** é o módulo trata da persistência de dados, assegura o acesso aos SGBD *Mysql*, *PostgreSQL* e *SQLite*, facilitando a sua utilização. Para o seu uso é apenas necessário identificar o SGBD pretendido e configurar a ligação ao motor de base de dados. É também importante referir que este módulo é escalável, podendo ser aplicado a outros motores de bases de dados.

- ***Ocsi_simple*** é o módulo que permite agora, de uma forma simples, concisa, estruturada e limpa, criar qualquer tipo de página HTML, sintaticamente correta. Este é o módulo que quando utilizado, influência positivamente a produtividade e a organização do código produzido. É um dos módulos estruturais da nova *framework*, pois interliga a utilização dos outros módulos presentes nesta, proporcionado desta forma, um padrão de desenvolvimento durante a sua utilização.

- ***Ocsi_simple_js*** é o módulo que implementa várias funções que replicam o comportamento, de funções *JavaScript* habitualmente utilizadas, pela grande maioria dos programadores WEB. A título de exemplo, o uso de uma função para adicionar um elemento a outro elemento. Este é o módulo, que permite dar aquele toque de interatividade nas páginas ou aplicações WEB criadas. Foi pensado e desenvolvido para facilitar a manipulação de elementos na parte cliente, da aplicação WEB.

Relativamente à pergunta 2 - O que se pode fazer, para tornar a *framework Ocsigen* utilizável em *Windows* e por conseguinte em ambiente empresarial, qualquer que seja o seu ecossistema tecnológico (*Windows*, *Linux*, etc.)?. Como resposta é apresentado no capítulo 4 uma secção (4.3). O qual culmina no anexo **Ocsigen no Windows** (B) e no anexo **Tutorial Ocsi_simple Framework** (C). O anexo **Ocsigen no Windows** (B), tem como finalidade explicar o processo de instalação e utilização da tecnologia *Ocsigen* em ambiente *Windows*. Desta forma é alcançado o objetivo da presente dissertação, o qual culminou num contributo considerado relevante à *framework Ocsigen* tendo em conta o desafio colocado na secção anterior. É importante frisar que a solução desenvolvida pretende completar a *framework Ocsigen*. Os resultados práticos obtidos são disponibilizadas na forma de cinco provas de conceito desenvolvidas ao longo da dissertação:

- `http://ocsisimple.ddns.net:8080` apresenta a prova de conceito 1 (4.1) presente no capítulo 4. Esta foi desenvolvida unicamente com a plataforma *Ocsigen* nativa.

- `http://ocsisimple.ddns.net:8081` apresenta a prova de conceito 2 (4.6) presente no capítulo 4. A qual foi desenvolvida com o resultado desta dissertação, a *Ocsi_simple Framework*.

- `http://ocsisimple.ddns.net:8082` apresenta uma página desenvolvida com a *Ocsi_simple Framework* a partir de um *template*.

- *`http://ocsisimple.ddns.net:8083` apresenta uma página onde foram testados os módulos da* Ocsi_simple Framework. *Nomeadamente os módulos* Ocsi_simple, Ocsi_simple_js *e* Ocsi_simple_persist.

- *http://ocsisimple.ddns.net:8084 apresenta o caso de estudo desenvolvido no anexo **Tutorial Ocsi_simple Framework** (C).*

1.5 Organização do Documento

De forma a apresentar o trabalho realizado, este documento encontra-se estruturado da seguinte forma:

1. O atual capítulo –**Introdução** (1)– apresenta o projeto e o âmbito em que se insere, assim como a definição do problema, plano de trabalho e resultados. Descreve também a respetiva organização deste documento.

2. O segundo capítulo –**Programação WEB em OCaml** (2)– descreve as tecnologias envolvidas, os conceitos teóricos presentes no contexto do projeto e por fim, os trabalhos relacionados que tem como objetivo principal dar ao leitor um termo de comparação com outra tecnologia semelhante ao caso de estudo.

3. O terceiro capítulo –**Análise e Arquitetura da Solução** (3)– trata todo o processo de engenharia de *software relativo ao projeto, assim como a estruturação do modelo de persistência.*

4. *O quarto capítulo –**Implementação e Validação** (4)- descreve o processo e implementação das diferentes partes do projeto. São também descritas as provas de conceito do processo de desenvolvimento. Estas englobam as diferentes partes apresentadas ao logo do documento.*

5. *O quinto capítulo –**Conclusões e Trabalho Futuro** (5) – apresenta os resultados do trabalho desenvolvido durante todo o processo. Por fim é também apresentado o trabalho que futuramente poderá vir a ser desenvolvido com base no presente projeto.*

6. *O anexo* A –***Diagramas de casos de uso da* Ocsi_simple Framework** *(A)- apresenta de forma exaustiva os diagramas de caso de uso referentes à* Ocsi_simple Framework. *Tem como objetivo complementar o capitulo 3.*

7. *O anexo* B –***Ocsigen no Windows** (B)- descreve todo o procedimento necessário, para configurar um ambiente de desenvolvimento no sistema operativo* Windows, *adequado à* framework Ocsigen.

8. *O anexo* C –***Tutorial Ocsi_simple Framework** (C)- apresenta de forma detalhada e prática o uso do resultado deste projeto, a nova* framework *com um exemplo didático para dar a conhecer ao leitor as potencialidades da sua utilização.*

Capítulo 2

Programação WEB em OCaml

2.1 Introdução

Com o objetivo de contextualizar a matéria apresentada neste documento, o presente capítulo, aborda as tecnologias utilizadas durante o desenvolvimento do projeto descrevendo a sua origem e o porquê da sua utilização. Dada a importância do tema é fundamental conhecer os conceitos teóricos, associados às tecnologias envolvidas e por essa razão são aqui também apresentados. Em último lugar e de forma a justificar a utilização das tecnologias envolvidas é apresentado um paralelo, entre estas e outras bastante utilizadas, para um fim semelhante.

2.2 Tecnologias Envolvidas

2.2.1 OCaml

A OCaml é uma linguagem de programação, multiparadigma (funcional, imperativa e orientada a objetos), com características bastante interessantes analisadas em seguida. A sua origem remonta aos primórdios da *Meta Language* (ML) original baseada na teoria de conjuntos e no cálculo λ. Foi desenvolvida por Robin Milner na década de 1970, com o objetivo de criar um *Logic for Computable Functions* (LCF) que concretizou com grande sucesso na altura marcando definitivamente a história da ciência computacional. Porém com uma utilização cada vez mais crescente da ML original por instituições académicas, os seus utilizadores rapidamente entenderam que o conceito inicial podia ser adaptado à programação corrente dando assim, origem ao aparecimento de vários dialetos da ML original. Uma dessas instituições foi a *INRIA* [INR] de França. Foi na *INRIA* [INR], que surgiu a primeira versão do que é conhecido nos dias de hoje, por OCaml. Criada por Pierre-Louis e originalmente conhecida por *Categorical Abstract Machine* (CAM) esta era uma máquina virtual capaz de executar procedimentos funcionais. Foi então que em 1984, outros investigadores associados, dos quais se destaca *Gérard Huet*, criaram uma versão ML para ser executar na CAM surgindo assim a *Categorical Abstract Machine Language* (CAML). No entanto esta implementação era extremamente custosa relativamente ao uso de memória. De forma a mitigar os problemas anteriores, em meados dos anos 90, foi desenvolvida por *Xavier Leroy* e *Damien Doligez* a *ZINC*. A *ZINC* [Ler90] é uma máquina virtual mais rápida e eficiente, que na mesma altura proporcionou a origem do *CAML Light*, uma versão muito mais leve e rápida que a antiga CAM.
Também na mesma altura estava em voga o paradigma de orientação a objetos. Foi então que em 1998, foi adicionada mais uma camada à *CAML Light* desenvolvida por

Didier Rémy e *Jérome Vouillon* [RV98] dando origem ao que hoje em dia se conhece por OCaml. Com o desenvolvimento da *ZINC* [Ler90] atualmente esta permite, não só interpretar o código e compilá-lo para *bytecode, como também compilá-lo para código nativo suportando atualmente, várias arquiteturas tendo o código gerado, um desempenho muito próximo da linguagem* C.

Para além de ser uma linguagem funcional, ou seja o seu paradigma é orientado à função como elemento principal e estruturante, o OCaml implementa também o paradigma imperativo e o de orientação a objetos, tornando-a numa linguagem multiparadigma, bem fundamentada devido à sua herança da família ML e de alto desempenho.

Atualmente o OCaml tem como símbolo um dromedário devido à ligação óbvia entre o seu nome e o nome do animal (2.1).

Figura 2.1: Logótipo do OCaml
Fonte: `https://OCaml.org/`.

A versão do OCaml utilizado para o desenvolvimento deste projeto é a 4.06.1*, que à data deste documento é versão oficial, mais recente. Esta é a linguagem sobre a qual, a* Plataforma Ocsigen*(2.2.3) está também construída, e por essa razão será a linguagem utilizada para o desenvolvimento deste projeto [OCaa].*

2.2.2 OPAM

O OCaml Package Manager (OPAM) [OPAc] é o gestor de pacotes oficial para a linguagem OCaml. Este gestor de pacotes funciona como repositório central que permite a gestão de versões dos pacotes, como também permite rapidamente preparar um ambiente de desenvolvimento organizado e pronto a trabalhar com a linguagem OCaml. No contexto deste projeto esta ferramenta será usada, para permitir a rápida instalação e preparação do ambiente de desenvolvimento necessário ao tema proposto.

2.2.3 Plataforma Ocsigen

A plataforma Ocsigen *[OCSe] é uma plataforma multicamada capaz de dar resposta ao desenvolvimento voltado para a WEB. É composta por vários projetos, cada um direcionado a um foco especifico do desenvolvimento WEB, como é o caso do* Ocsigen Server *(2.2.4), do* JS-Of-OCaml *(2.2.5), da* Lwt *(2.2.6), do* Eliom *(2.2.7) e do* Tyxml *(2.2.8). Atualmente a imagem do* Ocsigen *é representada pelo seguinte logótipo (2.2).*

Figura 2.2: Logótipo do Ocsigen
Fonte: `https://ocsigen.org/`.

O Ocsigen *tem como objetivo principal proporcionar o desenvolvimento WEB através da linguagem OCaml aproveitando características, tais como, o desempenho e a fiabilidade da mesma. Este processo é conseguido através da identificação do código* Servidor *e código* Cliente *durante a compilação, onde este último vai dar origem a código* JavaScript. *A par deste objetivo está outro que a diferencia, a possibilidade de compilar o código produzido, para as diferentes plataformas existentes, como por exemplo para Advanced RISC Machine (ARM) e arquiteturas* Mobile. *O processo é apresentado de forma sintética na imagem seguinte (2.3) [OCSb].*

Figura 2.3: Plataforma multicamada
Fonte: `https://ocsigen.org/`.

2.2.4 Ocsigen Server

O Ocsigen Server *[OCSf] é um projeto associado à plataforma* Ocsigen *tem como objetivo servir as aplicações WEB criadas sobre esta. Permite também associar aplicações já criadas em OCaml e previamente compiladas de forma a facilitar a integração com o próprio ecossistema OCaml. O* Ocsigen Server *implementa entre outros, o protocolo Hypertext Transfer Protocol (HTTP) e proporciona todas as configurações possíveis sobre este. No entanto é importante ressalvar que esta implementação ainda está a ser otimizada pela equipa que o desenvolve. De forma sintética o* Ocsigen Server *é um servidor WEB desenvolvido em OCaml, que serve as aplicações desenvolvidas nesta plataforma. No contexto deste projeto, o* Ocsigen Server *é utilizado como servidor das aplicações desenvolvidas com o objetivo desta dissertação. O* Ocsigen Server *é representado pela seguinte imagem (2.4).*

Figura 2.4: Logótipo do Ocsigen Server
Fonte: `https://ocsigen.org/`.

2.2.5 JS-Of-OCaml

O projeto JS-Of-OCaml *[OCSc] é um dos mais importantes, pois é este que permite criar a parte cliente da aplicação WEB, através da geração de código* JavaScript. *Este projeto não só proporciona a transformação de código OCaml para* JavaScript, *na plataforma* Ocsigen, *como também para todo o ecossistema OCaml. No contexto desta dissertação o* JS-Of-OCaml *é utilizado para dinamizar a interface cliente-dispositivo e segmentar funcionalidades que permitiram maior agilidade durante o desenvolvimento de outras aplicações WEB, sobre esta plataforma. A imagem do* JS-Of-OCaml *é apresentada abaixo (2.5).*

Figura 2.5: Logótipo do Ocsigen JS-Of-OCaml
Fonte: `https://ocsigen.org/`.

2.2.6 Lwt

A Lightweight Cooperative Threading (Lwt) [OCSd] é parte integrante do projeto Ocsigen. *É uma biblioteca de programação concorrente que permite a utilização de* threads *cooperativas, não bloqueantes. Está estruturada de forma a que qualquer operação computacional, seja considerada uma ação, que pode ou não ser executada num futuro próximo. A esta ação é dado o nome de promessa. A utilização da promessa passa em primeiro lugar, pela sua avaliação e posteriormente a sua execução, que poderá dar origem a um valor encapsulado (2.3.4), ou mesmo a uma exceção em caso de falha da operação. Com a utilização desta biblioteca, tanto o código principal como a promessa são executados na mesma* thread *ou seja a* thread *principal permitindo desta forma, o não bloqueio ou preempção de todo o programa. É também possível segmentar a utilização destas* threads, *de forma a assegurar outras situações, tais como, leituras demoradas ou chamadas a aplicações, que não tenham suporte à Lwt. A utilização da Lwt está inerente a todo o projeto desenvolvido, pois faz parte da base da plataforma* Ocsigen. *A imagem representativa da Lwt é apresentada abaixo (2.6).*

Figura 2.6: Logótipo do Ocsigen Lwt
Fonte: `https://ocsigen.org/`.

2.2.7 Eliom

O Eliom *[OCSa] é o projeto principal, da plataforma* Ocsigen. *Este projeto resulta num módulo que tem como principal função permitir a interação com toda a camada HTTP. De forma prática o* Eliom *permite definir como ponto de entrada qualquer Uniform Resource Locator (URL) criado para o efeito, como também, definir as ações que irão ser executadas, quando o ponto de entrada for despoletado. No* Eliom *estas ações têm o nome de serviços, que conjuntamente com os métodos REST associados ao protocolo HTTP, proporcionam o desenvolvimento de qualquer tipo de aplicação WEB. No contexto desta dissertação o* Eliom *é utilizado de forma a garantir as funcionalidades inerentes ao mesmo. A imagem representativa do* Eliom *é apresentada abaixo (2.7).*

Figura 2.7: Logótipo do Ocsigen Eliom
Fonte: `https://ocsigen.org/`.

2.2.8 TyXML e CSS3

O TyXML *[OCSg] é um projeto da plataforma* Ocsigen*, que permite a criação de páginas HTML tipificadas e semanticamente corretas. Este projeto está sob a forma de um módulo. Na prática este módulo serve para a criação da interface de uma aplicação WEB, de uma forma particular. Forma essa que entre outras situações esta dissertação visa refinar e aproximar do atual desenvolvimento* Front-end*. Outra tecnologia associada ao uso do* TyXML *é o* CSS3*, a qual permite a configuração do visual neste tipo de desenvolvimentos. A imagem representativa do* TyXML *é apresentada abaixo (2.8).*

Figura 2.8: Logótipo do Ocsigen TyXML
Fonte: `https://ocsigen.org/`.

2.2.9 Interface com Bases de Dados

Atualmente qualquer aplicação WEB necessita de um conjunto de funcionalidades que permitam guardar informações de forma persistente, para ser posteriormente consultada ou até mesmo carregada, no caso de conteúdos dinâmicos. Apesar da plataforma Ocsigen *apresentar suporte para, mecanismos de persistência de dados associados a alguns motores de base de dados e ficheiros estáticos apresenta também algumas limitações. Destacam-se de forma mais evidente, a curva de aprendizagem que não proporciona o seu uso imediato e a complexidade de uso. Ora de forma a resolver a presente dissertação propõem um mecanismo de persistência alternativo, ao atualmente implementado na plataforma* Ocsigen*, o qual tem como objetivo mitigar as limitações referidas. Desta forma será possível, a qualquer utilizador da nova* framework *resultado desta dissertação rapidamente montar e utilizar a camada de persistência, para os motores de base de dados* PostgreSQL *([Pos]),* Mysql *([Mys]) e* SQLite *([SQL]) permitindo futuramente associar outros motores de base de dados.*

Figura 2.9: Logótipos dos Motores de Base de Dados suportados pela nova *framework*.

2.3 Conceitos Teóricos

A presente dissertação culmina numa nova framework*, que visa mitigar os problemas anteriormente referidos. Mas para entender como foi possível criar a solução apresentada, é de extrema importância, que o leitor tenha conhecimento dos fundamentos teóricos desta. Os fundamentos teóricos, aqui expostos são descritos de forma articulada com exemplos, para desta forma dar a conhecer e contextualizar a sua utilização. Os exemplos expostos [corc] baseiam-se no curso* Structures and Functional Programming *[cora] da* Cornell University *[corb] e no livro* Real World OCaml *[Min13].*

2.3.1 Módulos

Os módulos [Ler00], na perspetiva da linguagem OCaml são estruturas únicas que podem ter no seu corpo, funções, estruturas de dados e até outros módulos. É um mecanismo do OCaml para o encapsulamento e reutilização de código, a par da orientação a objetos. O módulo pode também ser instanciado, situação que irá ser vista na subsecção funtores *(2.3.3). Da mesma forma que pode ser instanciado, pode também implementar uma* Interface*, situação analisada na subsecção seguinte,* Interfaces *(2.3.2). A titulo de exemplo segue o seguinte código.*

```
1 module Lstack = struct
2   type 'a t = 'a list
3   let empty = []
4   let push x s = x::s
5   let peek = function [] -> failwith "empty" | x::_ -> x
6 end
```

Listing 2.1: Implementação do módulo *Lstack*.

O código no exemplo (2.1) retrata a implementação do módulo Lstack*, o qual apresenta a estrutura de uma* Stack*. Este módulo está preparado para receber inúmeros tipos de dados, o que faz dele um módulo polimórfico. No entanto poderão existir situações, em que este módulo terá que ser reescrito para assumir tipos de dados mais específicos, como por exemplo, assumir o tipo de uma árvore. Sabendo que a estrutura de uma* Stack *é sempre a mesma e esta pode assumir várias implementações torna-se essencial existir um mecanismo, que represente a implementação. Esse mecanismo existe em OCaml. É designado por* Interface *e é introduzido na próxima subsecção (2.3.2).*

2.3.2 Interfaces

Uma interface ou assinatura no contexto de OCaml é um módulo que apenas apresenta as estruturas de dados, o tipo ou tipos do módulo, as exceções e por fim os protótipos das funções com os respetivos tipos dos parâmetros e o tipo devolvido pela função. Uma interface *ou assinatura representa assim, a implementação que um módulo terá de assumir. Para contextualizar a* interface *é apresentado o seguinte exemplo (2.2).*

```
module type StackSig = sig
  type 'a t
  val empty : 'a t
  val push  : 'a -> 'a t -> 'a t
  val peek  : 'a t -> 'a
end
```

Listing 2.2: Definição da *Interface StackSig*.

Como se pode constatar no exemplo (2.2), a Interface StackSig *define apenas o que tem de ser implementado ou definido, no módulo* ListStack *do exemplo (2.1). Desta forma é possível existirem várias implementações que seguem a mesma estrutura. O próximo exemplo, com base nos exemplos anteriores apresenta mais uma implementação da* interface StackSig. *Este tem como objetivo representar a estrutura de uma árvore para uma* Stack *e tem por nome* TreeStack.

```
module type StackSig = sig
  type 'a t
  val empty : 'a t
  val push  : 'a t -> 'a t -> 'a t
  val peek  : 'a t -> 'a t
end
module ListStack = struct
  type 'a t = Empty | Entry of 'a  * 'a t
  let empty = Empty
  let push x s = match x with
                  | Entry (a,b) -> Entry (a,s)
                  |Empty -> Empty
  let peek = function Empty -> Empty | Entry(x,_) -> Entry(x,Empty)
end
module TreeStack= struct
  type 'a t = Node of 'a t * 'a * 'a t | Leaf
  let empty = Leaf
  let push (x:'a t) s = match x with
                        |Node (a,b,c) -> Node (x,b,s)
                        |Leaf -> Leaf
  let peek = function Leaf -> Leaf | Node (left , x, right) -> left
end
```

Listing 2.3: Múltiplas implementações da Interface *StackSig*.

No entanto múltiplas implementações da mesma interface podem, por vezes criar duplicação de código. É exemplo disso (2.4)(2.5) a criação de testes para garantir a operabilidade dos módulos presentes, no exemplo anterior (2.3).

```
(* OCaml TopLevel Testes *)
#open ListStack;;
#assert ((empty |> push (Entry (5,Empty)) |> peek) = Empty);;
#assert ((empty |> push (Entry (5,Empty)) |> peek) = Entry (5,Empty));;
```

Listing 2.4: Duplicação de código (Parte 1).

```
(* OCaml TopLevel Testes *)
#open TreeStack;;
#assert ((empty |> push (Node (Leaf,5,Leaf)) |> peek) = Leaf);;
#assert ((empty |> push (Node (Leaf,5,Leaf)) |> peek) = Node (Leaf,5,Leaf));;
```

Listing 2.5: Duplicação de código (Parte 2).

Como se pode verificar, para dois módulos diferentes foi escrito o mesmo código. No entanto o OCaml permite resolver esta situação, através de módulos parametrizados, ou por outra palavras funtores *apresentados na seguinte subsecção (2.3.3).*

2.3.3 Funtores

Os funtores são objetos matemáticos presentes na teoria das categorias [Pie91]. Têm como objetivo mapear estruturas, que recebem outras estruturas como parâmetros resultando em novas estruturas. Na prática um funtor em OCaml é um módulo α que recebe como parâmetro, um ou vários módulos do tipo β e dá origem a um módulo γ. O módulo γ é então um módulo instanciado, com um ou vários módulos do tipo β. Ou seja, o módulo do tipo β é um módulo que implementa a interface β. De forma a dissipar algumas dúvidas é apresentado um exemplo, que tem como objetivo, a implementação de um funtor *que permite resolver o problema de duplicação de código presente no exemplo anterior (2.4).*

```
module StackTester (S:StackSig) = struct
  let teste v1 v2 = if (S.( empty |> push v1 |> peek ) = v2) then
                        print_endline "O teste foi concluído com sucesso!"
                    else
                        print_endline "O teste falhou!"
end
```

Listing 2.6: Implementação de um funtor para módulos do tipo *StackSig*.

Como se pode verificar no exemplo anterior (2.6), o módulo parametrizado ou funtor StackTester *recebe como parâmetro um módulo do tipo* StackSig. *Este implementa a função* teste *que é composta por dois parâmetros, o* v1 *que representa o valor introduzido na* Stack *e o* v2, *valor esperado obtido da* Stack. *Desta forma a utilização deste teste torna-se transparente para todas as implementações que respeitam a* interface StackSig. *Como se pode verificar nos seguintes exemplos (2.7), (2.8) e (2.9).*

```
module type StackSig = sig
  type 'a t
  val empty : 'a t
  val push  : 'a t -> 'a t -> 'a t
  val peek  : 'a t -> 'a t
end
```

Listing 2.7: Exemplo completo sobre o *funtor StackTester* (Parte 1).

```
module ListStack = struct
  type 'a t = Empty | Entry of 'a  * 'a t
  let empty = Empty
  let push x s = match x with
                  | Entry (a,b) -> Entry (a,s)
                  |Empty -> Empty
  let peek = function Empty -> Empty | Entry(x,_) -> Entry(x,Empty)
end
module TreeStack= struct
  type 'a t = Node of 'a t * 'a * 'a t | Leaf
  let empty = Leaf
  let push (x:'a t) s = match x with
                        |Node (a,b,c) -> Node (x,b,s)
                        |Leaf -> Leaf
  let peek = function Leaf -> Leaf | Node (left, x, right) -> left
end
```

Listing 2.8: Exemplo completo sobre o *funtor StackTester* (Parte 2).

```
module StackTester (S:StackSig) = struct
  let teste v1 v2 = if (S.( empty |> push v1 |> peek ) = v2) then
                        print_endline "O teste foi concluído com sucesso!"
                    else
                        print_endline "O teste falhou!"
end;;
(* OCaml TopLevel*)
# module SkTesterExec = StackTester(TreeStack) ;;
# SkTesterExec.teste (Node (Leaf,5,Leaf))  Leaf;;
O teste falhou!
# SkTesterExec.teste  (Node (Leaf,5,Leaf))  (Node (Leaf,5,Leaf));;
O teste foi concluído com sucesso!
# module SkTesterExec = StackTester(ListStack) ;;
# SkTesterExec.teste  (Entry (5,Empty)) Empty ;;
O teste falhou!
# SkTesterExec.teste  (Entry (5,Empty)) (Entry (5,Empty)) ;;
O teste foi concluído com sucesso!
```

Listing 2.9: Exemplo completo sobre o *funtor StackTester* (Parte 3).

2.3.4 Mónadas

Os mónadas *[Mog91] são estruturas, que à semelhança dos funtores (2.3.3) também pertencem à teoria das categorias [Pie91]. Têm como objetivo encapsular valores ou estruturas que irão dar origem a novos valores ou estruturas do tipo do* mónada*. Estas podem ser combinadas entre elas. Um mónada representa computações com estrutura sequêncial. Permite exprimir todo o tipo de mecanismos de execução que tenham estrutura sequêncial. De forma a entender rapidamente os* mónadas *segue a seguinte*

analogia. Os mónadas *são caixas que contêm valores, esses valores apenas podem ser tratados por funções que consigam abrir essas caixas e posteriormente encaixota-los novamente. Este conceito é fundamental para entender o funcionamento da biblioteca* Lwt *(2.2.6), pois a sua funcionalidade é conseguida exclusivamente por meio de uma estrutura monádica. Uma estrutura monádica nas linguagens funcionais é na prática um estrutura que implementa, na sua forma mais canónica, duas funções:*

- return : 'a -> 'a t = <fun>. *A função* return *recebe um valor e retorna-o no formato encapsulado* 'a t.
- bind : 'a t -> ('a -> 'b t) -> 'b t = <fun>. *A função* bind*, ou muitas vezes definida com o operador* »= *recebe dois parâmetros, um valor encapsulado* 'a t *e uma função* ('a -> 'b t)*. Internamente a função* bind *vai desencapsular o primeiro parâmetro e aplica-lo ao segundo parâmetro, a função, que por sua vez vai retornar um novo valor encapsulado, do mesmo tipo* t.

Em OCaml um mónada *pode ser definido, através de um módulo que, em primeiro lugar vai definir o que é o tipo e o que ele representa. Em segundo lugar definir e implementar o comportamento das funções descritas anteriormente, para o tipo definido criando desta forma um* mónada*. No exemplo seguinte retirado de [?] é apresentado um exemplo, de implementação de um* mónada*, que na sua definição, apresenta também uma função que permite desencapsular o valor do próprio* mónada.

```
module Maybemónada = struct
  type 'a t = None | Maybe of 'a
  let return (a: 'a) : 'a t = Maybe a
  let (>>=) (m: 'a t) (f: 'a -> 'b t) : 'b t = match m with
    | None -> None
    | Maybe a -> f a
    let get (m: 'a t) (a: 'a) = match m with
    | None -> a
    | Maybe a -> a
end

let sum x_opt y_opt z_opt = let open Maybemónada in
  let sum_opt =
    x_opt >>= fun a ->
    y_opt >>= fun b ->
    z_opt >>= fun c ->
    return (a + b + c + 2) in
  get sum_opt 0

let s = sum (Maybemónada.Maybe 9) (Maybemónada.Maybe 10) (Maybemónada.Maybe 11)
let _ = assert (s = 32)
let s = sum (Maybemónada.Maybe 9) (Maybemónada.Maybe 10) Maybemónada.None
let _ = assert (s = 0)
```

Listing 2.10: Exemplo do uso de um *mónada*.

2.3.5 Continuation Passing Style (CPS)

O estilo de programação por continuação ou CPS é um estilo de programação, que tem como lógica compartimentar o resultado de funções, por meio da aplicação de outras funções. Desta forma garante-se o correto tratamento, dentro do contexto aplicacional e a consecutivamente a sua continuação podendo ter ou não, comportamento assíncrono. De forma simples, uma continuação é o uso do resultado de uma função α *por uma função* β*. Este estilo de programação é amplamente utilizado, porém este jargão é normalmente conhecido por* callbacks*, bastante usadas em* JavaScript *e também por programação por eventos utilizado maioritariamente em interfaces gráficas. Na realidade estes últimos são subconjuntos do estilo de programação por continuação. No contexto deste projeto é fundamental compreender este conceito, pois a plataforma* Ocsigen *é construída sobre ele, em especial o módulo* Eliom*(2.2.7), que é fundamental para desenvolver aplicações dinâmicas na plataforma* Ocsigen*. Seguem agora dois exemplos em OCaml, os quais pretendem clarificar este estilo de programação comparando o mesmo exemplo, sem este estilo. O exemplo (2.11) apresenta a função* soma5 *a qual recebe dois parâmetros, o v1 e o v2, que vai subtrair e posteriormente somar cinco unidades.*

```
# let soma5 v1 v2 = (v1-v2)+5;;
val soma5 : int -> int -> int = <fun>
# soma5 2 2;;
- : int = 5
```

Listing 2.11: Função *soma5* sem CPS.

O exemplo (2.12) apresenta a mesma função soma5*, no entanto agora aparece uma parâmetro novo, que é a função que vai receber o valor da operação de subtração de entre* v1 *e* v2*. Esta nova função permite continuar a computação de forma a obter o resultado esperado. Note que a função* cps_fun *está a somar cinco unidades ao valor recebido por parâmetro, no entanto pode ser alterado para somar um valor diferente de cinco, sem que seja necessário alterar a função* soma5*. Isto permite que o código seja escalável evitando possíveis efeitos colaterais, mantendo a função como principio único e imutável representado desta forma, o paradigma funcional.*

```
# let cps_fun a = (+) a 5;;
val cps_fun : int -> int -> int = <fun>
# let soma5 v1 v2 cps = cps ((-) v1 v2) ;;
val soma5 : 'a -> 'b -> ('a -> 'b -> 'c) -> 'c = <fun>
# soma5 2 2 cps_fun;;
- : int = 5
```

Listing 2.12: Função *soma5* com CPS.

2.4 Trabalhos Relacionados

Esta secção tem como objetivo justificar a utilização do ecossistema Ocsigen. *Visto não existirem propriamente trabalhos semelhantes ao desenvolvido neste documento, em relação ao ecossistema* Ocsigen. *Este será comparado com outra plataforma mais difundida e amplamente utilizada pela comunidade de desenvolvimento WEB. A tecnologia em questão é a plataforma* NodeJS *[nod]. Muito à semelhança da plataforma* Ocsigen, *o* NodeJS *foi criado para dar resposta ao desenvolvimento WEB. O* NodeJS *possibilita que o código produzido possa ser escrito numa só linguagem, neste caso, em* JavaScript. *No entanto o código criado do lado servidor apresenta uma diferença substancial no momento da compilação. O código produzido com* NodeJS *é compilado apenas para bytecode e executado pela máquina virtual* Chrome's V8 JavaScript engine *[v8]. Já na plataforma* Ocsigen, *a compilação para* bytecode *é opcional, porque existe a possibilidade de compilar o código do lado servidor diretamente para código máquina.*

Figura 2.10: Logótipo do NodeJS
Fonte: `https://nodejs.org/en/`.

Esta diferença ao nível de desempenho é considerável, pois em condições normais, o código máquina é normalmente mais rápido que o bytecode *executado por uma máquina virtual. Outra diferença está na segurança proporcionada pela linguagem de programação. Comparando OCaml com* JavaScript *rapidamente se verifica que a tipificação dinâmica do* JavaScript *faz com que erros de tipo disfarçados na lógica do código gerem exceções em tempo de execução. Apesar de ao nível da segurança e de desempenho, a plataforma* Ocsigen *estar melhor assegurada. O desenvolvimento na plataforma* NodeJS *é bastante mais rápido e prático, devido a curva de aprendizagem ser muito menor relativamente à que existe na plataforma* Ocsigen. *Ora este trabalho visa resolver esse facto.*

2.5 Conclusão

No presente capítulo são abordadas as tecnologias envolvidas. Estas são descritas e contextualizadas com o projeto desenvolvido. São também apresentados os conceitos teóricos, essenciais para o entendimento do desenvolvimento da solução. Por fim é apresentada uma comparação, com uma tecnologia bastante utilizada e conhecida. É então, com base em toda a informação apresentada, que é agora possível definir uma análise centrada na planificação da solução. Desta forma, o próximo capítulo (3), apresenta a análise e arquitetura da solução desenvolvida no âmbito desta dissertação.

Capítulo 3

Análise e Arquitetura da Solução

3.1 Introdução

No primeiro capítulo (1) foi abordada a definição do problema (1.2). Esta foi detalhada ao longo segundo capítulo (2) e relacionada com as tecnologias envolvidas. Depois de uma análise profunda sobre as tecnologias em causa e a sua base teórica torna-se agora imperativo elaborar um plano, que seja capaz de dar resposta aos problemas citados 1.2. O objetivo deste capítulo é por meio de uma análise de requisitos e casos de uso apresentar uma solução formal e correta, decorrente um processo típico de engenharia de software. Esta análise aproxima-se dos padrões definidos pela IEEE *no âmbito da engenharia de software, [71015], [57310] e [72098], [15990].*

3.2 Objetivo

Com base no estudo desenvolvido nos capítulos anteriores, o objetivo deste capítulo é apresentar de uma forma analítica, a solução para os problemas referidos no capítulo 1 (1.2). Esta solução, passa pela criação de uma nova framework *designada por* Ocsi_Simple framework. *Esta irá ser composta por três módulos, os quais estão direcionados a áreas especificas do desenvolvimento WEB. De forma a esquematizar cada um deles, segue a seguinte lista:*

1. **Ocsi_simple_persist** *é o módulo que irá tratar da persistência de dados. Pretende-se que assegure o acesso, aos motores de base de dados* Mysql, PostgreSQL *e* SQ-Lite3, *de forma a facilitar a sua utilização. Pretende-se também que, para o seu uso seja apenas necessário configurar a ligação do motor de base de dados pretendido. Este módulo terá que ser escalável, de forma a poder permitir a implementação, de outros motores de base de dados.*

2. **Ocsi_simple** *é o módulo que permite, de uma forma simples, concisa, estruturada e limpa criar qualquer tipo de página HTML, sintaticamente correta. Pretende-se que este módulo quando utilizado possa influenciar positivamente a produtividade e a organização de desenvolvimento.*

3. **Ocsi_simple_js** *é o módulo que irá implementar várias funções, que têm como objetivo replicar o comportamento de funções* JavaScript *habitualmente utilizadas, pela grande maioria dos programadores WEB. Este módulo terá que permitir de uma forma simples manipular elementos em* front-end. *Assim como possibilitar a inclusão de novas funções.*

A figura seguinte (A.37) ilustra a arquitetura da solução, a desenvolver. Nas próximas secções esta irá ser analisada de forma estruturada, no âmbito dos seus requisitos.

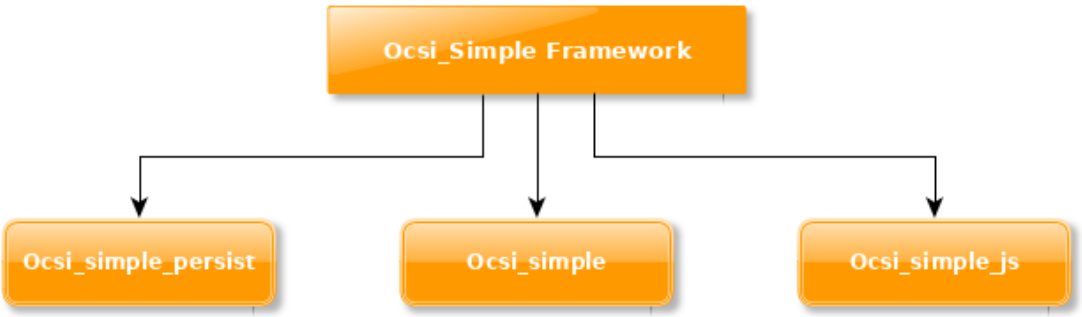


Figura 3.1: Ocsi_Simple Framework.

3.3 Assunções

É necessário ter em consideração alguns pontos fundamentais para a correta utilização da Ocsi_Simple Framework. *É esperado que o utilizador da* Ocsi_Simple Framework *tenha presente, os conceitos teóricos apresentados no segundo capítulo (2.3). É também esperado que o utilizador tenha conhecimentos, ao nível do paradigma de programação funcional, desenvolvimento WEB e utilização de sistemas de base de dados. O conceito* "Lado Cliente" *é utilizado para identificar código, que durante o processo de compilação vai dar origem a HTML e* JavaScript. *O conceito* "Lado Servidor" *é utilizado para identificar código, que durante o processo de compilação vai dar origem a código OCaml, posteriormente compilado para* bytecode *ou código máquina. O utilizador da* Ocsi_Simple Framework *é considerado o ator do sistema.*

3.4 Requisitos Funcionais

3.4.1 Requisitos Funcionais de Utilizador

1. *Será possível ao utilizador da* Ocsi_Simple Framework *desenvolver páginas HTML estáticas.*
2. *Será possível ao utilizador da* Ocsi_Simple Framework *desenvolver páginas HTML dinâmicas.*
3. *Será possível ao utilizador da* Ocsi_Simple Framework *desenvolver aplicações WEB.*
4. *Será possível ao utilizador da* Ocsi_Simple Framework *manipular elementos de páginas HTML.*
5. *Será possível ao utilizador da* Ocsi_Simple Framework *criar e utilizar ligações para motores de base de dados.*

3.5 Requisitos Não-Funcionais

3.5.1 Requisitos de Produto

3.5.1.1 Usabilidade

1. *O utilizador da* Ocsi_Simple Framework *deverá conseguir utilizar o módulo* Ocsi_simple_persist*, depois de ler o manual*.
2. *O utilizador da* Ocsi_Simple Framework *deverá conseguir utilizar o módulo* Ocsi_simple*, depois de ler o manual*.
3. *O utilizador da* Ocsi_Simple Framework *deverá conseguir utilizar o módulo* Ocsi_simple_js*, depois de ler o manual*.
4. *O utilizador da* Ocsi_Simple Framework *deverá ser capaz de utilizar o motor de base de dados* Postgreql*, de forma simples e transparente*.
5. *O utilizador da* Ocsi_Simple Framework *deverá ser capaz de utilizar o motor de base de dados* Mysql*, de forma simples e transparente*.
6. *O utilizador da* Ocsi_Simple Framework *deverá ser capaz de utilizar o motor de base de dados* SQLite3*, de forma simples e transparente*.
7. *O utilizador da* Ocsi_Simple Framework *deverá ser capaz de manipular elementos elementos* front-end*, de forma simples e transparente*.

3.5.1.2 Modularidade e Reutilização

1. *O código criado, deverá ser modular*.
2. *O código criado, deverá ser garantir a sua reutilização.*

3.5.1.3 Fiabilidade

1. *Todos os dados provenientes, tanto do* Front-end *como do* Back-end *deverão ser analisados na sua integridade, antes da sua importação*.
2. *A* Ocsi_Simple Framework *deverá garantir total integridade referencial*.

3.5.1.4 Segurança

1. *O acesso aos motores de base de dados deverá ser compartimentado, apenas à função que o utiliza*.
2. *A utilização de dados que digam respeito a conteúdo de utilizadores deverão ser tratados pelo servidor*.

3.5.2 Requisitos Organizacionais

3.5.2.1 Desenvolvimento

1. *O código deverá ser simples, de modo a que qualquer programador de* Software *o consiga ler e entende-lo facilmente.*
2. *A linguagem de programação a ser usada será OCaml.*
3. *O motor de base de dados a utilizar poderá ser* Mysql.
4. *O motor de base de dados a utilizar poderá ser* PostgreSQL.
5. *O motor de base de dados a utilizar poderá ser* SQLite3.

3.5.2.2 Documentação

1. *A* Ocsi_Simple Framework *será acompanhada pelo manual de utilizador e respetiva documentação técnica.*

3.6 Diagramas de Casos de Uso

Os diagramas de casos de uso apresentados de seguida retratam de forma geral, a utilização da Framework Ocsi_simple. *No anexo A, os diagramas de casos de uso são apresentados de forma exaustiva.*

3.6.1 Utilizador

3.6.1.1 Criar elementos de marcação HTML

Referente aos pontos 1, 2 e 3 de 3.4.1.

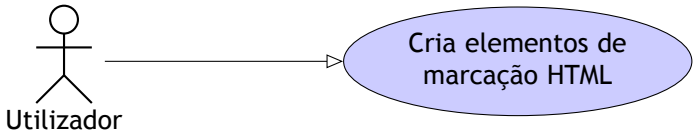


Figura 3.2: Caso de uso sobre criar elementos de marcação HTML.

Ator: *O utilizador da* Framework Ocsi_simple.
Descrição: *O utilizador da* Framework Ocsi_simple *poderá de forma simples e intuitiva, através do módulo* Ocsi_simple *criar qualquer página HTML. Para isso terá que utilizar devidamente os elementos de marcação presentes neste e instancia-los corretamente com as ligações aos ficheiros* CSS *e aos ficheiros* JavaScript*, casos os haja. Mais detalhes podem ser vistos no anexo A (A.2.1).*
Estímulo: *O utilizador usa o módulo* Ocsi_simple*, para cria elementos de marcação HTML.*
Comentários: *O utilizador deverá conhecer a* Framework Ocsi_simple.

3.6.2 Lado Cliente

3.6.2.1 Manipular elementos de marcação da página HTML

Referente ao ponto 4 de 3.4.1.

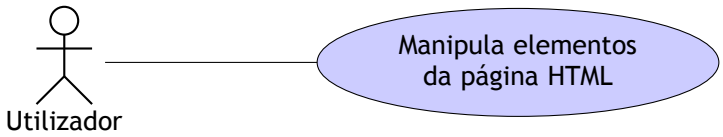


Figura 3.3: Caso de uso sobre manipular elementos da página HTML.

Ator: *O utilizador da* Framework Ocsi_simple.

Descrição: *O utilizador da* Framework Ocsi_simple *poderá manipular elementos da página HTML. Para isso terá que utilizar devidamente as funções presentes no módulo* Ocsi_simple_js. *De forma mais exaustiva, no anexo A (A.2.2), encontram-se os casos de uso para este módulo.*

Estímulo: *O utilizador usa as funções presentes no* Ocsi_simple_js *para manipula elementos da página HTML.*

Comentários: *O utilizador deverá conhecer a* Framework Ocsi_simple.

3.6.3 Lado Servidor

3.6.3.1 Usar os SGBD *Mysql*, *PostgreSQL* e *SQLite*

Referente aos pontos 5 de 3.4.1.

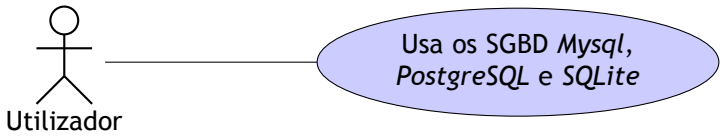


Figura 3.4: Caso de uso sobre usar os SGBD *Mysql*, *PostgreSQL* e *SQLite*.

Ator: *O utilizador da* Framework Ocsi_simple.

Descrição: *O utilizador da* Framework Ocsi_simple *poderá criar ligações para os SGBD* Mysql, PostgreSQL *e* SQLite*, através da criação de uma instância do módulo correspondente ao SGBD pretendido. Através desta instância é possível obter, atualizar, eliminar e criar novos dados sempre que necessário. Mais detalhes sobre a sua utilização encontram-se no anexo A (A.2.3).*

Estímulo: *O utilizador instancia o módulo pretendido, para criar a ligação e usa para o SGBD pretendido.*

Comentários: *O utilizador deverá conhecer a* Framework Ocsi_simple.

3.7 Estrutura do Modelo de Persistência

A estrutura do modelo de persistência será composta por três módulos que implementam uma interface *denominada por* Imdb. *A* interface Imdb *define as funções responsáveis por interagir com o SGBD e manipular os dados obtidos deste. Os três módulos implementam as funcionalidades para os SGBD* Mysql, PostgreSQL *e* SQLite. *É esperado que o utilizador apenas tenha que instanciar a ligação ao SGBD com o nome do SGBD pretendido e a respetiva configuração de acesso. Desta forma o seguinte trecho de código (3.1) apresenta a* interface Imdb.

```
type t = {
        host:string;
        database:string;
        port:int;
        user:string;
        password:string;
    }

module type Imdb =
sig
    exception Ok of string
    exception Error of string

    type h

    val connection: h

    val exec_cdu: query:string -> unit

    val exec_select: query:string -> string list list

    val show_exec_select: string list list -> unit list
end
```

Listing 3.1: Definição da *interface Imdb*.

3.8 Conclusão

Após a abordagem sobre a análise formal ao nível dos requisitos funcionais e não funcionais e seus respetivos casos de uso segue a para a próxima etapa lógica, a implementação. No próximo capítulo (4) será abordada a implementação da Framework Ocsi_simple. *Será também abordada a construção de duas provas de conceito, com e sem recurso à* Framework Ocsi_simple, *que culminam na validação do trabalho em causa.*

Capítulo 4

Implementação e Validação

4.1 Introdução

O presente capítulo começa por abordar a implementação da Framework Ocsi_simple*, através da apresentação das principais funções de cada módulo. Estas têm por base, a análise efetuada no capítulo 3. De seguida são apresentados dois casos de prova de conceito sobre o mesmo exemplo. Estes têm como objetivo, apresentar a implementação utilizando a forma padrão da tecnologia* Ocsigen *e a nova solução desenvolvida. Por fim é apresentada uma análise, que visa validar os casos desenvolvidos.*

4.2 Implementação

A implementação da Framework Ocsi_simple *deriva da análise apresentada nos capítulos anteriores. Esta teve como objetivo encontrar uma solução formal, prática e intuitiva para o desenvolvimento WEB, através da plataforma* Ocsigen*. A solução encontrada e desenvolvida responde diretamente à pergunta feita no capítulo 1,* "O que se pode fazer, para tornar a *framework Ocsigen* mais simples de utilizar ?" *(1). A qual culmina na* Framework Ocsi_simple*. Mais uma vez é referido que a solução desenvolvida visa complementar a tecnologia* Ocsigen*. Desta forma, na presente secção são apresentados os casos de implementação mais importantes, de forma a dar ao leitor uma visão sobre a construção da solução.*

4.2.1 Parte Cliente

4.2.1.1 Implementação da função *webpage*

Referente aos pontos 1, 2 e 3 de 3.4.1.

```
let webpage ?(p_title = "Simple Ocsigen") ?(p_meta = s_meta ())
?(p_css = css_links () ) ?(p_js = js_links () ) ~p_body () =
Eliom_content.Html.D.(
      (html
        (head
          (title (pcdata p_title))
          (p_meta@p_css@p_js) )
        (p_body)
      )
    )
```

Listing 4.1: Implementações da função *webpage*.

A função webpage *(4.1), pertence ao módulo* Ocsi_simple *e tem como objetivo a criação uma página HTML. Está desenhada para ser de produtiva garantindo a construção de uma página HTML com um código limpo, prático e organizado com tudo o que é necessário no desenvolvimento WEB. Apenas os parâmetros que são realmente necessários são introduzidos, como é o caso do título da página, dos links Cascading Style Sheets (CSS), JavaScript (JS) e* meta tags. *A função* webpage *recebe como parâmetro obrigatório o* p_body*, que trata de toda a construção relativa ao corpo da página HTML.*

4.2.1.2 Implementação da função *body_page*

Referente aos pontos 1, 2 e 3 de 3.4.1.

```
let body_page ?(css_id="") ?(css_class="") ?(content=[]) () =
    Eliom_content.Html.D.body ~a:[a_id css_id; a_class [css_class]] content
```

Listing 4.2: Implementações da função *body_page*.

A função body_page *(4.2) pertence ao módulo* Ocsi_simple *e tem como objetivo, a criação do corpo da página HTML. Está desenhada para que quando utilizada garanta a construção de um código limpo, prático e organizado. Apenas os parâmetros que são realmente necessários são introduzidos.*

4.2.1.3 Implementação da função *s_div*

Referente aos pontos 1, 2 e 3 de 3.4.1.

```
let s_div ?(css_id="") ?(css_class="") ?(style="") ~(content:'a list) () =
    Eliom_content.Html.D.div ~a:[ a_id css_id; a_style style;
    a_class [css_class] ] content
```

Listing 4.3: Implementações da função *s_div*.

A função s_div *(4.3) pertence ao módulo* Ocsi_simple *e tem como objetivo a criação de um elemento* div *pertencente ao corpo da página HTML. Está desenhada para que quando utilizada possa garantir a construção de um código limpo, prático e organizado. Apenas os parâmetros que são realmente necessários são introduzidos.*

4.2.1.4 Implementação da função *appendTo*

Referente ao ponto 4 de 3.4.1 da análise de requisitos 3.

```
let s_appendTo base_id block_id = Eliom_content.Html.Manip.appendChild
      (Eliom_content.Html.Of_dom.of_element (get_element_by_id base_id))
      (Eliom_content.Html.Of_dom.of_element (get_element_by_id block_id))
```

Listing 4.4: Implementações da função *appendTo*.

A função appendTo *(4.4) pertence ao módulo* Ocsi_simple_js *e tem como objetivo a adição de um elemento HTML a outro elemento HTML, através do seu* id. *Está desenhada para ser de prática utilização. Tem como parâmetros, o* id *do elemento base e o* id *do elemento que se pretende adicionar ao elemento base.*

4.2.2 Parte Servidor

4.2.2.1 Implementação do módulo *Ocsi_simple_persist*

Referente ao ponto 5 de 3.4.1 da análise de requisitos 3.

```
open Mdb_mysql
open Mdb_postgresql
open Mdb_sqlite
open Imdb

type db = Postgresql | Mysql | Sqlite3

module Make (DB: sig type t val db_type : db val connection_data : t end with type t = Imdb.t )=
struct

    exception Ok of string
    exception Error of string

    let m_list = match DB.db_type with
            |Postgresql -> (module (Mdb_postgresql.Make(struct type t = Imdb.t let connection_data = DB.connection_data  end) ) :Imdb)
            |Mysql -> (module (Mdb_mysql.Make(struct type t = Imdb.t let connection_data = DB.connection_data  end) ) :Imdb)
            |Sqlite3 -> (module (Mdb_sqlite.Make(struct type t = Imdb.t let connection_data = DB.connection_data  end)) :Imdb)

    let exec_select ~query =
        let module M = (val  m_list:Imdb) in
            try
                M.exec_select query
            with M.Error s  -> raise (Error s)

    let show_exec_select result =
        let module M = (val  m_list:Imdb) in
            try
                M.show_exec_select result
            with M.Error s -> raise (Error s)

    let exec_cdu ~query =
        let module M = (val  m_list:Imdb) in
            try
                M.exec_cdu query
            with
                | M.Error s ->  raise (Error s)
                | M.Ok s -> raise (Ok s)
end
```

Listing 4.5: Implementação do módulo *Ocsi_simple_persist*.

A implementação do módulo Ocsi_simple_persist *(4.5) apresenta vários conceitos teóricos expostos no segundo capítulo (2.3). Este módulo começa por abrir os módulos* Mdb_mysql, Mdb_postgresql *e* Mdb_sqlite *que implementam a* interface Imdb. *Esta define a estrutura que as implementações devem seguir. É também aberta a própria* interface Imdb *de forma a identificar os tipos necessários para criar o* funtor Make. *Este vai ser instanciado um módulo, que contempla o respetivo SGBD identificado através do tipo* db *e a estrutura de ligação ao SGBD de tipo* t, *apresentado abaixo (4.6).*

```
type t = {
        host:string;
        database:string;
        port:int;
        user:string;
        password:string;
    }
```

Listing 4.6: Definição do tipo *t* na *interface Imdb*.

Consoante o tipo de SGBD escolhido é criado um módulo de primeira classe que vai ser utilizado na definição das três funções que interagem com o SGBD. A função exec_select *recebe como parâmetro uma* query *de seleção de dados, os dados obtidos vão ser organizados dentro de uma lista, cujas linhas obtidas da seleção são também listas. Na prática a função* exec_select *devolve uma lista de listas. Para apresentar esta lista de listas existe a função* show_exec_select, *que recebe como parâmetro o resultado da função* exec_select. *Esta é uma função auxiliar pois apenas apresenta o resultado obtido da função* exec_select. *É importante notar que caso haja algum erro, por exemplo de sintaxe nas* queries *ou de ligação, estes podem ser devidamente tratados. Por fim a função* exec_cdu *é definida para tratar de* query *que apenas efetuem ações de atualização eliminação ou inserção. Desta forma o acesso aos SGBD pretendidos, pode ser feita de forma simples, rápida e segura.*

4.3 Ocsigen no Windows

Atualmente existem poucas alternativas para instalar o ecossistema OCaml no sistema operativo Windows. *Apesar do compilador de OCaml ter a possibilidade de ser compilado e executado no Sistema Operativo (SO)* Windows*, através de mecanismos nativos [git] ou emulados, como por exemplo recorrendo ao* Cygwin *[cyg], o seu gestor de pacotes* OPAM *ainda não é suportado na integra. Existe uma promessa na comunidade OCaml, que tem vindo a ser desenvolvida para tornar* OPAM *nativo em* Windows*, o* OPAM 2.0 *[opab]. No entanto as metas temporais têm sido consecutivamente ultrapassadas. Porém existem implementações paralelas. Um caso de sucesso é solução desenvolvida por Andreas Hauptmann [opaa], que criou um repositório paralelo de pacotes do ecossistema OCaml preparados para serem compilados de forma nativa em* Windows *recorrendo para isso ao* Cygwin *juntamente com o* MinGW *[min]. É uma solução engenhosa porém não é centralizada e os pacotes oficiais terão que ser sempre alterados um a um para permitirem, um processo de compilação correto em* Windows *sendo de facto uma limitação considerável. No entanto com os últimos avanços da* Microsoft *[mic], em trazer várias tecnologias do mundo* Unix *para o* Windows *existe agora uma possibilidade real, de trabalhar de forma prática com ecossistema OCaml. Essa possibilidade traduz-se na WSL [winc], que possibilita trabalhar sobre um sistema* Linux *dentro do SO* Windows. *Desta forma apesar da WSL não ser uma solução direta é aquela que proporciona maior estabilidade durante a sua utilização. Através da WSL é possível usar o ecossistema* OCaml *sem reservas. Desta forma é apresentado o anexo B* Ocsigen no Windows*, que tem como objetivo principal demonstrar como é possível usar a WSL no desenvolvimento com a tecnologia* Ocsigen *respondendo desta forma à pergunta 2 - O que se pode fazer, para tornar a tecnologia* Ocsigen *utilizável em* Windows *?. É assim provado desta forma, que é possível utilizar o ecossistema OCaml no sistema operativo* Windows.

4.4 Prova de Conceito 1

A prova de conceito aqui apresentada utiliza a forma de desenvolvimento nativa da tecnologia Ocsigen. *Todo o código apresentado pode ser consultado em* `http://ocsisimple.ddns.net:8085`. *A imagem (4.1) apresenta o resultado obtido.*

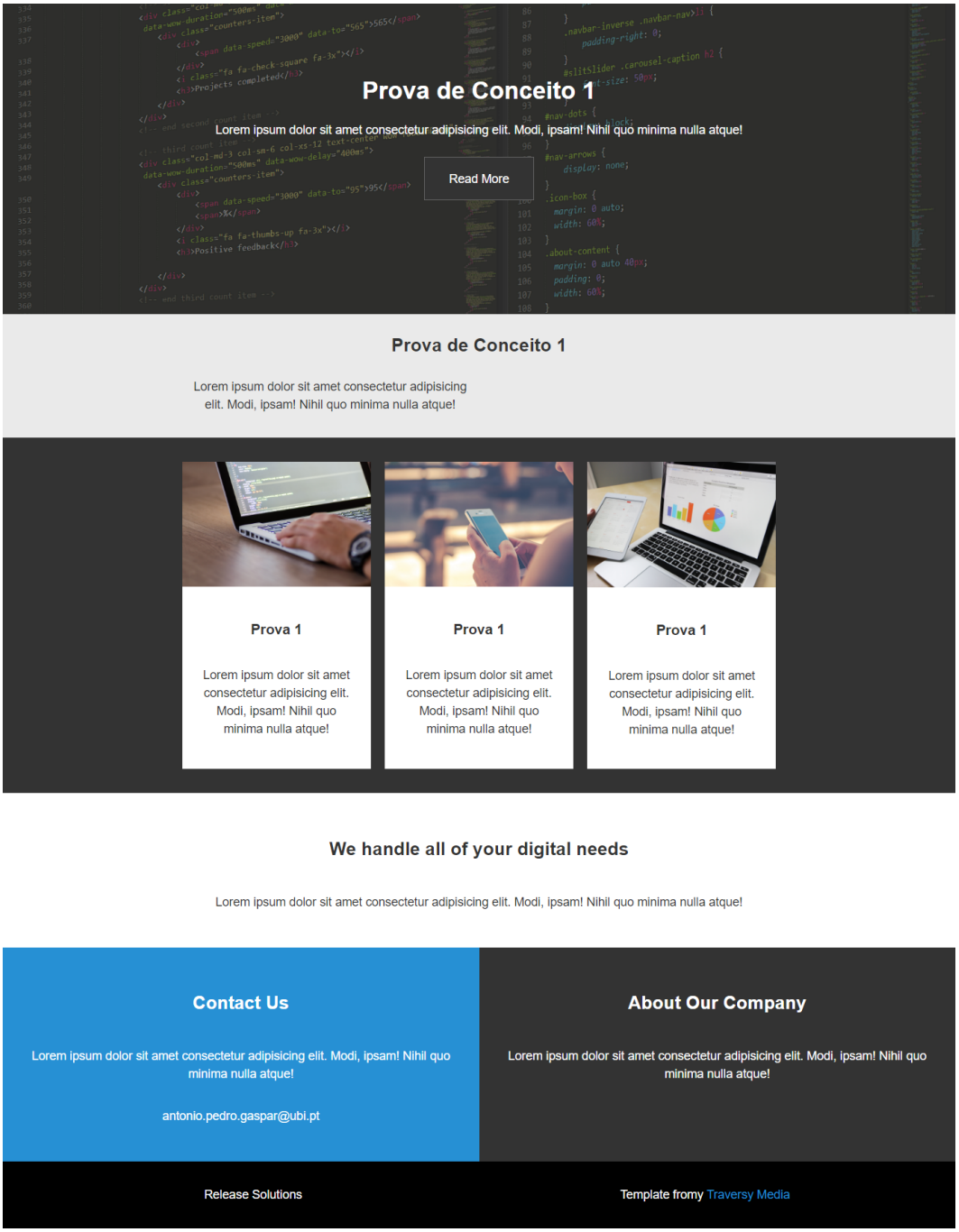


Figura 4.1: Resultado da prova de conceito 1.

Como se pode verificar, a imagem (4.1) apresenta uma página HTML. Esta é composta por sete partes, as quais são analisadas de seguida. A figura (4.2) apresenta a primeira parte da implementação da prova de conceito. Como se pode verificar a partir a da linha 25, é definida a construção da página HTML.

No primeiro bloco situado entre as linhas 26 e 35 são definidas as ligações e configurações que a página HTML vai assumir. Depois da linha 35 é definido o corpo da página. Este é composto pelas restantes seis partes. A parte entre as linhas 35 e 47 define o cabeçalho da página HTML.

```
1  [%%shared
2      open Eliom_lib
3      open Eliom_content
4      open Html.D
5  ]
6
7  module Site2_app =
8    Eliom_registration.App (
9      struct
10       let application_name = "site2"
11       let global_data_path = None
12     end)
13
14 let main_service =
15   Eliom_service.create
16     ~path:(Eliom_service.Path [])
17     ~meth:(Eliom_service.Get Eliom_parameter.unit)
18     ()
19
20 let () =
21   Eliom_content.Html.D.
22   (Site2_app.register
23     ~service:main_service
24     (fun () () ->
25       Lwt.return
26         (html
27           (head
28             (title (pcdata "Prova 1"))
29             [
30               meta ~a:[ a_charset "UTF-8"; a_name "viewport"; a_content "width=device-width, initial-scale=1"]();
31               css_link ~uri:(make_uri (Eliom_service.static_dir ()) ["css"; "site2.css"]) ();
32               css_link ~uri:(Raw.uri_of_string "https://use.fontawesome.com/releases/v5.0.10/css/all.css") ();
33             ]
34           )
35           (body [
36 (*Header ShowCase*) header ~a:[ a_id "showcase"; a_class ["grid"] ]
37                 [
38                   div ~a:[ a_class ["bg-image"] ] [];
39                   div ~a:[ a_class ["content-wrap"] ]
40                   [
41                     h1 [pcdata "Prova de Conceito 1"];
42                     p [pcdata "Lorem ipsum dolor sit amet consectetur
43                                adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!"
44                       ];
45                     Raw.a ~a:[a_href (Raw.uri_of_string "#"); a_class ["btn"]] [pcdata "Read More"];
46                   ];
47                 ];
```

Figura 4.2: Código da prova de conceito 1 (Parte 1).

De seguida, como apresentado na imagem (4.3) entre as linhas 48 e 61 é definida a segunda estrutura, que define a delimitação entre o cabeçalho e a terceira estrutura.

```
48                 (*MAIN AREA*) main ~a:[ a_class ["main"]]
49                   [
50                     section ~a:[ a_id "section-a"; a_class ["grid"] ]
51                     [
52                     div ~a:[ a_class ["content-wrap"] ][
53                       h2 ~a:[a_class ["content-title"]] [pcdata "Prova de Conceito 1"];
54                       div ~a:[ a_class ["content-text"] ]
55                       [
56                         p [pcdata "Lorem ipsum dolor sit amet consectetur
57                                    adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!"
58                           ];
59                       ]
60                     ];
61                     ];
```

Figura 4.3: Código da prova de conceito 1 (Parte 2).

A imagem (4.4) apresenta a terceira estrutura da página HTML, a qual é composta por uma lista com três linhas formatadas de forma horizontal. Estas por sua vez definem novas estruturas que estão inseridas dentro da lista principal.

```
(*Section B*) section ~a:[ a_id "section-b"; a_class ["grid"] ]
                [
                  ul[
                    li [
                      div ~a:[ a_class ["card"] ]
                      [
                        img ~alt:("teste")
                          ~src:(Raw.uri_of_string "https://static.pexels.com/photos/574077/pexels-photo-574077.jpeg")();
                        div ~a:[ a_class ["card-content"] ]
                        [
                          h3 ~a:[a_class ["content-title"]] [pcdata "Prova 1"];
                          p [pcdata "Lorem ipsum dolor sit amet consectetur
                                  adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!"
                            ];
                        ];
                      ]
                    ];
                    li [
                      div ~a:[ a_class ["card"] ]
                      [
                        img ~alt:("teste")
                          ~src:(Raw.uri_of_string "https://static.pexels.com/photos/261628/pexels-photo-261628.jpeg")();
                        div ~a:[ a_class ["card-content"] ]
                        [
                          h3 ~a:[a_class ["content-title"]] [pcdata "Prova 1"];
                          p [pcdata "Lorem ipsum dolor sit amet consectetur
                                  adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!"
                            ];
                        ];
                      ]
                    ];
                    li [
                      div ~a:[ a_class ["card"] ]
                      [
                        img ~alt:("teste")
                          ~src:(Raw.uri_of_string "https://static.pexels.com/photos/265087/pexels-photo-265087.jpeg")();
                        div ~a:[ a_class ["card-content"] ]
                        [
                          h3 ~a:[a_class ["content-title"]] [pcdata "Prova 1"];
                          p [pcdata "Lorem ipsum dolor sit amet consectetur
                                  adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!"
                            ];
                        ];
                      ]
                    ];
                  ];
                ];
```

Figura 4.4: Código da prova de conceito 1 (Parte 3).

A ultima parte corresponde às três ultimas estruturas.

```

(*Section C*) section ~a:[ a_id "section-c"; a_class ["grid"] ]
                [
                  div ~a:[ a_class ["content-wrap"] ]
                  [
                    h2 ~a:[a_class ["content-title"]] [pcdata "We handle all of your digital needs"];
                    p [pcdata "Lorem ipsum dolor sit amet consectetur
                            adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!"
                      ];
                  ];
                ];

(*Section D*) section ~a:[ a_id "section-d"; a_class ["grid"] ]
                [
                  div ~a:[ a_class ["box"] ]
                  [
                    h2 ~a:[a_class ["content-title"]] [pcdata "Contact Us"];
                    p [pcdata "Lorem ipsum dolor sit amet consectetur
                            adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!"
                      ];
                    p [pcdata "antonio.pedro.gaspar@ubi.pt"];
                  ];
                  div ~a:[ a_class ["box"] ]
                  [
                    h2 ~a:[a_class ["content-title"]] [pcdata "About Our Company"];
                    p [pcdata "Lorem ipsum dolor sit amet consectetur
                            adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!"
                      ];
                  ];
                ];

(* Footer *) footer ~a:[ a_id "main-footer"; a_class ["grid"]]
                [
                  div [pcdata "Release Solutions"];
                  div
                  [
                    pcdata "Template fromy ";
                    Raw.a ~a:[a_href (Raw.uri_of_string "http://traversymedia.com"); a_target "_blank"][pcdata "Traversy Media"];
                  ];
                ];
              ];
            ])
          )
        )
      )
```

Figura 4.5: Código da prova de conceito 1 (Parte 4).

4.5 Prova de Conceito 2

A prova de conceito aqui apresentada utiliza a nova solução desenvolvida Framework Ocsi_simple. *Todo o código apresentado pode ser consultado em* `http://ocsisimple.ddns.net:8086`. *A imagem (4.6) apresenta o resultado obtido. As diferenças com primeira prova de conceito são notórias no entanto essa verificação será efetuada na secção seguinte.*

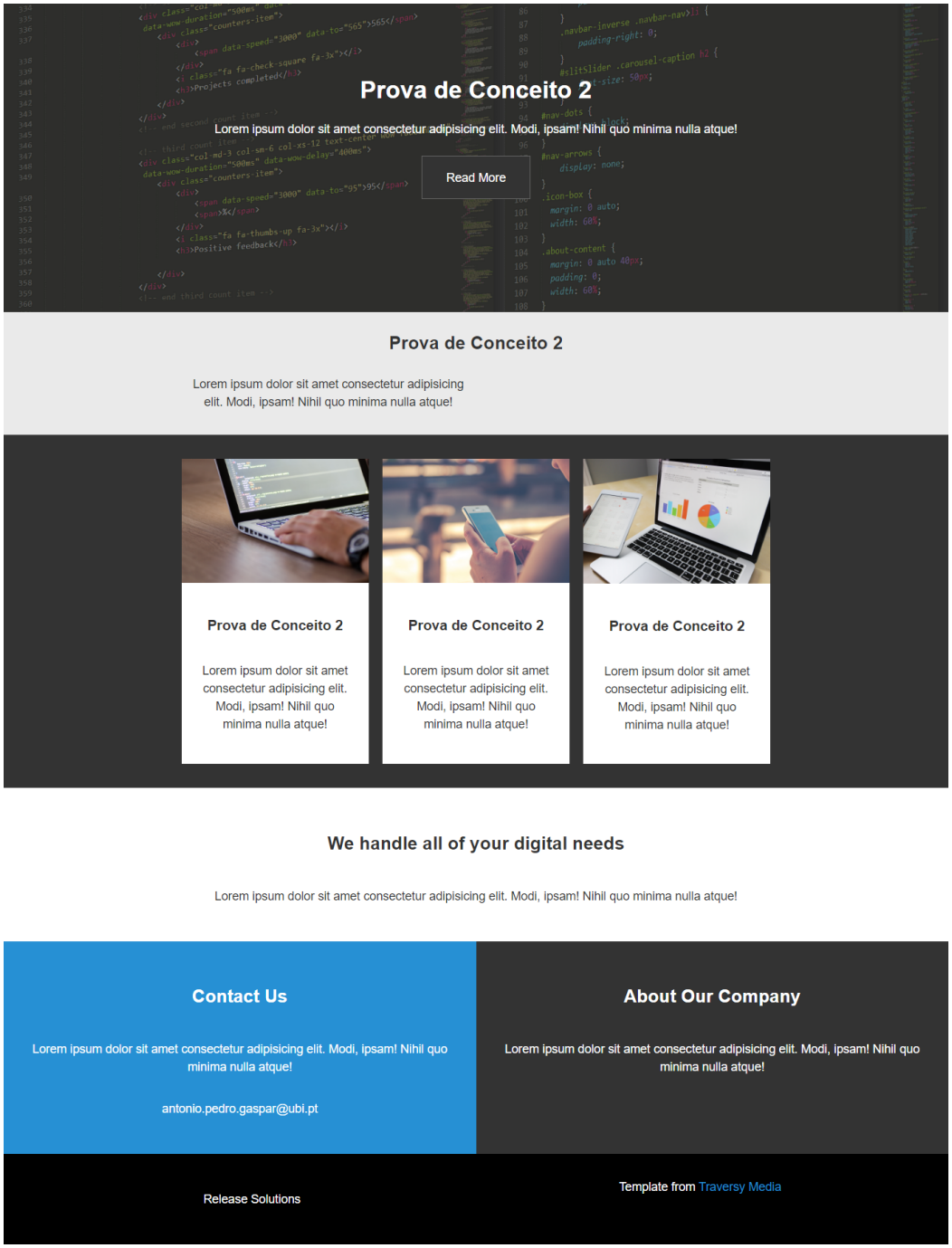


Figura 4.6: Resultado da prova de conceito 2.

Como se pode verificar, a imagem (4.6) apresenta uma página HTML. Esta é composta por sete partes, as quais são analisadas de seguida.

A figura (4.7) apresenta a primeira parte da implementação da prova de conceito. Como se pode verificar a partir a da linha 28 é definida a construção da página HTML. No primeiro bloco, situado entre as linhas 28 e 31 são definidas as ligações e configurações que a página HTML vai assumir. Depois da linha 31 é definido o corpo da página. Este é composto pelas restantes seis partes. A parte entre as linhas 33 e 43 define o cabeçalho da página HTML.

```
1  [%%shared
2      open Eliom_lib
3      open Eliom_content
4      open Html.D
5      open Ocsi_simple
6  ]
7
8  module Site2_app =
9    Eliom_registration.App (
10     struct
11       let application_name = "site2"
12       let global_data_path = None
13     end)
14
15 let main_service =
16   Eliom_service.create
17     ~path:(Eliom_service.Path [])
18     ~meth:(Eliom_service.Get Eliom_parameter.unit)
19     ()
20
21 let () =
22   Eliom_content.Html.D.
23   (Site2_app.register
24     ~service:main_service
25     (fun () () ->
26       Lwt.return(
27         (Ocsi_simple.
28           (webpage
29             ~p_title:"Prova 2"
30             ~p_css:(css_links ~css_local:["css/site2.css"] ~links:["https://use.fontawesome.com/releases/v5.0.10/css/all.css"] () )
31             ~p_body:(body_page
32               ~content: [
33 (*Header ShowCase*)       s_header ~css_id:"showcase" ~css_class:"grid"
34                           ~content:[
35                             s_div ~css_class:"bg-image" ~content:[] ();
36                             s_div ~css_class:"content-wrap"
37                               ~content:[
38                                 s_h1 (s_text "Prova de Conceito 2" ()) ();
39                                 s_p (s_text "Lorem ipsum dolor sit amet consectetur
40                                              adipisicing elit. Modi, ipsam! Nihil quo minima nulla atque!" ()) ();
41                                 s_ahref ~link:"#" ~css_class:"btn" ~stext:(s_text "Read More" ()) ();
42                               ] ();
43                           ] ();
```

Figura 4.7: Código da prova de conceito 1 (Parte 1).

De seguida, como apresentado na imagem (4.8) entre as linhas 45 e 59 é definida a segunda estrutura, que define a delimitação entre o cabeçalho e a terceira estrutura.

```
45 (*MAIN AREA*) s_main ~css_class:"main"
46                 ~content: [
47                   s_section ~css_id:"section-a" ~css_class:"grid"
48                     ~content: [
49                       s_div ~css_class:"content-wrap"
50                         ~content: [
51                           s_h2 ~css_class:"content-title" ~stext:(s_text "Prova de Conceito 2" ()) ();
52                           s_div ~css_class:"content-text"
53                           ~content: [
54                             s_p ~stext:(s_text "Lorem ipsum dolor sit amet consectetur
55                                                 adipisicing elit. Modi, ipsam! Nihil quo minima
56                                                 nulla atque!" ()) ();
57                           ] ();
58                         ] ();
59                     ] ();
60
```

Figura 4.8: Código da prova de conceito 1 (Parte 2).

A imagem (4.9) apresenta a terceira estrutura da página HTML, a qual é composta por uma lista com três linhas formatadas de forma horizontal. Estas por sua vez definem novas estruturas que estão inseridas dentro da lista principal.

```
("Section B") s_section ~css_id:"section-b" ~css_class:"grid"
                ~content: [
                  ul [
                    li [
                      s_div ~css_class:"card"
                        ~content: [
                          s_img ~link:"https://static.pexels.com/photos/574077/pexels-photo-574077.jpeg" ();
                          s_div ~css_class:"card-content"
                            ~content: [
                              s_h3 ~css_class:"content-title" ~stext:(s_text "Prova de Conceito 2" ()) ();
                              s_p ~stext:(s_text "Lorem ipsum dolor sit amet consectetur
                                                  adipisicing elit. Modi, ipsam! Nihil quo minima
                                                  nulla atque!" ()) ();
                            ] ();
                        ] ()
                    ];
                    li [
                      s_div ~css_class:"card"
                        ~content: [
                          s_img ~link:"https://static.pexels.com/photos/261628/pexels-photo-261628.jpeg" ();
                          s_div ~css_class:"card-content"
                            ~content: [
                              s_h3 ~css_class:"content-title" ~stext:(s_text "Prova de Conceito 2" ()) ();
                              s_p ~stext:(s_text "Lorem ipsum dolor sit amet consectetur
                                                  adipisicing elit. Modi, ipsam! Nihil quo minima
                                                  nulla atque!" ()) ();
                            ] ();
                        ] ()
                    ];
                    li [
                      s_div ~css_class:"card"
                        ~content: [
                          s_img ~link:"https://static.pexels.com/photos/265087/pexels-photo-265087.jpeg" ();
                          s_div ~css_class:"card-content"
                            ~content: [
                              s_h3 ~css_class:"content-title" ~stext:(s_text "Prova de Conceito 2" ()) ();
                              s_p ~stext:(s_text "Lorem ipsum dolor sit amet consectetur
                                                  adipisicing elit. Modi, ipsam! Nihil quo minima
                                                  nulla atque!" ()) ();
                            ] ();
                        ] ()
                    ];
                  ]
                ] ();
```

Figura 4.9: Código da prova de conceito 2 (Parte 3).

A ultima parte corresponde às três ultimas estruturas.

```
("Section C") s_section ~css_id:"section-c" ~css_class:"grid"
                ~content: [
                  s_div ~css_class:"content-wrap"
                    ~content: [
                      s_h2 ~css_class:"content-title" ~stext:(s_text "We handle all of your digital needs" ()) ();
                      s_p ~stext:(s_text "Lorem ipsum dolor sit amet consectetur
                                          adipisicing elit. Modi, ipsam! Nihil quo minima
                                          nulla atque!" ()) ();
                    ] ()
                ] ();

("Section D") s_section ~css_id:"section-d" ~css_class:"grid"
                ~content: [
                  s_div ~css_class:"box"
                    ~content: [
                      s_h2 ~css_class:"content-title" ~stext:(s_text "Contact Us" ()) ();
                      s_p ~stext:(s_text "Lorem ipsum dolor sit amet consectetur
                                          adipisicing elit. Modi, ipsam! Nihil quo minima
                                          nulla atque!" ()) ();
                      s_p ~stext:(s_text "antonio.pedro.gaspar@ubi.pt" ()) ();
                    ] ();
                  s_div ~css_class:"box"
                    ~content: [
                      s_h2 ~css_class:"content-title" ~stext:(s_text "About Our Company" ()) ();
                      s_p ~stext:(s_text "Lorem ipsum dolor sit amet consectetur
                                          adipisicing elit. Modi, ipsam! Nihil quo minima
                                          nulla atque!" ()) ();
                    ] ();
                ] ();

("Footer") s_section ~css_id:"main-footer" ~css_class:"grid"
              ~content: [ s_div ~content: [ s_p ~stext:(s_text "Release Solutions" ()) () ] ();
                          s_div ~content: [
                            s_text "Template from " () ;
                            s_ahref ~link:"http://traversymedia.com" ~target:"_blank" ~stext:(s_text "Traversy Media" ()) ();
                          ] ()
                        ]()
          ]()
        ]()
      )
    )()
  )
 )
)
```

Figura 4.10: Código da prova de conceito 2 (Parte 4).

4.6 Validação

Esta secção tem como objetivo expor a comparação entre as duas provas de conceito apresentadas. Desta forma e com base no estudo prévio dos capítulos anteriores é justificada a criação da Framework Ocsi_simple. *De seguida são apresentados e comparados, trechos de código das duas provas de conceito. É importante referir que aqui é apenas analisado o módulo* Ocsi_simple, *pois é este que tem como principal objetivo suprimir os problemas referidos nos capítulos anteriores. Os restantes módulos* Ocsi_simple_persist *e* Ocsi_simple_js, *são módulos que têm como objetivo complementar o uso do módulo* Ocsi_simple, *desta forma no anexo C, serão apresentados e descritos de forma prática.*

```
(html
(head
(title (pcdata "Prova 1"))
[
meta ~a:[ a_charset "UTF-8"; a_name "viewport"; a_content "width=device-width, initial-scale=1"]();
css_link ~uri:(make_uri (Eliom_service.static_dir ()) ["css"; "site2.css"]) ();
css_link ~uri:(Raw.uri_of_string "https://use.fontawesome.com/releases/v5.0.10/css/all.css") ();
]
)
(body [
(*Header ShowCase*)
header ~a:[ a_id "showcase"; a_class ["grid"] ]
[
div ~a:[ a_class ["bg-image"] ] [];
div ~a:[ a_class ["content-wrap"] ]
[
h1 [pcdata "Prova de Conceito 1"];
p [pcdata "Lorem ipsum dolor sit amet consectetur
    adipisicing elit. Modi, ipsam!
    Nihil quo minima nulla atque!"
];
Raw.a ~a:[a_href (Raw.uri_of_string "#"); a_class ["btn"]] [pcdata "Read More"];
]; ];
```

Listing 4.7: (POC-1) Criação da página HTML e corpo.

```
(webpage
~p_title:"Prova 2"
~p_css:(css_links ~css_local:["css/site2.css"] ~links:["https://use.fontawesome.com/releases/v5.0.10/css/all.css"] () )
~p_body:(body_page
~content: [
(*Header ShowCase*)
s_header ~css_id:"showcase"
~css_class:"grid"
~content:[
s_div ~css_class:"bg-image" ~content:[] ();
s_div ~css_class:"content-wrap"
~content:[
s_h1 (s_text "Prova de Conceito 2" () ) ();
s_p (s_text "Lorem ipsum dolor sit amet consectetur
    adipisicing elit. Modi, ipsam!
    Nihil quo minima nulla atque!" ()) ();
s_ahref ~link:"#" ~css_class:"btn" ~stext:(s_text "Read More" ()) ();
] (); ] ();
```

Listing 4.8: (POC-2) Criação da página HTML e corpo.

Como se pode verificar nos trechos de código (4.7) e (4.8), o tamanho do código é desde logo inferior no trecho de código (4.8) pertencente à segunda prova de conceito que utiliza a nova solução desenvolvida, a Framework Ocsi_simple. *No entanto essa não é mais valia principal. É de notar que o código apresentado em (4.8) é muito mais compacto e organizado, devido à utilização da função* webpage, *que abstrai o utilizador de conhecer a fundo a sintaxe nativa da tecnologia* Ocsigen. *Desta forma é reduzida significativamente a curva de aprendizagem, o que leva a uma maior produtividade e aceitação. Um outro ponto notório é a construção de elementos de marcação, os quais podem ser facilmente instanciados com as possíveis configurações que suportam. Apesar de nova solução desenvolvida, a* Framework Ocsi_simple *ser mais verbosa em termos dos seus parâmetros, isso confere-lhe uma maior assertividade na utilização destes, pois estão bem identificados.*

```
1  (*Section C*)
2  section ~a:[ a_id "section-c";
       a_class ["grid"] ]
3  [
4  div ~a:[ a_class ["content-wrap"] ]
5  [
6  h2 ~a:[a_class ["content-title"]] [
       pcdata "We handle all of your
       digital needs"];
7  p [pcdata "Lorem ipsum dolor sit amet
        consectetur
8  adipisicing elit. Modi, ipsam! Nihil
       quo minima nulla atque!"
9  ];
10 ];
11 ];
```

Listing 4.9: (POC-1) Criação da secção *C*.

```
1  (*Section C*)
2  s_section ~css_id:"section-c" ~
       css_class:"grid"
3  ~content: [
4  s_div ~css_class:"content-wrap"
5  ~content: [
6  s_h2 ~css_class:"content-title" ~
       stext:(s_text "We handle all of
       your digital needs" ()) ();
7  s_p ~stext:(s_text "Lorem ipsum dolor
        sit amet consectetur
8  adipisicing elit. Modi, ipsam! Nihil
       quo minima
9  nulla atque!" ()) ();
10 ] ()
11 ] ();
```

Listing 4.10: (POC-2) Criação da secção *C*.

Os trechos de código (4.9) e (4.10) apresentam a criação de uma section. *Verifica-se que apesar das semelhanças óbvias e do trecho de código (4.10), da segunda prova de conceito ser até um pouco mais verboso. Este é desde logo à partida de fácil leitura e interpretação, enquanto que, o trecho (4.9) pode causar alguma confusão se não houver um conhecimento prévio da tecnologia* Ocsigen.

```
footer ~a:[ a_id "main-footer";
    a_class ["grid"]]
[
div [pcdata "Release Solutions"];
div
[
pcdata "Template fromy ";
Raw.a ~a:[a_href (Raw.uri_of_string "
    http://traversymedia.com");
    a_target "_blank"][pcdata "
    Traversy Media"];
];
];
```

Listing 4.11: (POC-1) Criação do elemento *footer*.

```
s_footer ~css_id:"main-footer" ~
    css_class:"grid"
~content: [ s_div ~content: [ s_p ~
    stext:(s_text "Release Solutions"
    ()) () ] ();
s_div ~content: [
s_text "Template from " () ;
s_ahref ~link:"http://traversymedia.
    com" ~target:"_blank" ~stext:(
    s_text "Traversy Media" ()) ();
] ()
]()
```

Listing 4.12: (POC-2) Criação do elemento *footer*.

Os trechos de código (4.11) e (4.12) apresentam a criação de um footer. *De forma prática estes exemplos vão reforçar as ideias expostas nos textos anteriores. É de notar que quando utilizada, a* Framework Ocsi_simple *torna o desenvolvimento sobre a tecnologia* Ocsigen, *mais prático e intuitivo, para programadores que estão habituados a lidar com desenvolvimento WEB.*

4.7 Conclusão

Com a analise obtida a partir dos capítulos anterior, que culminou na implementação da Framework Ocsi_simple, *no presente capítulo esta foi analisada ao nível da sua implementação e comparada com a tecnologia nativa. É agora altura de apresentar os resultados e conclusões deste trabalho, conteúdo que é apresentado no capítulo seguinte,* Conclusões e Trabalho Futuro *(5).*

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Resultados

É possível agora apresentar as devidas conclusões. Com base na definição do problema (1.2) e da experiência adquirida em contexto industrial foi originada uma análise profunda sobre as tecnologias envolvidas e respetivas metodologias. Esta culminou num plano de trabalhos (1.3), que responde diretamente às questões levantadas na secção ***Definição do Problema*** *(1.2). Desta forma à pergunta, 1 - O que se pode fazer para tornar a* framework Ocsigen *mais simples de utilizar?. É apresentada como resposta a criação de uma nova framework, denominada por* Ocsi_simple Framework *e que é composta por três módulos:*

- **Ocsi_simple_persist** *é o módulo trata da persistência de dados, assegura o acesso aos SGBD* Mysql, PostgreSQL *e* SQLite *facilitando a sua utilização. Para o seu uso é apenas necessário identificar o SGBD pretendido e configurar a ligação ao motor de base de dados. É também importante referir que este módulo é escalável podendo ser aplicado a outros motores de bases de dados.*
- **Ocsi_simple** *é o módulo que permite agora, de uma forma simples, concisa, estruturada e limpa criar qualquer tipo de página HTML, sintaticamente correta. Este é o módulo que quando utilizado influência positivamente a produtividade e a organização do código produzido. É um dos módulos estruturais da nova* framework*, pois interliga a utilização dos outros módulos presentes nesta, proporcionado desta forma, um padrão de desenvolvimento durante a sua utilização.*
- **Ocsi_simple_js** *é o módulo que implementa várias funções que replicam o comportamento, de funções* JavaScript *habitualmente utilizadas pela grande maioria dos programadores WEB. A título de exemplo, o uso de uma função para adicionar um elemento a outro elemento. Este é o módulo, que permite dar aquele toque de interatividade nas páginas ou aplicações WEB criadas. Foi pensado e desenvolvido para facilitar a manipulação de elementos na parte cliente, da aplicação WEB.*

Relativamente à pergunta 2 - O que se pode fazer para tornar a framework Ocsigen *utilizável em* Windows *e por conseguinte em ambiente empresarial, qualquer que seja o seu ecossistema tecnológico (*Windows, Linux*, etc.)?. Como resposta é apresentado no capítulo 4 uma secção (4.3). O qual culmina no anexo* ***Ocsigen no Windows*** *(B) e no anexo* ***Tutorial Ocsi_simple Framework*** *(C). O anexo* ***Ocsigen no Windows*** *(B) tem como finalidade explicar o processo de instalação e utilização da tecnologia* Ocsigen

em ambiente Windows. *Desta forma é alcançado o objetivo da presente dissertação, o qual culminou num contributo considerado relevante à framework Ocsigen, tendo em conta o desafio colocado na secção anterior. É importante frisar que a solução desenvolvida pretende completar a* framework Ocsigen. *Os resultados práticos obtidos são disponibilizadas na forma de cinco provas de conceito desenvolvidas ao longo da dissertação:*

- `http://ocsisimple.ddns.net:8080` *apresenta a prova de conceito 1 (4.1) presente no capítulo 4. Esta foi desenvolvida unicamente com a plataforma* Ocsigen *nativa.*
- `http://ocsisimple.ddns.net:8081` *apresenta a prova de conceito 2 (4.6) presente no capítulo 4. A qual foi desenvolvida com o resultado desta dissertação, a* Ocsi_simple Framework.
- `http://ocsisimple.ddns.net:8082` *apresenta uma página desenvolvida com a* Ocsi_simple Framework *a partir de um template.*
- `http://ocsisimple.ddns.net:8083` *apresenta uma página onde foram testados os módulos da* Ocsi_simple Framework. *Nomeadamente os módulos* Ocsi_simple, Ocsi_simple_js *e* Ocsi_simple_persist.
- `http://ocsisimple.ddns.net:8084` *apresenta o caso de estudo desenvolvido no anexo* ***Tutorial Ocsi_simple Framework*** *(C).*

5.2 Trabalho Futuro

A Ocsi_simple Framework *foi desenhada para ser escalável, desta forma é possível acrescentar novas funcionalidades a cada módulo. No caso do módulo* Ocsi_simple *é possível adicionar novas estruturas de marcação. No módulo* Ocsi_simple_js *é possível adicionar novas funcionalidades que manipulem ou interajam com elementos HTML. Por fim no módulo* Ocsi_simple_persist *respeitando a* interface Imdb *(3.1) é possível construir e associar novos módulos que permitem o acesso a outros SGBD. A solução desenvolvida pode no futuro contribuir para o desenvolvido de novos projetos, tanto a nível académico, pessoal e até empresarial. O facto de ser possível desenvolver sobre o sistema operativo Windows abre inúmeras possibilidades que até pouco tempo eram notoriamente limitadas. Desta forma é agora possível utilizar o resultado alcançado e desenvolver aplicações, seguras por construção.*

Bibliografia

[15990] *Ieee standard glossary of software engineering terminology.* IEEE Std 610.12-1990, *pages 1-84, Dec 1990. 19*

[57310] *Iso/iec/ieee international standard - systems and software engineering - vocabulary.* ISO/IEC/IEEE 24765:2010(E), *pages 1-418, Dec 2010. 19*

[71015] *Iso/iec/ieee international standard - systems and software engineering - system life cycle processes.* ISO/IEC/IEEE 15288 First edition 2015-05-15, *pages 1-118, May 2015. 19*

[72098] *Ieee recommended practice for software requirements specifications.* IEEE Std 830-1998, *pages 1-40, Oct 1998. 19*

[cora] *A course from Cornell University, Data Structures and Functional Programming ,2017 . https://www.cs.cornell.edu/courses/cs3110/2017sp/. Último Acesso: 15-06-2018. 11*

[corb] *idem, «Cornell University», 2018. https://www.cs.cornell.edu/. Último Acesso: 15-06-2018. 11*

[corc] *idem, «Cornell University», Data Structures and Functional Programming, Code Examples, 2017. https://www.cs.cornell.edu/courses/cs3110/2017fa/l/08-functors/notes.html. Último Acesso: 15-06-2018. 11*

[cyg] *Cygwin Official Site. https://www.cygwin.com/. Último Acesso: 15-06-2018. 29*

[git] *Official Github OCaml Compiler. https://github.com/ocaml/ocaml/blob/trunk/README.win32.adoc. Último Acesso: 15-06-2018. 29*

[INR] *INRIA Official Site. https://www.inria.fr/. Último Acesso: 15-06-2018. 5*

[Ler90] *Xavier Leroy. The zinc experiment, an economical implementation of the ml language.* Technical report 117, *1990. Available from: http://caml.inria.fr/pub/papers/xleroy-zinc.pdf. 5, 6*

[Ler00] *Xavier Leroy. A modular module system.* J. Funct. Program., *10(3):269-303, May 2000. Available from: http://dx.doi.org/10.1017/S0956796800003683. 11*

[mic] *Microsoft Official Site. https://www.microsoft.com/. Último Acesso: 15-06-2018. 29*

[min] *MinGW Official Site. http://www.mingw.org/. Último Acesso: 15-06-2018. 29*

[Min13] *Y. Minsky.* Real World Ocaml. Functional Programming for the Masses. *2013. Available from: https://realworldocaml.org/. 11*

[Mog91] *Eugenio Moggi. Notions of computation and monads.* Information and Computation, *93(1):55 - 92, 1991. Selections from 1989 IEEE Symposium on Logic in Computer Science. Available from: http://www.sciencedirect.com/science/article/pii/0890540191900524. 14*

[Mys] *Mysql Official Site. https://www.mysql.com/. Último Acesso: 15-06-2018. 10*

[nod] *NodeJS Official Site. https://nodejs.org/en/. Último Acesso: 15-06-2018. 17*

[OCaa] *OCaml Official Site. https://ocaml.org/. Último Acesso: 15-06-2018. 6*

[OCab] *OCaml OPIUM Official Site. https://github.com/rgrinberg/opium. Último Acesso: 15-06-2018. 1*

[OCac] *OCaml-Webmachine Official Site. https://github.com/inhabitedtype/ocaml-webmachine. Último Acesso: 15-06-2018. 1*

[OCSa] *Ocsigen ELIOM Official Site. http://ocsigen.org/eliom/. Último Acesso: 15-06-2018. 9*

[OCSb] *Ocsigen GitHub Official Site. https://ocsigen.github.io/. Último Acesso: 15-06-2018. 7*

[OCSc] *Ocsigen JS-Of-OCaml Official Site. http://ocsigen.org/js_of_ocaml/. Último Acesso: 15-06-2018. 8*

[OCSd] *Ocsigen LWT Official Site. http://ocsigen.org/lwt/. Último Acesso: 15-06-2018. 9*

[OCSe] *Ocsigen Official Site. http://ocsigen.org/. Último Acesso: 15-06-2018. 7*

[OCSf] *Ocsigen SERVER Official Site. https://ocsigen.org/ocsigenserver/2.9/manual/quickstart. Último Acesso: 15-06-2018. 8*

[OCSg] *Ocsigen TyXML Official Site. http://ocsigen.org/tyxml/. Último Acesso: 15-06-2018. 10*

[opaa] *Andreas Hauptmann GitHub Official Site. https://fdopen.github.io/opam-repository-mingw/installation/. Último Acesso: 15-06-2018. 29*

[opab] *OPAM 2.0 Official Site. https://opam.ocaml.org/doc/2.0/. Último Acesso: 15-06-2018. 29*

[OPAc] *OPAM Official Site. https://opam.ocaml.org/. Último Acesso: 15-06-2018. 6*

[Pie91] *B.C. Pierce.* Basic Category Theory for Computer Scientists. *Foundations of computing series: research reports and notes. MIT Press, 1991. Available from: http://www.cis.upenn.edu/~bcpierce/. 13, 14*

[Pos] *PostgreSQL Official Site*. *https://www.postgresql.org/*. *Último Acesso: 15-06-2018. 10*

[rea] *ReasonML Official Site*. *https://reasonml.github.io/*. *Último Acesso: 01-06-2018. 1*

[rel] *UBI RELEASE Official Site*. *https://release.di.ubi.pt/*. *Último Acesso: 01-06-2018. iii*

[RV98] *Didier Rémy and Jérome Vouillon. Objective ml: An effective object-oriented extension to ml*. In Theory And Practice of Objects Systems, *page 27–50, 1998. Available from: http://caml.inria.fr/pub/papers/remy_vouillon-objective_ml-tapos98.pdf. 6*

[SQL] *Sqlite3 Official Site*. *https://www.sqlite.org/*. *Último Acesso: 15-06-2018. 10*

[v8] *Chrome V8 Official Site*. *https://developers.google.com/v8/*. *Último Acesso: 15-06-2018. 17*

[wina] *Microsoft Blog Official Site*. *https://blogs.msdn.microsoft.com/commandline/2018/01/12/chmod-chown-wsl-improvements/*. *Último Acesso: 15-06-2018. 72*

[winb] *Visual Studio Code Official Site*. *https://code.visualstudio.com/*. *Último Acesso: 15-06-2018. 67*

[winc] *Windows WSL Official Site*. *https://docs.microsoft.com/en-us/windows/wsl/install-win10*. *Último Acesso: 15-06-2018. 29, 65*

Apêndice A

Diagramas de casos de uso da *Ocsi_simple Framework*

A.1 Introdução

Este anexo tem como objetivo complementar o capítulo sobre análise e arquitetura da solução (3). São descritos de forma exaustiva todos os casos de uso da Ocsi_simple Framework.

A.2 Diagramas de Casos de Uso

A.2.1 Utilizador

A.2.1.1 Criar uma página HTML

Referente aos pontos 1, 2 e 3 de 3.4.1.

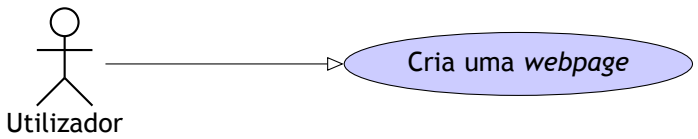


Figura A.1: Caso de uso sobre criar uma página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar qualquer página HTML através da função* webpage. *Esta terá como parâmetros opcionais, o título da página, as meta* tags, *as ligações aos ficheiros* CSS, *as ligações aos ficheiros* JavaScript *e por último, como parâmetro obrigatório, o corpo da página designado por* body.
- ***Estímulo:*** *O utilizador usa a função* webpage *para criar uma página HTML.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.2 Criar o corpo da página HTML

Referente aos pontos 1, 2 e 3 de 3.4.1.

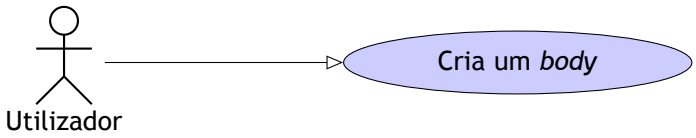


Figura A.2: Caso de uso sobre criar o corpo da página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *deverá criar o corpo da página HTML, através da função* body_page. *Esta terá como parâmetros opcionais, o identificador do* CSS, *a classe correspondente ao* CSS *e o conteúdo do corpo, que poderá ser formado por outros elementos de marcação.*
- ***Estímulo:*** *O utilizador usa a função* body_page *para criar o corpo de uma página HTML.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.3 Criar *meta tags* para página a HTML

Referente aos pontos 1, 2 e 3 de 3.4.1.

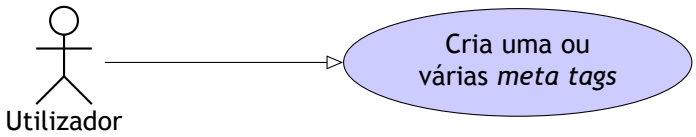


Figura A.3: Caso de uso sobre criar *meta tags* para a página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar* meta tags *através da função* s_meta. *Esta terá um parâmetro opcional, que permite adicionar uma ou várias* meta tags. *Por omissão será sempre aplicada a* meta tag meta charset="UTF-8" name="viewport" content="width=device-width, initial-scale=1.0".
- ***Estímulo:*** *O utilizador usa a função* s_meta *para criar uma ou várias* meta tags.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.4 Criar ligações *CSS* para a página HTML

Referente aos pontos 1, 2 e 3 de 3.4.1.

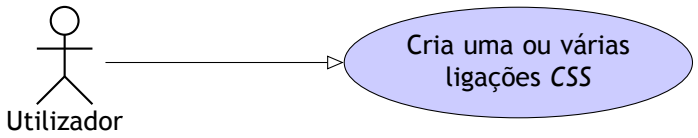


Figura A.4: Caso de uso sobre criar ligações *CSS* para a página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar ligações* CSS *através da função* css_links. *Esta terá dois parâmetros opcionais, que permitem adicionar uma ou várias ligações para ficheiros* CSS *online e localmente, respetivamente.*
- ***Estímulo:*** *O utilizador usa a função* css_links *para criar uma ou várias ligações para ficheiros* CSS.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.5 Criar ligações *JavaScript* para a página HTML

Referente aos pontos 1, 2 e 3 de 3.4.1.

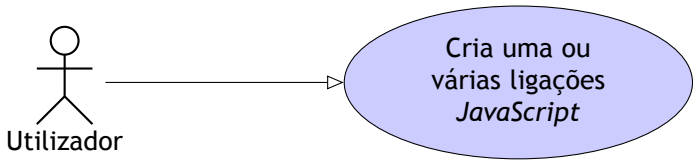


Figura A.5: Caso de uso sobre criar ligações *JavaScript* para a página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar ligações* JavaScript *através da função* js_links. *Esta terá dois parâmetros opcionais, que permitem adicionar uma ou várias ligações para ficheiros* JavaScript *online e localmente, respetivamente.*
- ***Estímulo:*** *O utilizador usa a função* js_links *para criar uma ou várias ligações para ficheiros* JavaScript.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.6 Criar uma *div*

Referente aos pontos 1, 2 e 3 de 3.4.1.

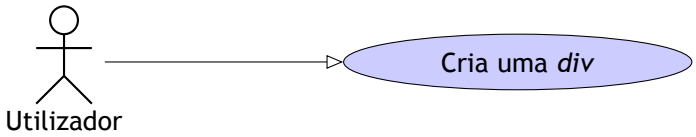


Figura A.6: Caso de uso sobre criar uma *div*.

- ***Ator**: O utilizador da* Framework Ocsi_simple.
- ***Descrição**: O utilizador da* Framework Ocsi_simple *poderá criar uma* div*, através da função* s_div. *Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o conteúdo da* div*, que poderá ser formado por outros elementos de marcação.*
- ***Estímulo**: O utilizador usa a função* s_div *para criar uma* div.
- ***Comentários**: O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.7 Criar um *header*

Referente aos pontos 1, 2 e 3 de 3.4.1.

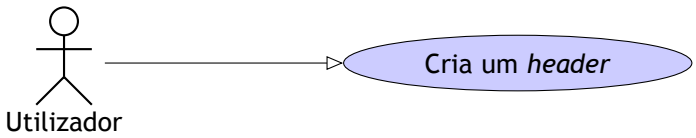


Figura A.7: Caso de uso sobre criar um *header*.

- ***Ator**: O utilizador da* Framework Ocsi_simple.
- ***Descrição**: O utilizador da* Framework Ocsi_simple *poderá criar um* header*, através da função* s_header. *Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o conteúdo do* header*, que poderá ser formado por outros elementos de marcação.*
- ***Estímulo**: O utilizador usa a função* s_header *para criar um* header.
- ***Comentários**: O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.8 Criar um *article*

Referente aos pontos 1, 2 e 3 de 3.4.1.

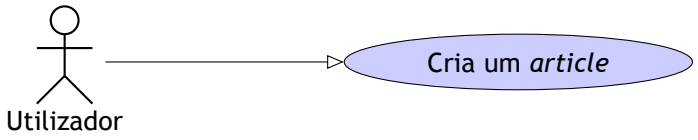


Figura A.8: Caso de uso sobre criar um *article*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* article*, através da função* s_article*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o conteúdo do* article*, que poderá ser formado por outros elementos de marcação.*
- ***Estímulo:*** *O utilizador usa a função* s_article *para criar um* article.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.9 Criar um *aside*

Referente aos pontos 1, 2 e 3 de 3.4.1.

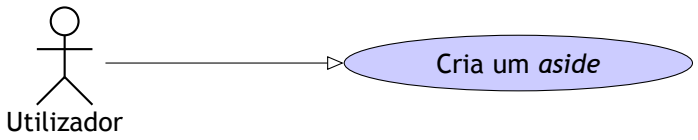


Figura A.9: Caso de uso sobre criar um *aside*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* aside*, através da função* s_aside*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o conteúdo do* aside*, que poderá ser formado por outros elementos de marcação.*
- ***Estímulo:*** *O utilizador usa a função* s_aside *para criar um* aside.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.10 Criar uma *section*

Referente aos pontos 1, 2 e 3 de 3.4.1.

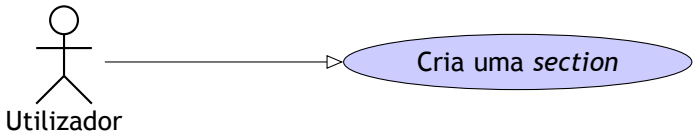


Figura A.10: Caso de uso sobre criar uma *section*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar uma* section, *através da função* s_section. *Esta terá como parâmetros opcionais, o identificador do* CSS, *a classe correspondente ao* CSS, *o estilo do elemento e o conteúdo da* section, *que poderá ser formado por outros elementos de marcação.*
- ***Estímulo:*** *O utilizador usa a função* s_section *para criar uma* section.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.11 Criar uma *nav*

Referente aos pontos 1, 2 e 3 de 3.4.1.

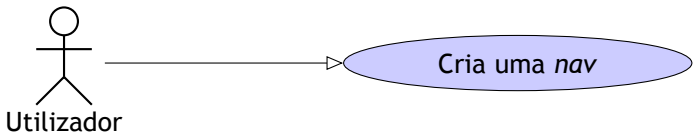


Figura A.11: Caso de uso sobre criar uma *nav*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar uma* nav, *através da função* s_nav. *Esta terá como parâmetros opcionais, o identificador do* CSS, *a classe correspondente ao* CSS, *o estilo do elemento e o conteúdo da* nav, *que poderá ser formado por outros elementos de marcação.*
- ***Estímulo:*** *O utilizador usa a função* s_nav *para criar uma* nav.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.12 Criar um *footer*

Referente aos pontos 1, 2 e 3 de 3.4.1.

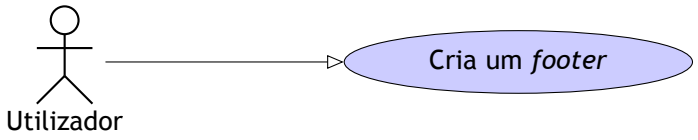


Figura A.12: Caso de uso sobre criar um *footer*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* footer*, através da função* s_footer*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o conteúdo do* footer*, que poderá ser formado por outros elementos de marcação.*
- ***Estímulo:*** *O utilizador usa a função* s_footer *para criar um* footer.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.13 Criar um *stext*

Referente aos pontos 1, 2 e 3 de 3.4.1.

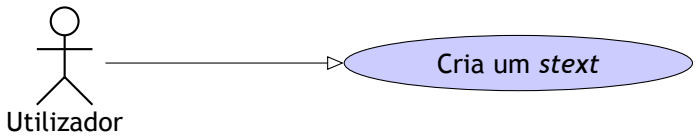


Figura A.13: Caso de uso sobre criar um *stext*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* stext*. Um* stext*, é texto formatado com determinados estilos. Será criado através da função* s_text*. Esta terá como parâmetros opcionais, a formatação do texto em* bold *e em* itálico*, como parâmetro obrigatório terá o próprio texto.*
- ***Estímulo:*** *O utilizador usa a função* s_text *para criar um* stext.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.14 Criar um *span*

Referente aos pontos 1, 2 e 3 de 3.4.1.

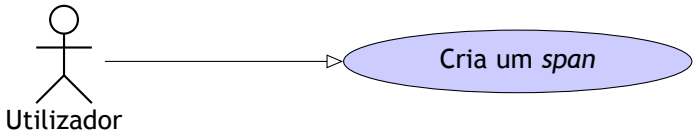


Figura A.14: Caso de uso sobre criar um *span*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* span*, através da função* s_span*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo*stext.
- ***Estímulo:*** *O utilizador usa a função* s_span *para criar um* span.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.15 Criar um *p*

Referente aos pontos 1, 2 e 3 de 3.4.1.

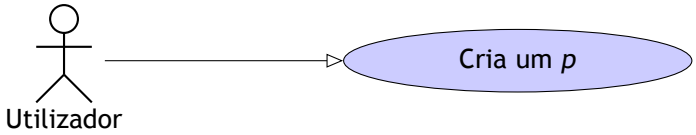


Figura A.15: Caso de uso sobre criar um *p*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* p*, através da função* s_p*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_p *para criar um* p.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.16 Criar um *h1*

Referente aos pontos 1, 2 e 3 de 3.4.1.

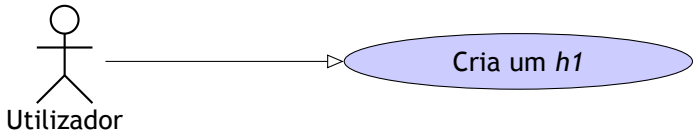


Figura A.16: Caso de uso sobre criar um *h1*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* h1*, através da função* s_h1*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_h1 *para criar um* h1.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.17 Criar um *h2*

Referente aos pontos 1, 2 e 3 de 3.4.1.

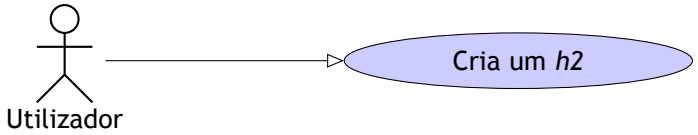


Figura A.17: Caso de uso sobre criar um *h2*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* h2*, através da função* s_h2*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_h2 *para criar um* h2.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.18 Criar um *h3*

Referente aos pontos 1, 2 e 3 de 3.4.1.

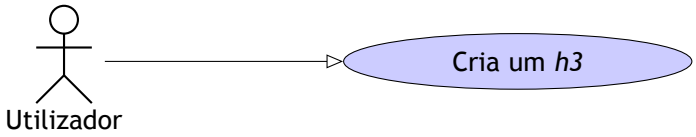


Figura A.18: Caso de uso sobre criar um *h3*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* h3*, através da função* s_h3*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_h3 *para criar um* h3.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.19 Criar um *h4*

Referente aos pontos 1, 2 e 3 de 3.4.1.

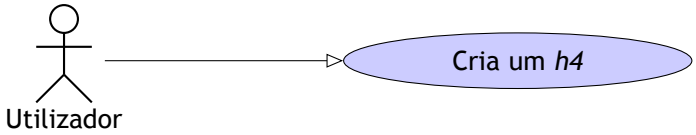


Figura A.19: Caso de uso sobre criar um *h4*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* h4*, através da função* s_h4*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_h4 *para criar um* h4.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.20 Criar um *h5*

Referente aos pontos 1, 2 e 3 de 3.4.1.

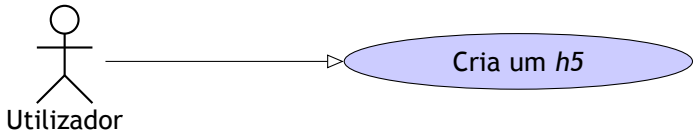


Figura A.20: Caso de uso sobre criar um *h5*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* h5*, através da função* s_h5*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_h5 *para criar um* h5.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.21 Criar um *h6*

Referente aos pontos 1, 2 e 3 de 3.4.1.

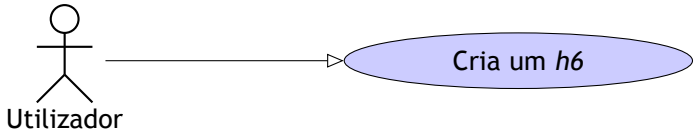


Figura A.21: Caso de uso sobre criar um *h6*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* h6*, através da função* s_h6*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_h6 *para criar um* h6.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.22 Criar um *hr*

Referente aos pontos 1, 2 e 3 de 3.4.1.

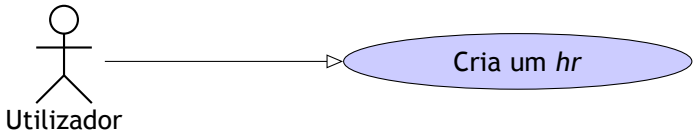


Figura A.22: Caso de uso sobre criar um *hr*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* hr*, através da função* s_hr*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_hr *para criar um* hr.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.23 Criar um *br*

Referente aos pontos 1, 2 e 3 de 3.4.1.

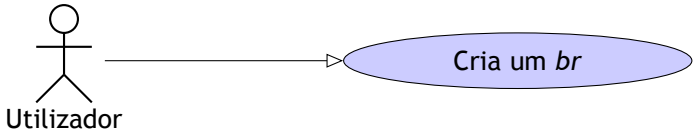


Figura A.23: Caso de uso sobre criar um *br*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* br*, através da função* s_br*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento e o respetivo* stext.
- ***Estímulo:*** *O utilizador usa a função* s_br *para criar um* br.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.24 Criar uma *img*

Referente aos pontos 1, 2 e 3 de 3.4.1.

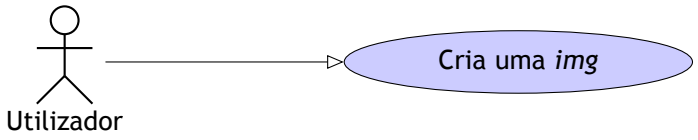


Figura A.24: Caso de uso sobre criar uma *img*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar uma* img, *através da função* s_img. *Esta terá como parâmetros opcionais, o identificador do* CSS, *a classe correspondente ao* CSS, *o estilo do elemento, a designação do objeto, uma ligação* online *e uma ligação* local.
- ***Estímulo:*** *O utilizador usa a função* s_img *para criar um* img.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.25 Criar um *button*

Referente aos pontos 1, 2 e 3 de 3.4.1.

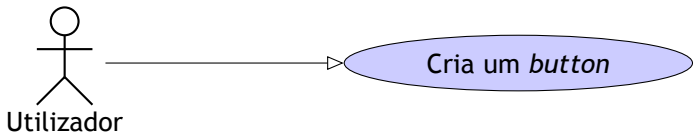


Figura A.25: Caso de uso sobre criar um *button*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* button, *através da função* s_button. *Esta terá como parâmetros opcionais, o identificador do* CSS, *a classe correspondente ao* CSS, *o estilo do elemento, a designação pretendida e por fim, a função que vai manipular a ação do elemento.*
- ***Estímulo:*** *O utilizador usa a função* s_button *para criar um* button.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.26 Criar um *href*

Referente aos pontos 1, 2 e 3 de 3.4.1.

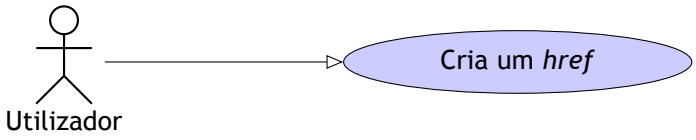


Figura A.26: Caso de uso sobre criar um *href*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* href*, através da função* s_href*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento, a ligação que se pretende referir e por fim a designação pretendida.*
- ***Estímulo:*** *O utilizador usa a função* s_href *para criar um* href.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.1.27 Criar um *aservice*

Referente aos pontos 1, 2 e 3 de 3.4.1.

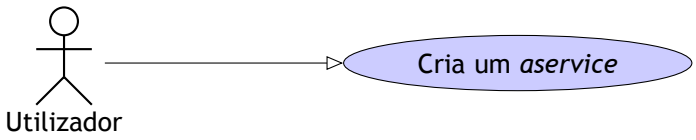


Figura A.27: Caso de uso sobre criar um *aservice*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar um* aservice*, através da função* s_aservice*. Esta terá como parâmetros opcionais, o identificador do* CSS*, a classe correspondente ao* CSS*, o estilo do elemento, o serviço que se pretende referir e por fim a designação pretendida.*
- ***Estímulo:*** *O utilizador usa a função* s_aservice *para criar um* aservice.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.2 Lado Cliente

A.2.2.1 Obter um elemento através do seu *id*

Referente ao ponto 4 de 3.4.1.

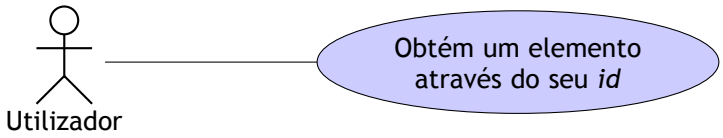


Figura A.28: Caso de uso sobre obter um elemento através do seu *id*.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá obter um elemento através do seu* id*, através da função* s_get_element_by_id. *Esta terá como parâmetro, o identificador do elemento que se pretende obter*.
- ***Estímulo:*** *O utilizador usa a função* s_get_element_by_id *para obter um elemento através do seu* id.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.2.2 Lançar um alerta na página HTML

Referente ao ponto 4 de 3.4.1.

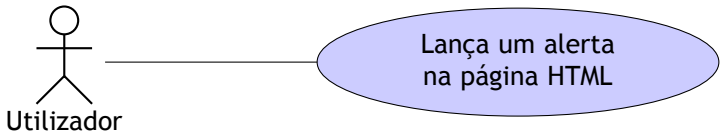


Figura A.29: Caso de uso sobre lançar um alerta na página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá lançar um alerta na página HTML, através da função* s_alert. *Esta terá como parâmetro a mensagem que se pretende transmitir*.
- ***Estímulo:*** *O utilizador usa a função* s_alert *para lançar um alerta na página HTML*.
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.2.3 Apresentar e ocultar elementos na página HTML

Referente ao ponto 4 de 3.4.1.

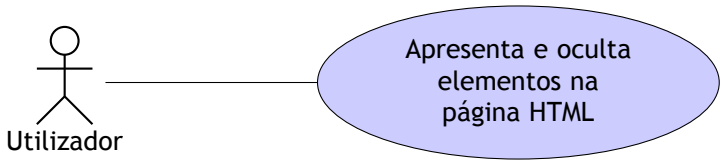


Figura A.30: Caso de uso sobre apresentar e ocultar elementos na página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá apresentar e ocultar elementos na página HTML, através da função* s_switch_visibility. *Esta terá dois parâmetros, que dizem respeito aos elementos que se pretendem apresentar e ocultar alternadamente.*
- ***Estímulo:*** *O utilizador usa a função* s_switch_visibility *para apresentar e ocultar elementos na página HTML.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.2.4 Adicionar um elemento a outro elemento na página HTML

Referente ao ponto 4 de 3.4.1.

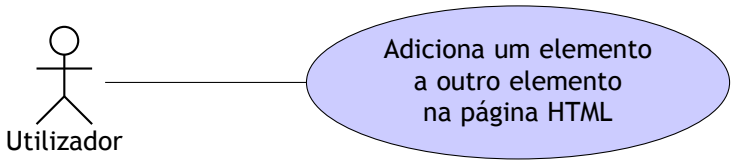


Figura A.31: Caso de uso sobre adicionar um elemento a outro elemento na página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá adicionar um elemento a outro elemento na página HTML, através da função* s_appendTo. *Esta terá dois parâmetros, que dizem respeito aos identificadores dos elementos, base e novo elemento.*
- ***Estímulo:*** *O utilizador usa a função* s_appendTo *para adicionar um elemento a outro elemento na página HTML.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.2.5 Remover um elemento de outro elemento na página HTML

Referente ao ponto 4 de 3.4.1.

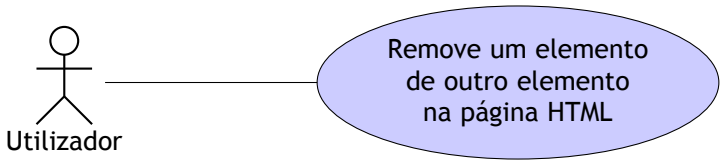


Figura A.32: Caso de uso sobre remover um elemento de outro elemento na página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá adicionar um elemento a outro elemento na página HTML, através da função* s_removeFrom. *Esta terá dois parâmetros, que dizem respeito aos identificadores dos elementos, base e elemento a remover.*
- ***Estímulo:*** *O utilizador usa a função* s_removefrom *para remover um elemento de outro elemento na página HTML.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.2.6 Limpar o conteúdo de um elemento na página HTML

Referente ao ponto 4 de 3.4.1.

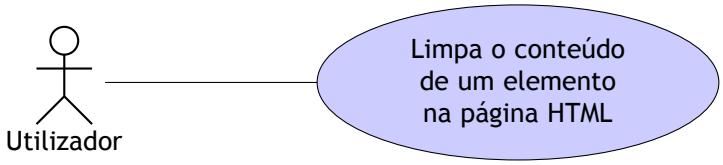


Figura A.33: Caso de uso sobre limpar o conteúdo de um elemento na página HTML.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá Limpar o conteúdo de um elemento na página HTML, através da função* s_clearbox. *Esta terá um parâmetro, que diz respeito ao identificador do elemento, ao qual se pretende limpar o seu conteúdo.*
- ***Estímulo:*** *O utilizador usa a função* s_clearbox *para limpar o conteúdo de um elemento na página HTML.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.3 Lado Servidor

A.2.3.1 Criar ligação para o SGBD pretendido

Referente ao ponto 5 de 3.4.1.

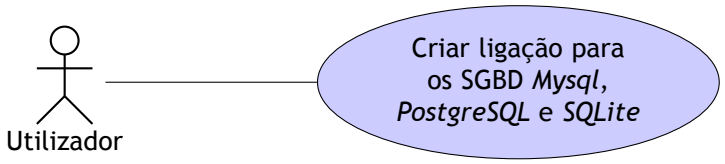


Figura A.34: Caso de uso sobre criar ligação, para o SGBD pretendido.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá criar uma ou várias ligações para o SGBD pretendido (*Mysql, PostgreSQL, SQLite*), através da criação de uma instancia, do módulo de base de dados pretendido.*
- ***Estímulo:*** *O utilizador instancia o módulo pretendido, para criar a ligação para o SGBD em causa (*Mysql, PostgreSQL, SQLite*).*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.3.2 Usar a ligação ao SGBD para inserir, atualizar e eliminar dados

Referente ao ponto 5 de 3.4.1.

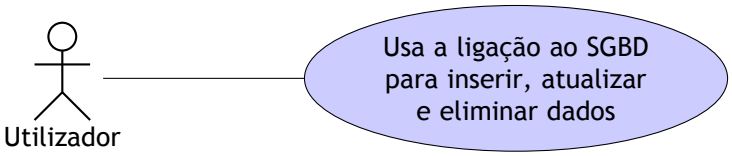


Figura A.35: Caso de uso sobre como usar a ligação ao SGBD para inserir, atualizar e eliminar dados.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá usar a ligação ao SGBD, previamente criada (A.2.3.1) para inserir, atualizar e eliminar dados. Esta operação é realizada através da função* exec_cdu, *tem como parâmetro a* query *que vai executar o tipo de manipulação de dados pretendida.*
- ***Estímulo:*** *O utilizador usa a função* exec_cdu *para inserir, atualizar e eliminar dados do SGBD pretendido.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.3.3 Usar a ligação ao SGBD para consultar dados

Referente ao ponto 5 de 3.4.1.

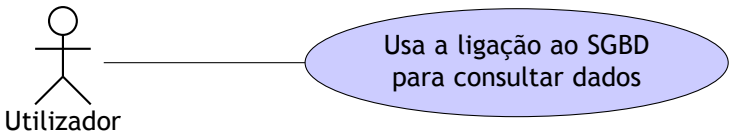


Figura A.36: Caso de uso sobre como usar a ligação ao SGBD para consultar dados.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá usar a ligação ao SGBD, previamente criada (A.2.3.1) para consultar dados. Esta operação é realizada através da função* exec_select, *tem como parâmetro a* query *que vai executar a seleção dos dados pretendidos.*
- ***Estímulo:*** *O utilizador usa a função* exec_select *para consultar dados no SGBD.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

A.2.3.4 Apresentar a consulta de dados ao SGBD (A.2.3.3)

Referente ao ponto 5 de 3.4.1.

Figura A.37: Caso de uso sobre a apresentação dos dados obtidos na consultar ao SGBD.

- ***Ator:*** *O utilizador da* Framework Ocsi_simple.
- ***Descrição:*** *O utilizador da* Framework Ocsi_simple *poderá apresentar os dados, obtidos através da consulta de dados previamente criada (A.2.3.3). Esta operação é realizada através da função* show_exec_select *tem como parâmetro a consulta de dados previamente criada (A.2.3.3).*
- ***Estímulo:*** *O utilizador usa a função* show_exec_select *para apresentar os dados da consultar realizado ao SGBD.*
- ***Comentários:*** *O utilizador deverá conhecer a* Framework Ocsi_simple.

Apêndice B

Ocsigen no Windows

B.1 Introdução

Este documento tem como objetivo demonstrar o uso da plataforma Ocsigen *no sistema operativo* Windows 10. *O* Windows 10 *possui um sistema chamado WSL [winc], que na prática é um emulador de um sistema operativo* Linux *com ambiente partilhado com o próprio* Windows 10. *Apesar desta tecnologia ainda estar em desenvolvimento e apresentar alguns problemas já é possível utiliza-la de forma prática. É importante ressalvar que todas as operações aqui apresentadas, apenas funcionam na versão do* Windows 10 *com compilação igual ou superior à* 17134.48*, como mostra a imagem seguinte (B.1). Para verificar a versão do* Windows 10*, basta no iniciar procurar pelo comando* winver.

Figura B.1: Número de compilação do *Windows 10*.

B.2 Instalação

Assegurando as condições necessárias parte-se agora para o processo de instalação. Para instalar o WSL é preciso ir à Microsoft Store *através do iniciar do* Windows 10. *Depois basta pesquisar por* "ubuntu" *e instalar, como é apresentado na imagem (B.2). Pode ser outra distribuição* Linux, *no entanto para âmbito desta demonstração, é a distribuição aconselhada devido à sua fiabilidade.*

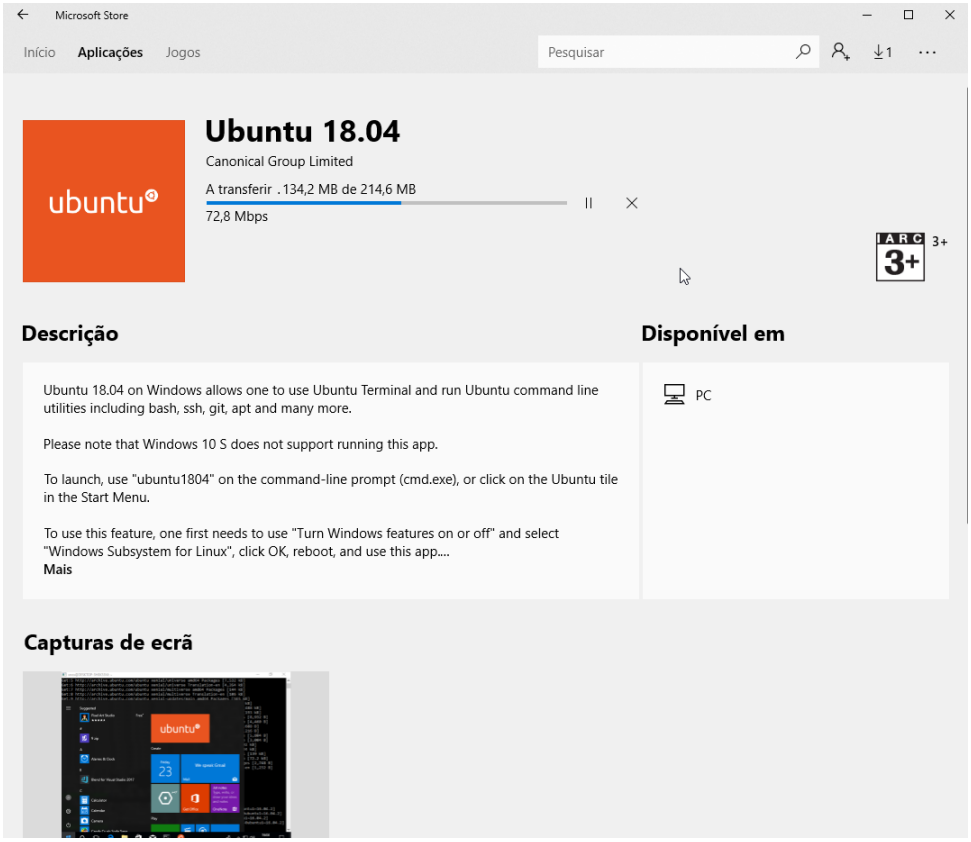


Figura B.2: Instalação da WSL (Parte 1).

Durante o processo de instalação irá aparecer uma linha de comandos como é apresentado na imagem (B.3). A operação pode levar vários minutos.

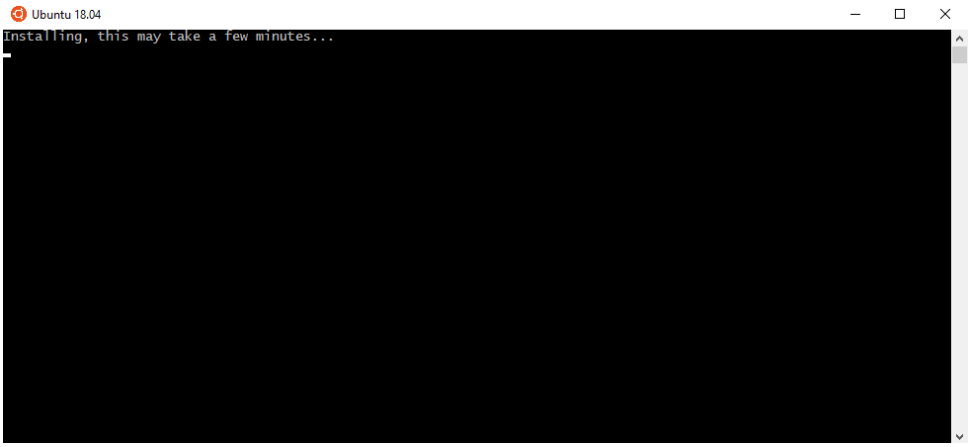


Figura B.3: Instalação da WSL (Parte 2).

Para concluir a instalação é necessário configurar um utilizador. A imagem B.4 apresenta esse passo.

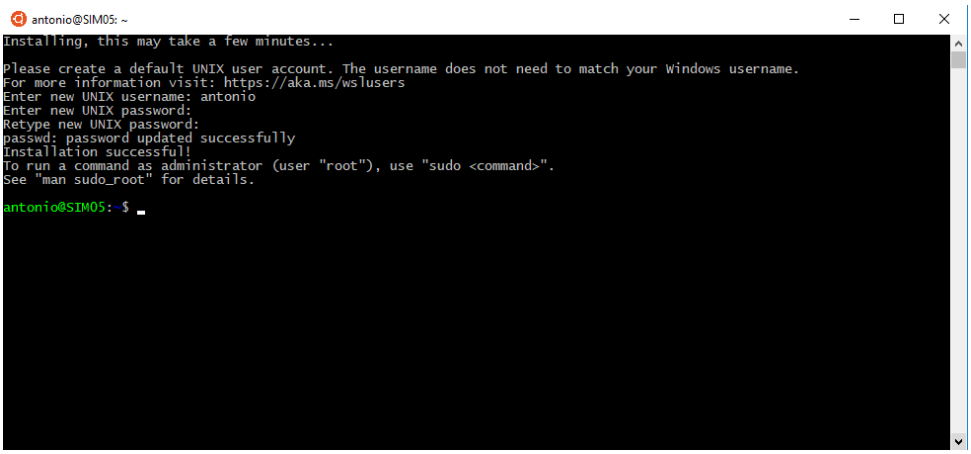


Figura B.4: Instalação da WSL (Parte 3).

Verifica-se agora na imagem (B.5), através do comando "lsb_release -a" *a versão da distribuição* Linux *instalada. Agora é possível executar praticamente todos os comandos naturais de uma distribuição* Linux. *A próxima fase trata de preparar o ambiente de desenvolvimento, a qual vai proporcionar o desenvolvimento sobre a tecnologia* Ocsigen *em ambiente* Windows.

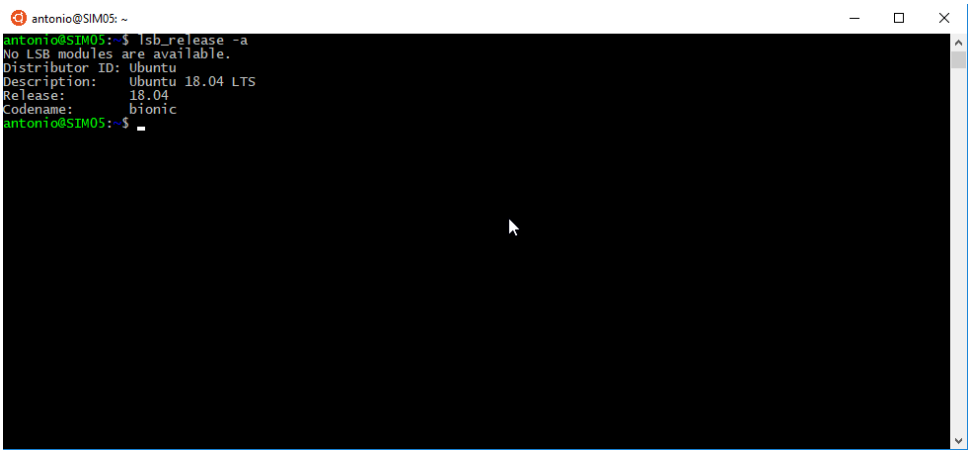


Figura B.5: Versão da distribuição *Linux*.

B.3 Ambiente de Desenvolvimento

Esta fase apresenta agora a configuração, para preparar o ambiente de desenvolvimento. Esta configuração terá como objetivo associar um editor de texto altamente parametrizável, o Visual Studio Code *[winb] à WSL. É importante referir que é possível utilizar qualquer outro editor. No entanto para efeitos de demonstração será utilizado o* Visual Studio Code. *Como apresentado na imagem seguinte (B.6) a primeira coisa a fazer é atualizar o sistema da WSL. Para isso basta usar o comando* "sudo apt update && sudo apt upgrade && sudo apt dist-upgrade".

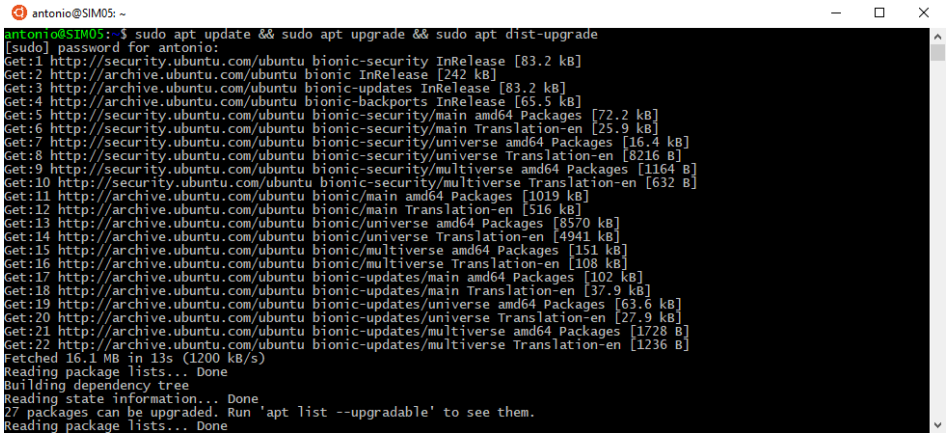


Figura B.6: Atualização da distribuição *Linux*.

Depois da atualização da distribuição Linux *é necessário instalar as ferramentas, que vão proporcionar a instalação e configuração do ecossistema OCaml. Para isso é necessário instalar o gestor de pacotes oficial do OCaml o* OPAM *executando o comando* "sudo apt-get install opam", *como apresentado na imagem (B.7).*

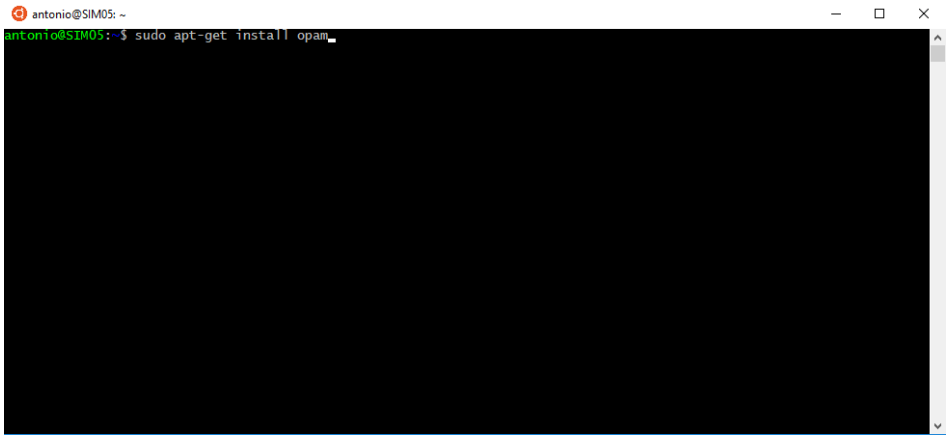


Figura B.7: Instalação do *OPAM*.

Ao concluir a instalação é agora necessário configurar o OPAM para o sistema alvo. Para isso basta executar o comando "opam init", *como apresentado na imagem (B.8).*

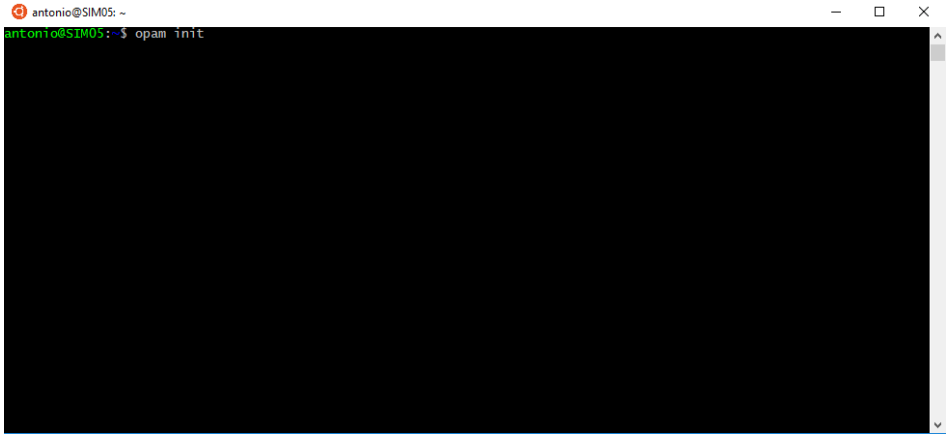


Figura B.8: Configuração do *OPAM*.

Depois de configurado o OPAM*, é necessário configurar a* Bash *da distribuição* Linux*, para manter as variáveis de ambiente do* OPAM *sempre atualizadas. Para isso é preciso editar o ficheiro* ”.bashrc”*, como é apresentado na imagem seguinte (B.9).*

Figura B.9: Edição do ficheiro *”.bashrc”* (Parte 1).

No final do ficheiro ”.bashrc” *adiciona-se a seguinte linha* ”eval eval ’opam config env’”*. como é apresentado na imagem seguinte (B.10).*

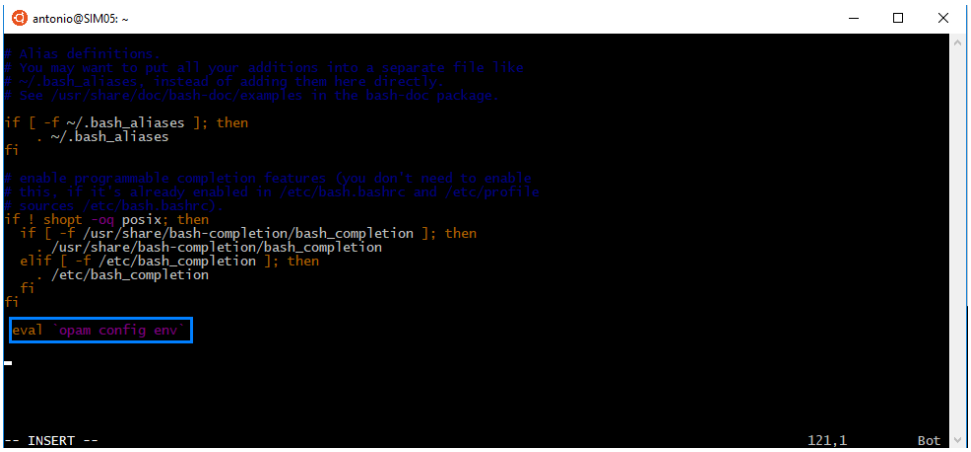


Figura B.10: Edição do ficheiro *”.bashrc”* (Parte 2).

Em seguida, é instalada a ultima versão oficial do compilador OCaml 4.06.1 *à data* 01-06-2018*, através do comando* ”opam switch 4.06.1”.

Figura B.11: Instalação do compilador *OCaml*.

Confirma-se através do comando "ocaml"*, que a versão do compilador anteriormente instalado é de facto a pretendida, como se comprova na imagem B.12*

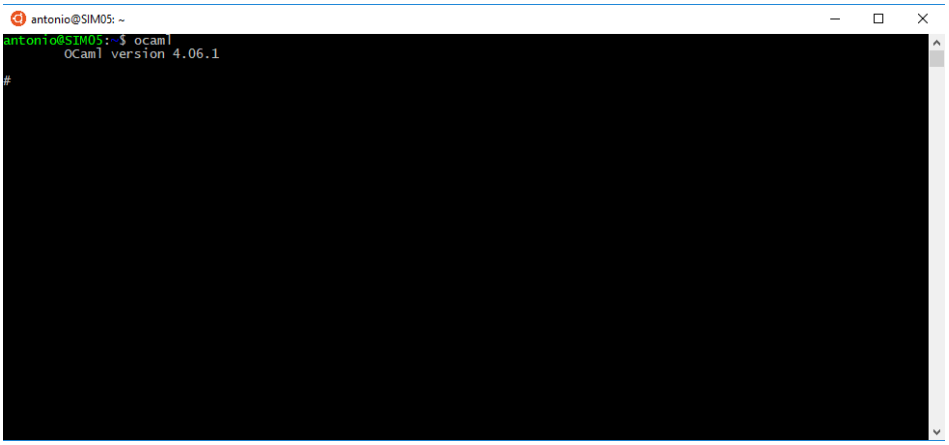


Figura B.12: *TopLevel* OCaml.

É também necessário instalar uma ferramenta que agilize a instalação de outros pacotes, através da localização de dependências. Essa ferramenta é instalada através do comando "opam install depext"*, apresentado na imagem (B.13).*

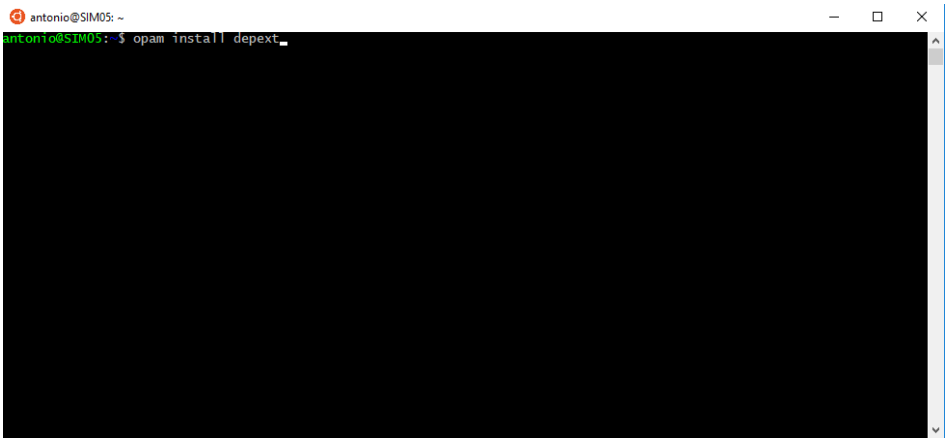


Figura B.13: Instalação do pacote *depext*.

Parte-se agora para a instalação de todos os pacotes inerentes à tecnologia Ocsigen. *Usa-se o comando* "opam depext -i ocsigen-start", *como é apresentado na imagem (B.14).*

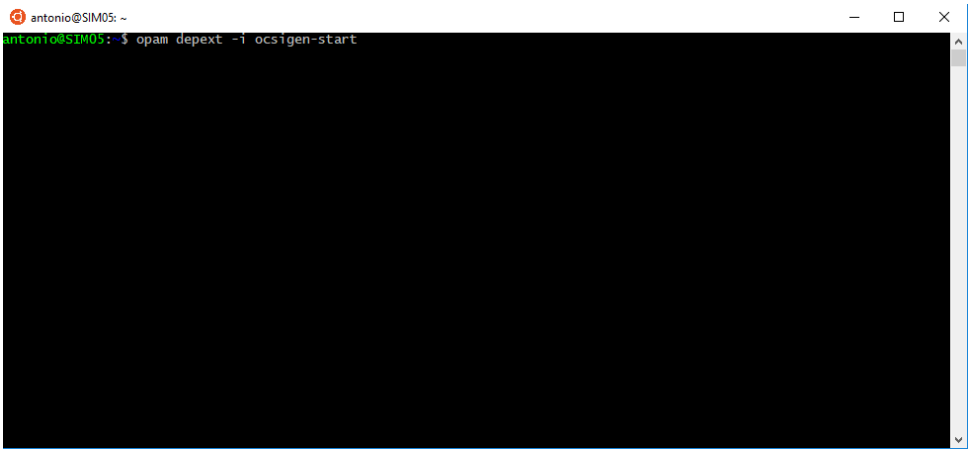


Figura B.14: Instalação da tecnologia *Ocsigen* (Parte 1).

Como se pode verificar na imagem (B.15), o depext *obtém todas as dependências necessárias à instalação do pacote* "ocsigen-start".

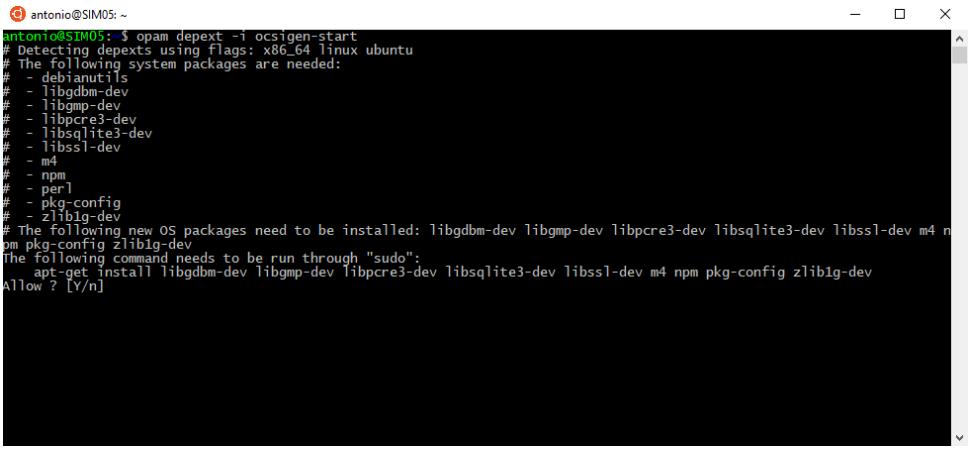


Figura B.15: Instalação da tecnologia *Ocsigen* (Parte 2).

Depois de instalados todos os pacotes necessários parte-se agora para a configuração da estrutura que vai partilhar o ambiente Windows *e o ambiente da WSL. Numa primeira fase cria-se uma nova pasta na raiz do disco* C. *Na WSL, o disco* C *esta montado na localização* "/mnt/c". *Desta forma é criada a pasta* "ocsigen_teste", *através do comando* "mkdir /mnt/c/ocsigen_teste", *abrindo-a de seguida. Com a estrutura criada é possível de imediato gerar um projeto* Ocsigen. *Utilizando a ferramenta* "eliom-distillery", *através do comando* "eliom-distillery -name ocsi_teste -template basic.ppx -target-directory ." *é criado um novo projeto já com todas as dependências prontas, como é ilustrado na imagem (B.16).*

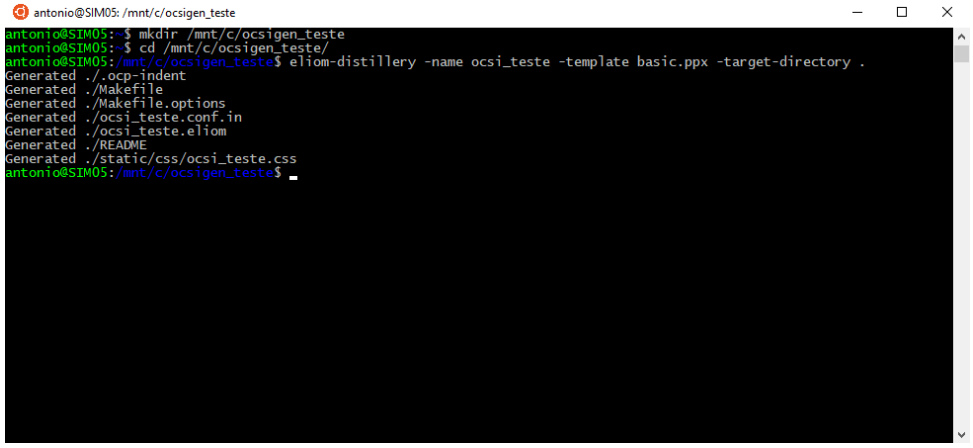


Figura B.16: Geração de um projeto com *Ocsigen*.

Neste ponto está praticamente tudo concluído, no entanto para haver uma partilha de ambientes bem configurada de forma a suportar pipes *é necessário configurar o ponto de montagem do disco* C. *Este é um ponto de extrema importância, pois sem ele o ambiente de trabalho em* Windows *deixa de ser possível devido à utilização de* pipes *e ligações dinâmicas que a tecnologia* Ocsigen *necessita para realizar uma compilação correta e execução correta do executável gerado [wina]. É então fundamental desmontar o ponto de montagem do disco* C, *através do comando* "sudo umount /mnt/c" *e de seguida monta-lo novamente com o comando* "sudo mount -t drvfs C: /mnt/c -o metadata", *como é apresentado na imagem (B.17).*

Figura B.17: Configuração do ambiente partilhado.

Para completar a configuração do ambiente partilhado, agora basta apenas dar permissões de controlo total à nova pasta criada, como é apresentado na imagem (B.18). Está assim concluída a configuração de ambiente partilhado entre o Windows *e a WSL.*

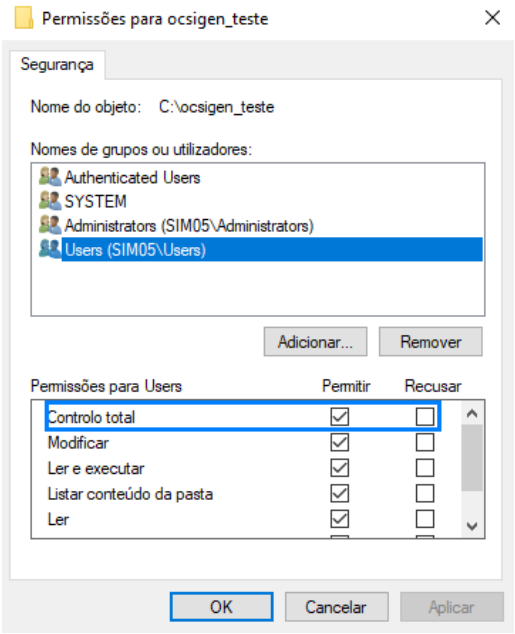


Figura B.18: Atribuição de permissões à pasta de *ocsigen_teste*.

Com a configuração de ambiente partilhado concluída, agora basta utilizar o Visual Studio Code *ou outro editor para editar código. No* Visual Studio Code *é então aberta a pasta do projeto, como é apresentado na imagem (B.19).*

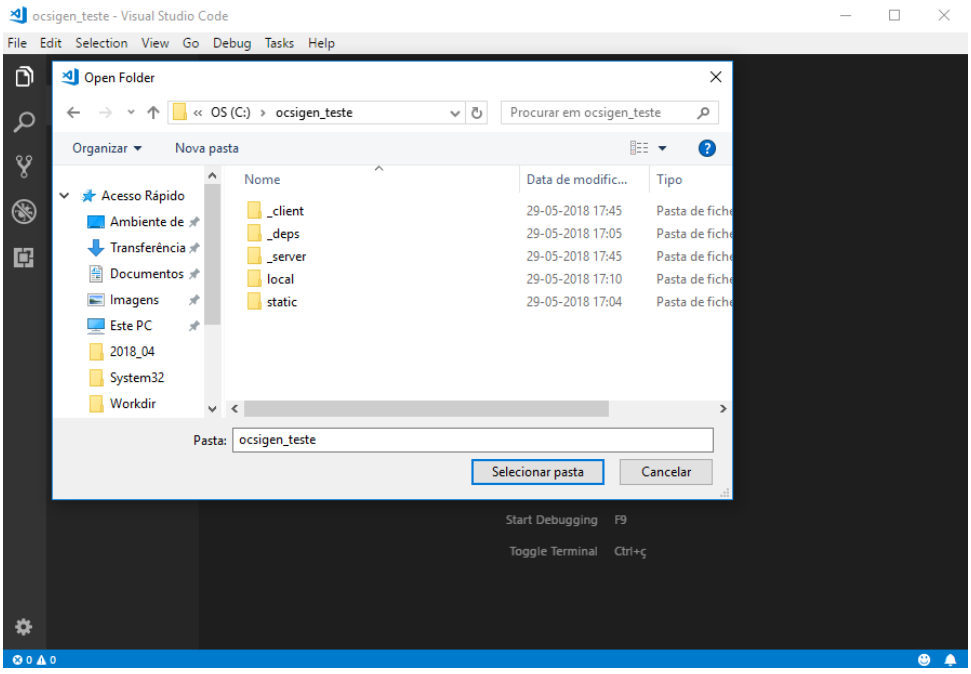


Figura B.19: Uso do *Visual Studio Code*.

De seguida é aberto o ficheiro de código eliom. *É importante notar que o texto do elemento* h1 *apresentado após a compilação é* "Welcome from Eliom's distillery", *como se pode constatar na imagem (B.20).*

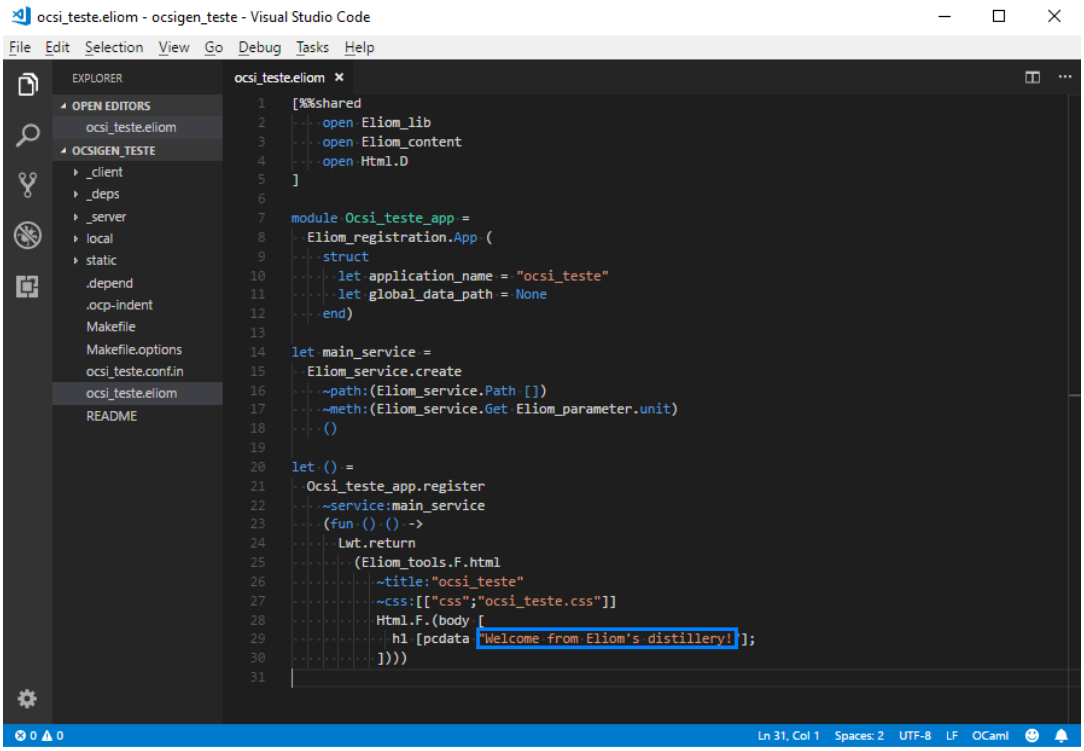


Figura B.20: Ficheiro de código *eliom*.

Existem agora todas as condições para compilar o código e ver resultados. Para compilar um projeto Ocsigen *gerado através da ferramenta* eliom-disttilery*, basta na consola digitar o seguinte comando,* "make test.byte"*, como se pode verificar na imagem (B.21).*

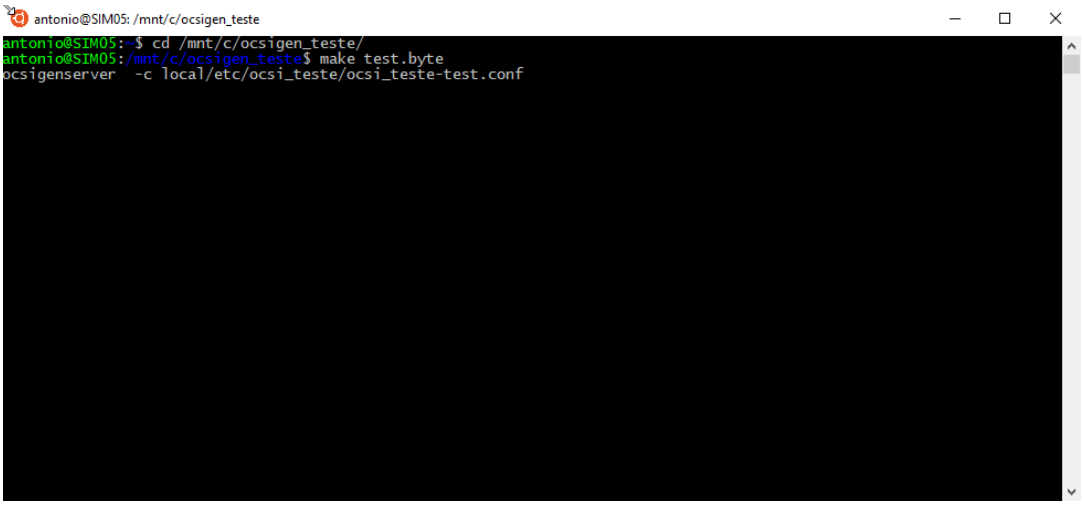


Figura B.21: Compilação do projeto *ocsi_teste*.

Este passo vai compilar o código para bytecode*, no entanto também podia ser compilado diretamente para código nativo alterado o* test.byte *para* test.opt*. O resultado é disponibilizado no endereço* 127.0.0.1:8080*, como se constata na imagem (B.22).*

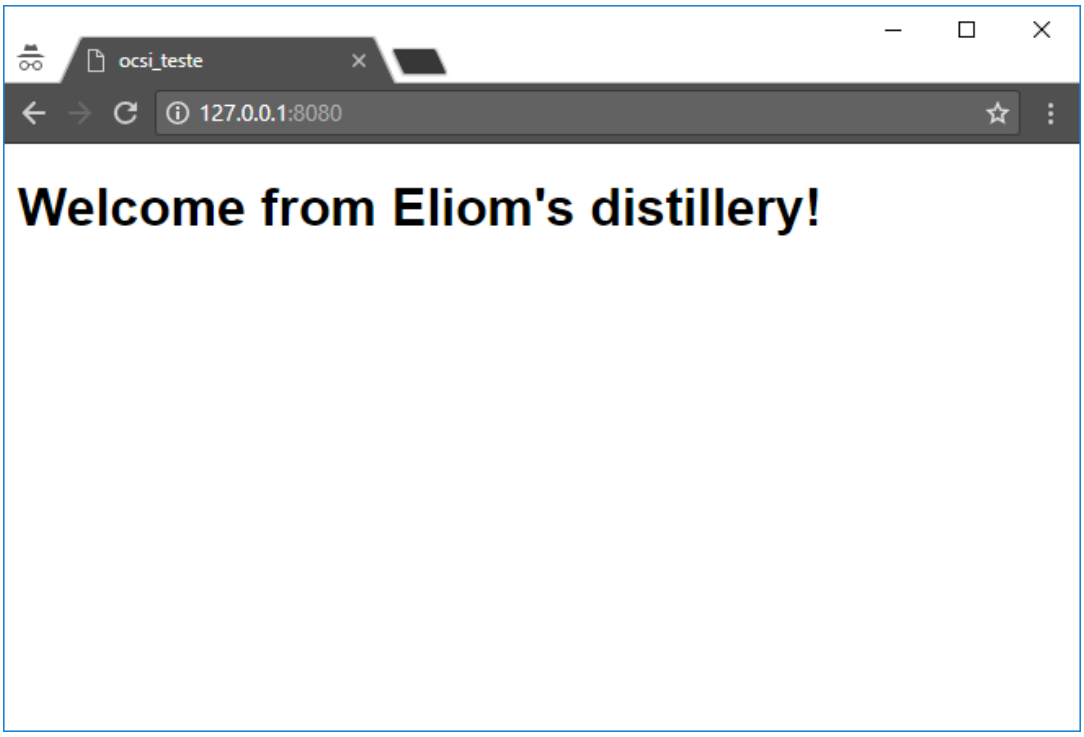


Figura B.22: Resultado do projeto *ocsi_teste* (Parte 1).

É possível editar todo o código que se considere para um projeto. A título de exemplo é alterado o texto do elemento h1, *para* "Hello Ocsigen on Windows!", *como se pode verificar na imagem (B.23).*

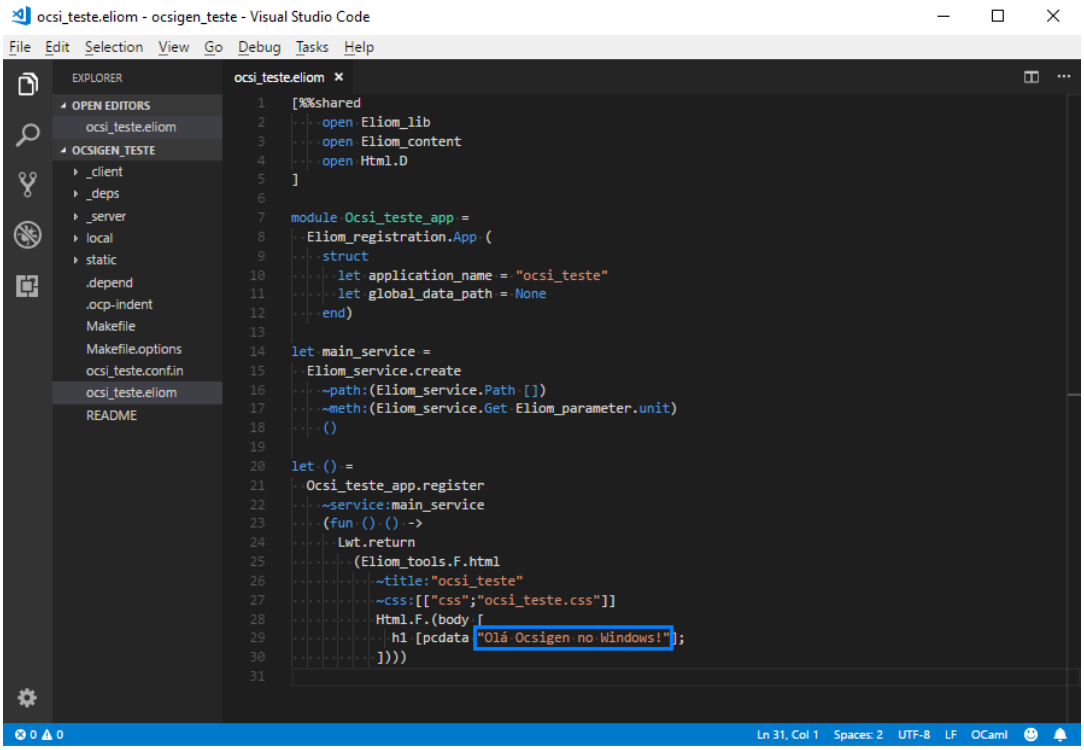


Figura B.23: Nova edição do ficheiro *ocsi_teste*.

Compilando novamente o projeto é então obtido o resultado esperado como se pode constatar na imagem (B.24).

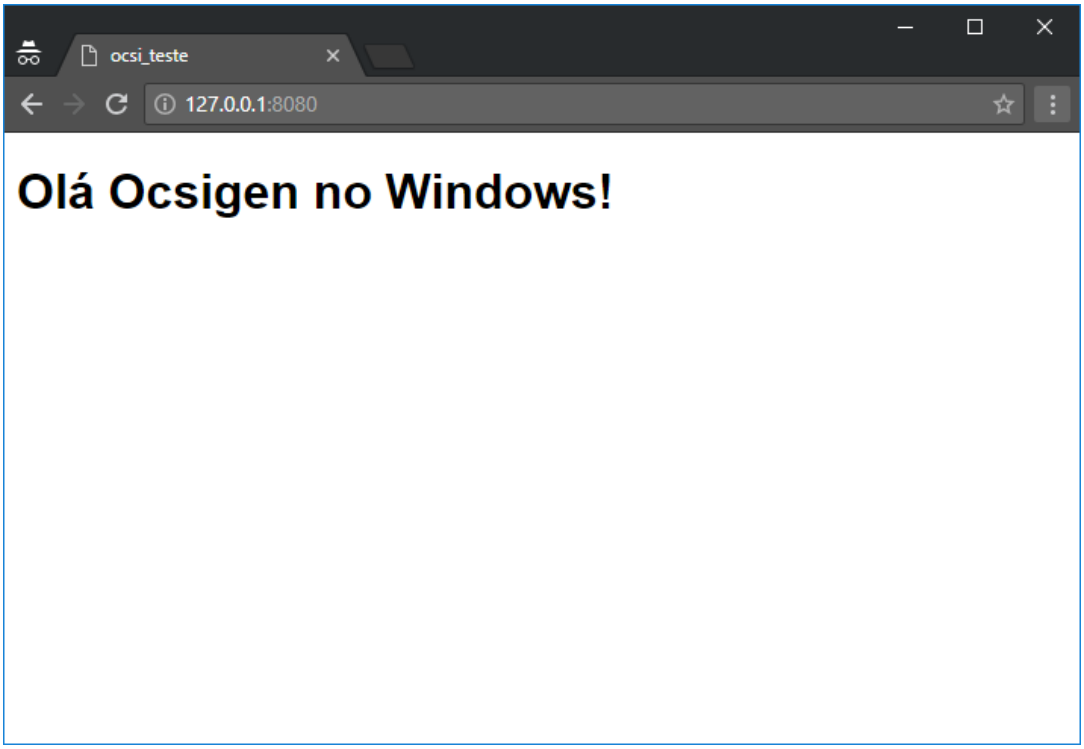


Figura B.24: Resultado do projeto *ocsi_teste* (Parte 2).

B.4 Conclusão

Após uma abordagem prática foram contextualizadas as tecnologias envolvidas e a sua aplicação. Está confirmado que é possível desenvolver projetos com a tecnologia Ocsigen *sobre o sistema operativo* Windows. *Desta forma é aberta uma porta para a sua aplicação em ambiente empresarial, assim como a sua aceitação por parte de programadores que adotaram o sistema operativo* Windows *como ferramenta de trabalho levando a que esta tecnologia seja mais divulgada e utilizada.*

Apêndice C

Tutorial *Ocsi_simple Framework*

C.1 Introdução

Este documento tem como objetivo apresentar o uso da Ocsi_simple Framework. *O documento está estruturado de forma encadeada com os respetivos passos de desenvolvimento. Na secção* Ambiente de Trabalho *(C.2) é descrito como preparar o ambiente de trabalho, este terá como base o sistema operativo* Windows, *pelo que será necessário antes de prosseguir ler o anexo (B). A secção seguinte* Caso de Estudo *(C.3) apresenta o caso de estudo a ser desenvolvido. Este trata da implementação de um sistema de* Login *com recurso a base de dados, sessões e redirecionamento de páginas. Posteriormente é apresentado o resultado com uma breve analise. O resultado pode ser consultado de forma prática, através de* `http://ocsisimple.ddns.net:8085`.

C.2 Ambiente de Desenvolvimento

De forma a realçar a utilização da tecnologia Ocsigen *no sistema operativo* Windows, *este tutorial será desenvolvimento sobre este. Para continuar recomenda-se a leitura do anexo (B).*
O projeto será nomeado por "login_project" *e a imagem seguinte (C.1), apresenta de forma concisa a preparação do ambiente de desenvolvimento para o projeto em causa.*

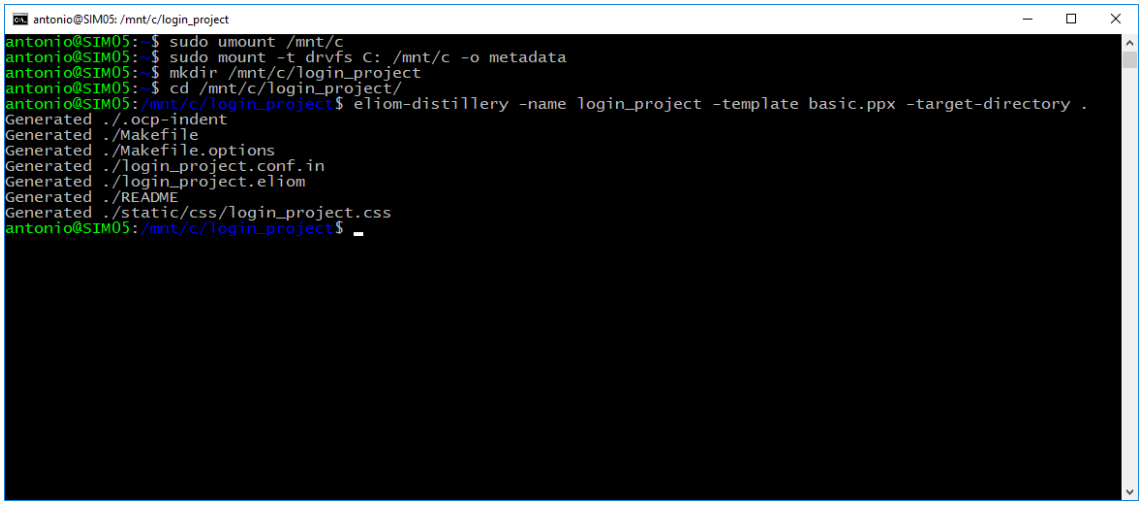


Figura C.1: Configuração de ambiente (Parte 1).

Depois de configurado o ambiente e da criação do projeto é agora essencial atribuir permissões de controlo total à pasta "login_project". *A imagem (C.2), apresenta de forma detalhada este passo.*

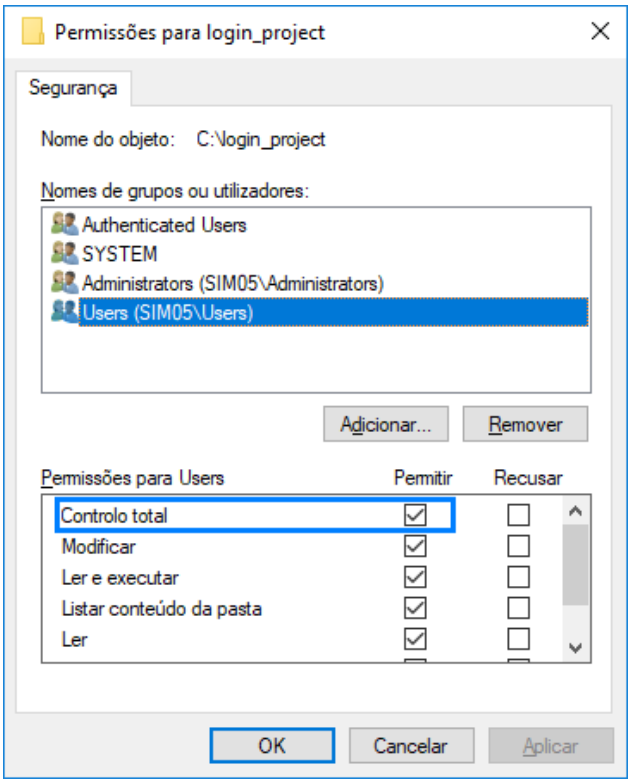


Figura C.2: Configuração de ambiente (Parte 2).

Com as permissões atribuídas é agora possível abrir o projeto com o editor preferido. Neste caso irá ser utilizado o Visual Studio Code*, como é apresentado na imagem (C.3).*

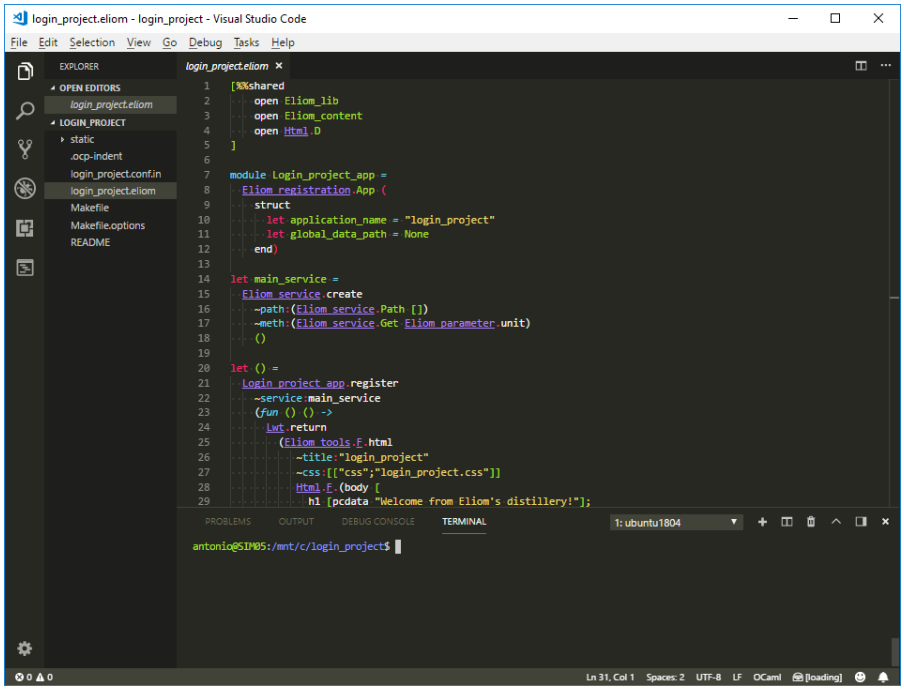


Figura C.3: Configuração de ambiente (Parte 3).

Com a configuração do ambiente de desenvolvimento concluída parte-se agora para a análise e implementação do caso de estudo.

C.3 Caso de Estudo

Neste caso de estudo pretende-se demonstrar o uso da tecnologia Ocsigen *juntamente com a* Ocsi_simple Framework. *Irá ser usado como tema, um sistema simples de autenticação. Este sistema tem como objetivo garantir a autenticação de utilizadores registados numa base de dados* SQlite *acedida através do módulo* Ocsi_simple_persist. *Após o utilizador estar autenticado é criada uma sessão. Esta é destruída após o* logout. *Apesar de simples este caso de estudo envolve o uso da grande maioria da tecnologia* Ocsigen. *Desta forma o leitor deste documento terá uma referência, com a qual pode desenvolver os seus próprio projetos, com base nas tecnologias apresentadas.*

C.3.1 Base de Dados

Para este projeto, de forma a termos persistência de dados será utilizado o SQlite. *A base de dados é composta apenas por uma tabela. O nome da tabela é* users, *e terá como colunas, o* user *e a* password. *A imagem seguinte (C.4) apresenta a criação da base de dados* logindb.db, *com registos inseridos.*

```
antonio@SIM05:/mnt/c/login_project$ sqlite3 logindb.db
SQLite version 3.22.0 2018-01-22 18:45:57
Enter ".help" for usage hints.
sqlite> CREATE TABLE users (user PRIMARY KEY, password);
sqlite> select * from users;
sqlite> insert into users values ("user1","12345");
sqlite> insert into users values ("user2","56789");
sqlite> insert into users values ("user3","15263");
sqlite> select * from users;
user1|12345
user2|56789
user3|15263
sqlite>
```

Figura C.4: Base de dados do projeto.

C.3.2 Configurações

Nesta secção será apresentado numa primeira fase, a configuração necessária à compilação do projeto e posteriormente a explicação do código produzido.
Para permitir a correta compilação do projeto é necessário incluir na raiz, deste os ficheiros correspondestes à Ocsi_simple Framework. *A imagem seguinte (C.5), apresenta os ficheiros a incluir no projeto.*

Figura C.5: Ficheiros da *Ocsi_simple Framework*.

A próxima fase é configurar os ficheiros que vão produzir a aplicação, que vai ser servida pelo Ocsigen Server. *Estas configurações são essenciais para utilizar a* Ocsi_simple Framework. *No ficheiro* Makefile *é necessário adicionar as flags selecionadas, como é apresentado na imagem (C.6).*

```
include Makefile.options

##-------------------------------------------------------------------
##  Internals

## Required binaries
ELIOMC          := eliomc -ppx
ELIOMOPT        := eliomopt -ppx
JS_OF_ELIOM     := js_of_eliom -jsopt +nat.js -jsopt +base/runtime.js -ppx
ELIOMDEP        := eliomdep
OCSIGENSERVER   := ocsigenserver
OCSIGENSERVER.OPT := ocsigenserver.opt
```

Figura C.6: Ficheiro *Makefile*.

A próxima configuração necessária é a alteração do ficheiro login_project.conf.in, *para o qual é necessário adicionar as seguintes linhas selecionadas na imagem (C.7).*

```
<extension findlib-package="eliom.server"/>

<extension findlib-package="postgresql"/>
<extension findlib-package="mysql"/>
<extension findlib-package="sqlite3"/>

<library module="%%LIBDIR%%/ocsi_simple_persist.cma" />

%%% This will include the packages defined as SERVER_PACKAGES in your Makefile:
%%PACKAGES%%
<host hostfilter="*">
```

Figura C.7: Ficheiro *login_project.conf.in*.

Para concluir a configuração de todo o ambiente, a próxima etapa é compilar o projeto e os ficheiros da Ocsi_simple Framework. *Para isso basta digitar os comandos na ordem apresentados na imagem (C.8).*

```
antonio@SIM05:/mnt/c/login_project$ clear
antonio@SIM05:/mnt/c/login_project$ ocamlc -c Imdb.mli
antonio@SIM05:/mnt/c/login_project$ ocamlfind ocamlc -c -custom -thread -linkpkg -package postgresql Mdb_postgresql.ml
antonio@SIM05:/mnt/c/login_project$ ocamlfind ocamlc -c -custom -thread -linkpkg -package mysql Mdb_mysql.ml
antonio@SIM05:/mnt/c/login_project$ ocamlfind ocamlc -c -custom -thread -linkpkg -package sqlite3 Mdb_sqlite.ml
antonio@SIM05:/mnt/c/login_project$ ocamlfind ocamlc -c -custom Mdb_mysql.cmo Mdb_postgresql.cmo Mdb_sqlite.cmo Ocsi_simple_persist.ml
antonio@SIM05:/mnt/c/login_project$ ocamlc -a -o ocsi_simple_persist.cma -linkall Mdb_mysql.cmo Mdb_postgresql.cmo Mdb_sqlite.cmo Ocsi_simple_persist.cmo
antonio@SIM05:/mnt/c/login_project$ make && cp ocsi_simple_persist.cma local/lib/login_project/
```

Figura C.8: Compilação do projeto.

Agora com todo o processo de configuração concluído será apresentada a próxima fase que trata do desenvolvimento do caso de estudo.

C.3.3 Desenvolvimento

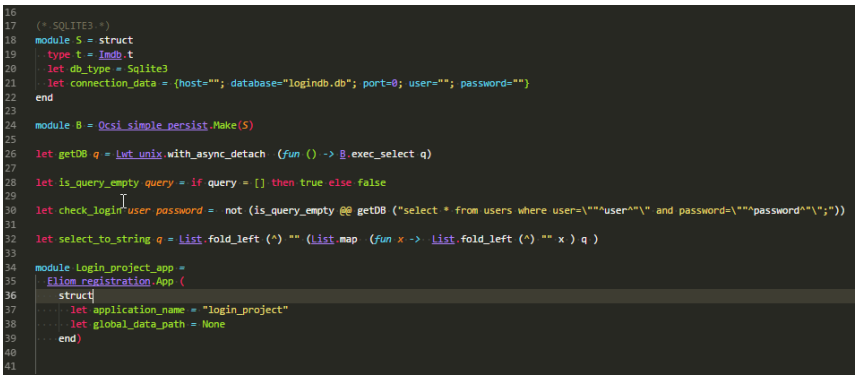
No desenvolvimento de qualquer tipo de aplicação é necessário seguir um padrão de organização. Com a tecnologia Ocsigen *passa-se precisamente o mesmo. Para este caso de estudo, o padrão seguido, tem uma ordem definida. Ordem essa que passa pela, abertura dos módulos necessários, definição de funções auxiliares, definição de referências de contexto, definição de serviços, definição de páginas e blocos HTML e por fim o registo dos serviços e lógica de apresentação das páginas criadas.*

Como se pode verificar na imagem (C.9) é feita a abertura dos módulos necessários. Dentro do bloco %%shared *estão os módulos que vão ser compilados, tanto para* Backend *como para* Frontend*, ou seja, para OCaml e* JavaScript*. Em vez da utilização do* %%shared *é possível utilizar* %%client *e* %%server *para especificar, o tipo de compilação. Quando não se utilizam estes blocos, por definição o código será tratado, como código servidor. Exemplo disso é a abertura do módulo* Ocsi_simple_persist*, que vai tratar das ligações aos SGBD.*

```
1  [%%shared
2      open Eliom_lib
3      open Eliom_content
4      open Eliom_client
5      open Eliom_service
6      open Dom
7      open Dom_html
8      open Eliom_parameter
9      open Html.D
10     open Ocsi_simple
11     open Ocsi_simple_js
12 ]
13
14 open Ocsi_simple_persist
15
```

Figura C.9: Código (Parte 1).

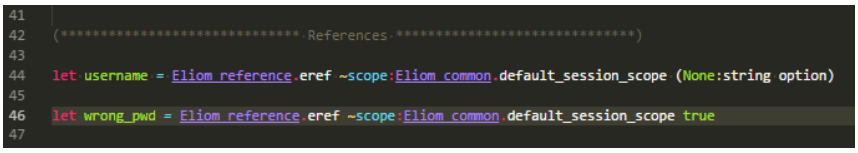
Na imagem (C.10) estão representadas as funções auxiliares e de acesso à base de dados. Primeiramente é criado um módulo com a informação necessária para ligar à base de dados SQLite*. Podia ser para outro SGBD suportado pela* Ocsi_simple Framework *sendo que para isso basta alterar o* db_type *e o record de ligação, identificado por* connection_data*. Em seguida é instanciado o módulo* B*, através da aplicação do módulo recém criado* S*, no funtor* Make *pertencente ao módulo* Ocsi_simple_persist*. É importante notar a função definida na linha 26. Esta função vai criar um mecanismo assíncrono. Isto proporciona que a cada chamada, a aplicação inteira, não é bloqueada durante a execução da* query *de seleção. É importante ter sempre presente este tipo de situações, quando se programa uma aplicação que pode ser acedida, por vários utilizadores em simultâneo. Por fim a partir da linha 34, a aplicação é instanciada.*



```
(* SQLITE3 *)
module S = struct
  type t = Imdb.t
  let db_type = Sqlite3
  let connection_data = {host=""; database="logindb.db"; port=0; user=""; password=""}
end

module B = Ocsi_simple_persist.Make(S)

let getDB q = Lwt_unix.with_async_detach  (fun () -> B.exec_select q)

let is_query_empty query = if query = [] then true else false

let check_login user password =  not (is_query_empty @@ getDB ("select * from users where user=\""^user^"\" and password=\""^password^"\";"))

let select_to_string q = List.fold_left (^) "" (List.map  (fun x ->  List.fold_left (^) "" x ) q )

module Login_project_app =
  Eliom_registration.App (
    struct
      let application_name = "login_project"
      let global_data_path = None
    end)
```

Figura C.10: Código (Parte 2).

A figura (C.11) apresenta a criação de duas referências. As referências servem para criar e guardar determinada informação de contexto. O contexto no âmbito da tecnologia Ocsigen *significa a comunicação entre o servidor e o cliente que consome a aplicação. Cada cliente da aplicação têm um único contexto. Então dentro deste contexto é possível guardar informações ou dados, que apenas podem ser validados e alterados dentro do contexto do cliente. Desta forma, ao utilizar este mecanismo de referências que a tecnologia* Ocsigen *proporciona é possível construir sessões. A referência* username *serve unicamente para guardar o nome de utilizador, enquanto que, a referência* wrong_pwd *inicializada a com valor* true *serve como ponto de verificação, para o inicio de sessões. O uso da referência* wrong_pwd *é parte da estratégia de autenticação que será analisada posteriormente.*



```
(*****************************References*****************************)

let username = Eliom_reference.eref ~scope:Eliom_common.default_session_scope (None:string option)

let wrong_pwd = Eliom_reference.eref ~scope:Eliom_common.default_session_scope true
```

Figura C.11: Código (Parte 3).

A figura (C.12) apresenta a criação de serviços. Na tecnologia Ocsigen*, serviços são os pontos de ação de qualquer aplicação. Os serviços recorrem ao protocolo HTTP, para executar ações consoante o seu tipo, por exemplo pode ser uma ação do tipo* GET *ou* POST*, entre outros. Os serviços são componentes indispensáveis à utilização da tecnologia* Ocsigen*, por isso a sua compreensão é essencial para conseguir produzir o resultado esperado. Os serviços são normalmente divididos em duas fases, a sua definição e o seu registo, porém podem ser automaticamente definidos e registados se forem* Actions*. As* Actions *são serviços especiais que tratam normalmente de requisições* POST*, porém podem tratar de outras situações em que não haja um ponto de entrada especificado. Por exemplo, pode ser definida uma* Action *para executar uma determinada tarefa de* x *em* x *minutos. A figura (C.12) apresenta a criação dos cinco serviços do caso de estudo.*

- *O serviço* main_service *é a raiz da aplicação. Este é criado e definido o caminho que irá escutar. É também definido que o parâmetro de entrada é vazio ou seja do tipo* unit*. Significa isto que ao aceder ao URL, por exemplo* http://127.0.0.1/*, o serviço* main_service *irá ser despoletado.*

- *O serviço* redir_service *tem como objetivo identificar qualquer acesso, quando ao URL é acrescentado um sufixo, por exemplo,* http://127.0.0.1/teste. *Posteriormente na fase de registo será definida a ação de redirecionamento para uma página de aviso.*

- *O serviço* no_disconnection_service *serve unicamente para redirecionar o estado da aplicação para o contexto atual. Na prática quando existe um pedido de* logout *e o utilizador pretende não continuar este é mantido na mesma página. Note-se que este serviço é criado através da função* create_attached_post, *o que significa, que vai utilizar o mesmo URL, para passar a informação necessária durante o seu ciclo de vida.*

- *O serviço* disconnection_service *à semelhança do serviço* no_disconnection_service *também vai tratar do processo de* logout, *porém na fase de registo está definido para limpar o contexto aplicacional da comunicação. O que significa limpar toda a sessão do utilizador ou aplicação cliente.*

- *O serviço* login_service *é especial. Este é uma* Action, *que para além de declarada é também definida. A lógica deste serviço passa por receber quatro parâmetros via* POST *e instanciar o contexto aplicacional. Durante este processo são também apresentadas no lado cliente, mensagens do género* alert, *que notificam o utilizador cada vez que ele introduz dados de acesso. Paralelamente as funções das linhas 82 e 88 servem como* log *e vão apresentar as mensagens no lado do servidor. Este é o serviço que trata do processo de autenticação.*

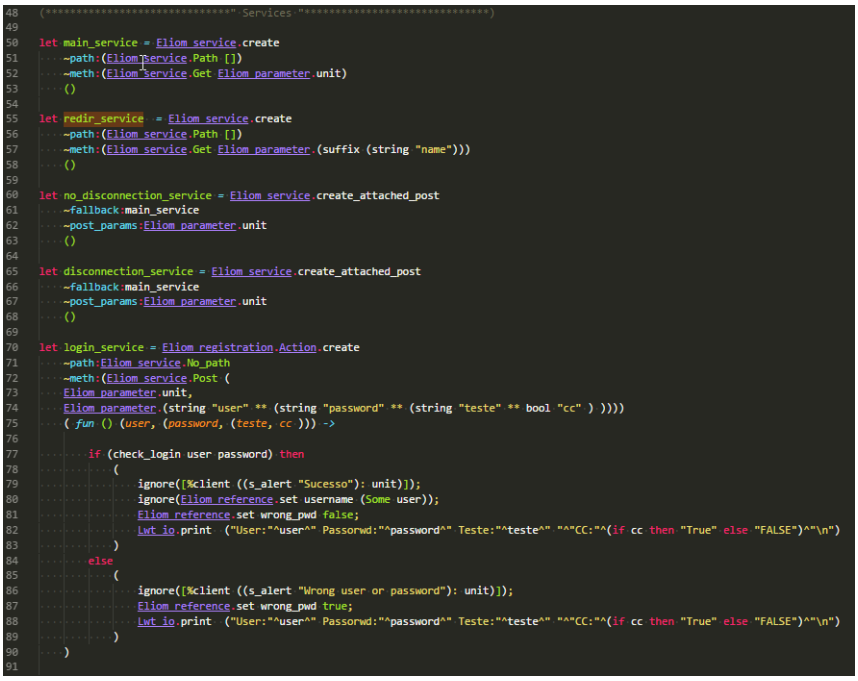


Figura C.12: Código (Parte 4).

A figura (C.13) apresenta a criação da função login_page. *Esta função nada mais é que a definição de uma página HTML. Esta usa o modulo* Ocsi_simple *da* Ocsi_simple

Framework. *Como se pode verificar a criação da página é composta por um título, links locais e externos de CSS, assim como um link externo de* JavaScript. *É importante notar que o uso da* Ocsi_simple Framework *complementa a tecnologia* Ocsigen *tornando-a mais simples de utilizar, garantindo a compatibilidade com as funções originais. Neste caso, verifica-se plenamente o uso conjugado das duas tecnologias. A função* login_page *define o formulário de autenticação, como se pode verificar na linha 107 esta vai enviar pelo método* POST, *do protocolo HTTP, quatro parâmetros, entre os quais esta o nome de utilizador a senha de acesso e mais dois parâmetros correspondentes à* Checkbox Remember me *e um parâmetro enviado de forma oculta, que para o caso de estudo, servem apenas para motivo de demonstração, não tendo propriamente um uso próprio.*

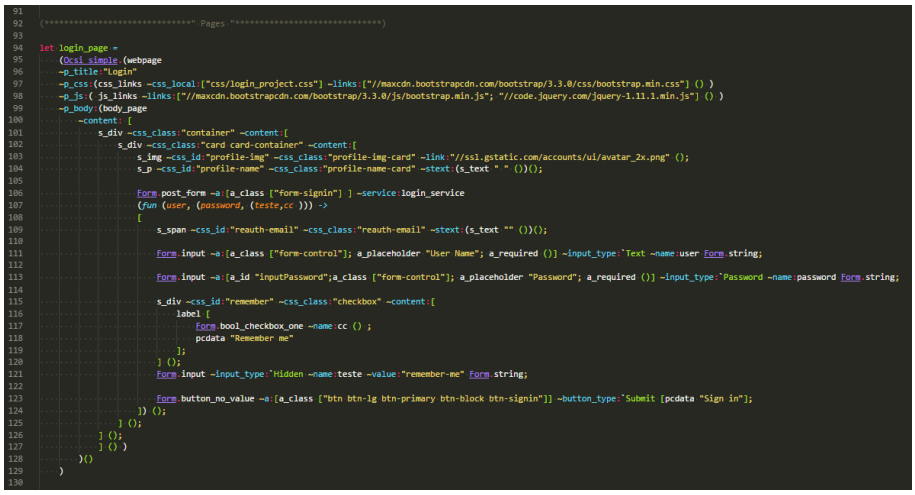


Figura C.13: Código (Parte 5).

A figura (C.14) apresenta a definição da função warning_page, *indo de encontro ao exemplo anterior. Esta página apenas é apresentada, quando existir um acesso indevido ou não autorizado. O acesso é controlado através do URL, por exemplo se for inserido o URL* http://127.0.0.1/teste *a* warning_page *irá ser apresentada. Na linha 140 está criada uma hiperligação que redireciona o utilizador para a* main_page. *Pela razão que o utilizador pode já estar autenticado e dessa forma o redirecionamento será direto, caso contrário é apresentado o formulário de autenticação.*

Figura C.14: Código (Parte 6).

A figura (C.15), apresenta a função logout_block *que define um bloco HTML, composto por uma* Div, *ma qual estão presentes dois botões especiais que utilizam os serviços previamente criados, para o efeito de* logout *e de permanência na página.*

```
let logout_block =
    Ocsi_simple.(
        s_div ~css_id:"logout_block" ~css_class:"alert alert-warning center" ~content:[
            s_h3 ~stext:(s_text "Logout ?") ();
            Ocsi_simple.simpleForm ~title:"Logout: " ~btnLabel:"Não" no_disconnection_service;
            Ocsi_simple.simpleForm ~title:"Logout: " ~btnLabel:"Sim" disconnection_service
        ] ()
    )

```

Figura C.15: Código (Parte 7).

A figura (C.16) apresenta a função main_page*, a qual tens desde logo a particularidade de na linha 158 definir uma função que no lado cliente, vai acrescentar o bloco* logout_block*, quando executada através do botão definida na linha 170. É desta forma que na tecnologia* Ocsigen *se podem manipular elementos HTML, criando um dinamismo igual, ao usado com* Javascript standard.

```
let main_page =
    let logout_handler = [%client (fun _ ->  appendTo "logout_div" ~%logout_block)]
        in
    Ocsi_simple.(webpage
    ~p_title:"Caso de Estudo"
    ~p_css:(css_links ~css_local:["css/login_project.css"] ~links:["//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/css/bootstrap.min.css"] () )
    ~p_js:( js_links ~links:["//maxcdn.bootstrapcdn.com/bootstrap/3.3.0/js/bootstrap.min.js"; "//code.jquery.com/jquery-1.11.1.min.js"] () )
    ~p_body:(body_page
        ~content: [
            s_div ~css_class:"container" ~content:[
                s_div ~css_id:"success_div" ~css_class:"alert alert-success center" ~content:[
                    s_h1 ~stext:(s_text "Welcome !") ();
                    s_div ~css_id:"logout_div" ~css_class:"alert alert-success center" ~content:[
                        button ~a:[a_onclick logout_handler] [pcdata "Logout"]
                    ]();
                ] ();
            ]()
        ] ()
    )()
)
```

Figura C.16: Código (Parte 8).

Por fim a figura (C.17) apresenta a conclusão da implementação da aplicação de caso de estudo. Nesta está definida a ultima parte, que trata do registo dos serviços previamente criados. Segue agora a sua descrição.

- *O serviço* redir_service *é registado através do módulo* Any*, como resposta vai enviar uma página HTML a* warning_page*, com o código de erro 404 do protocolo HTTP, que significa que a página não existe.*

- *O serviço* disconnection_service *é na realidade uma* Action*, que vai limpar o contexto da ligação que o executa.*

- *O serviço* no_disconnection_service*, nada mais é que o redirecionamento para a página* main_page*.*

- *O serviço* main_service *é o que implementa o fluxo do caso de estudo aqui desenvolvido. Quando o utilizador é ou tenta ser autenticado vai instanciar a referência* wrong_pwd *com um valor booleano, desta forma quando o utilizador é corretamente autenticado é apenas apresentada a* main_page*, o contrário leva à pagina de autenticação.*

```
177
178  (*****************************" Services Resgistration "*****************************)
179
180  let () =
181
182      Eliom_registration.Any.register
183      ~service:redir_service
184      (fun _ name -> Eliom_registration.Html.send ~code:404 warning_page );
185
186      Eliom_registration.Action.register
187      ~service:disconnection_service
188      (fun () () -> Eliom_state.discard ~scope:Eliom_common.default_session_scope ());
189
190      Eliom_registration.Redirection.register
191      ~service:no_disconnection_service
192      (fun () () -> Lwt.return (Eliom_registration.Redirection main_service));
193
194      Login_project_app.register
195      ~service:main_service
196      (fun () () ->
197          Let%lwt wp = Eliom_reference.get wrong_pwd in
198              Lwt.return(
199                  if wp then
200                      login_page
201                  else
202                      main_page
203              )
204      )
205
```

Figura C.17: Código (Parte 9).

C.4 Resultado

Por fim na presente secção são apresentados os resultados do caso de estudo. Desta forma seguem as imagens dos diferentes cenários criados, com uma breve explicação. A figura (C.18) apresenta o formulário de autenticação, que é definido pela função login_page.

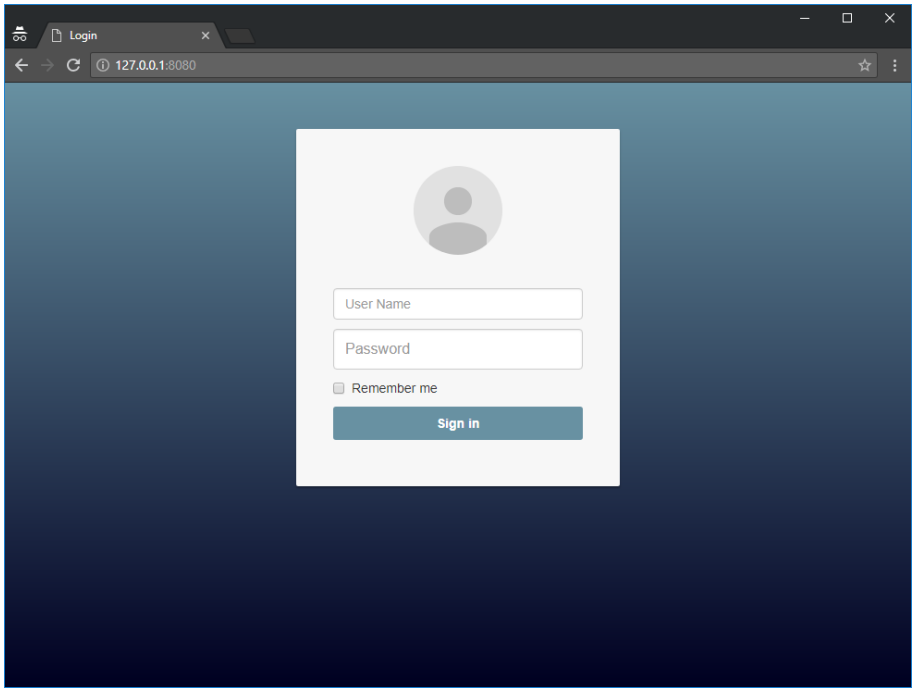


Figura C.18: Resultado (Parte 1).

A figura (C.19) apresenta o formulário de autenticação, com o aviso dos campos requeridos para realizar um envio correto da informação, presente nos respetivos campos.

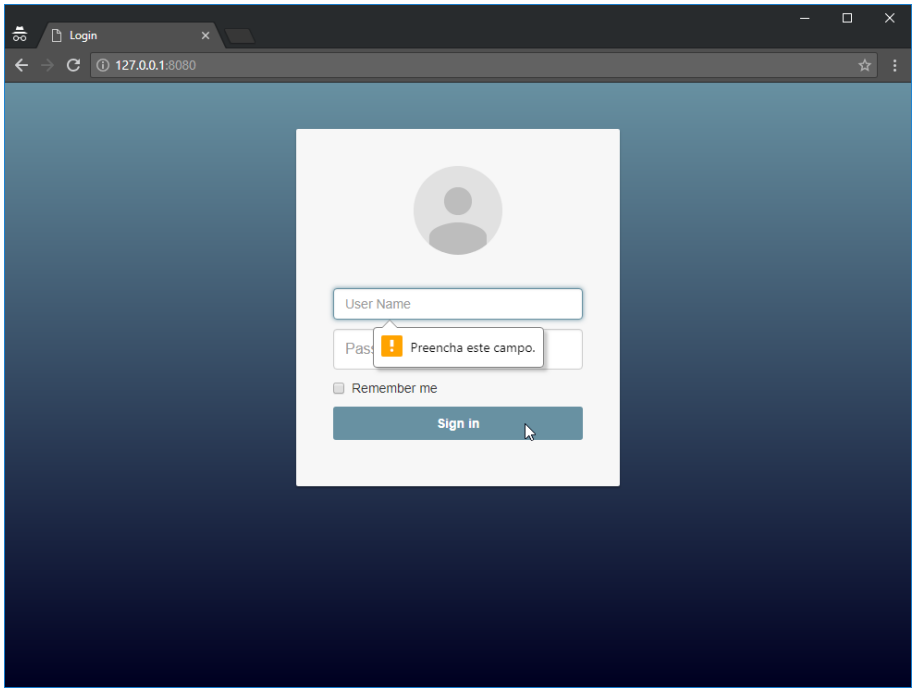


Figura C.19: Resultado (Parte 2).

A figura (C.20) apresenta o formulário de autenticação, a Checkbox Remember me *selecionada. Esta opção apenas se encontra disponível para efeitos demonstrativos pelo que o seu comportamento pode ser modelado, para assumir outras situações.*

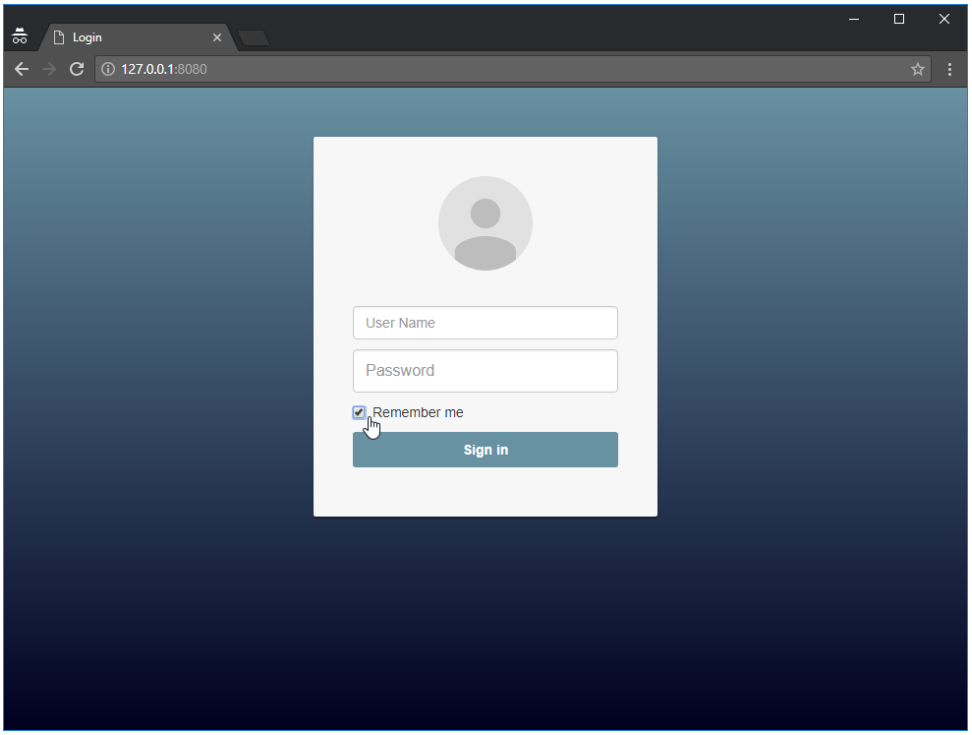


Figura C.20: Resultado (Parte 3).

A figura (C.21) apresenta o formulário de autenticação, com a mensagem de alerta de utilizador ou palavra passe desconhecidos.

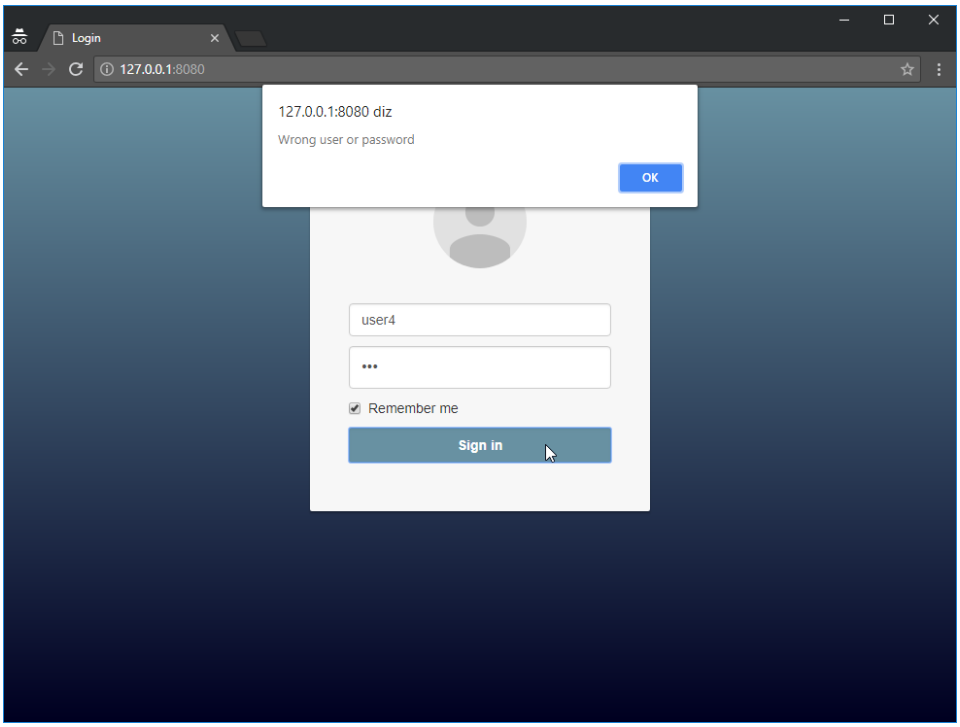


Figura C.21: Resultado (Parte 4).

A figura (C.22) apresenta o formulário de autenticação, com a mensagem de sucesso, quando a autenticação é validada com sucesso.

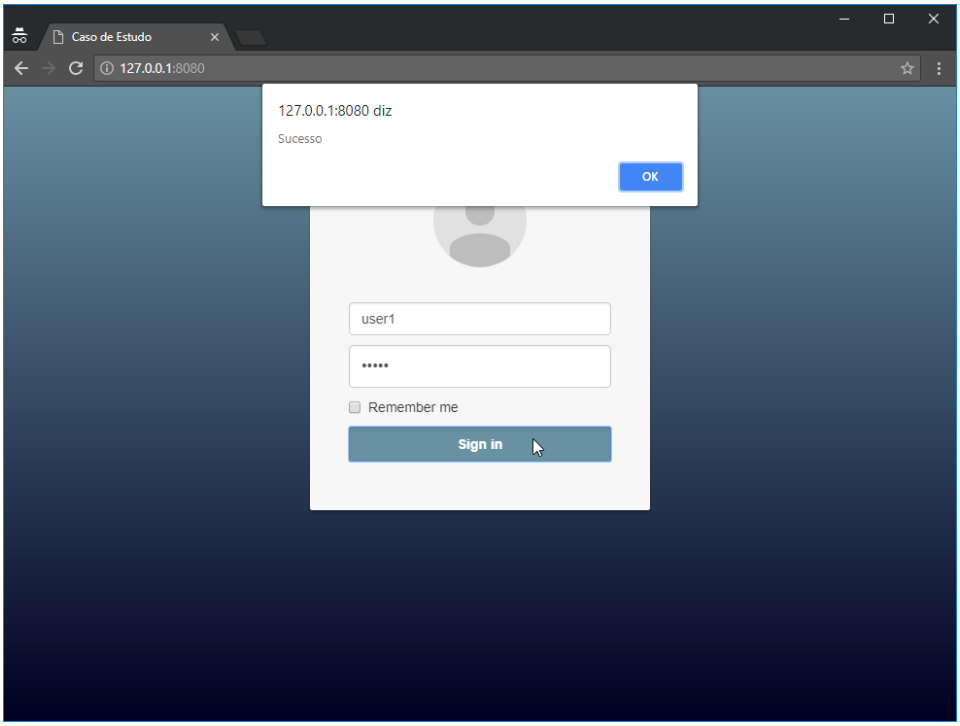


Figura C.22: Resultado (Parte 5).

A figura (C.23) apresenta a área de sessão de um utilizador autenticado com sucesso. Nesta área é possível clicar no botão logout.

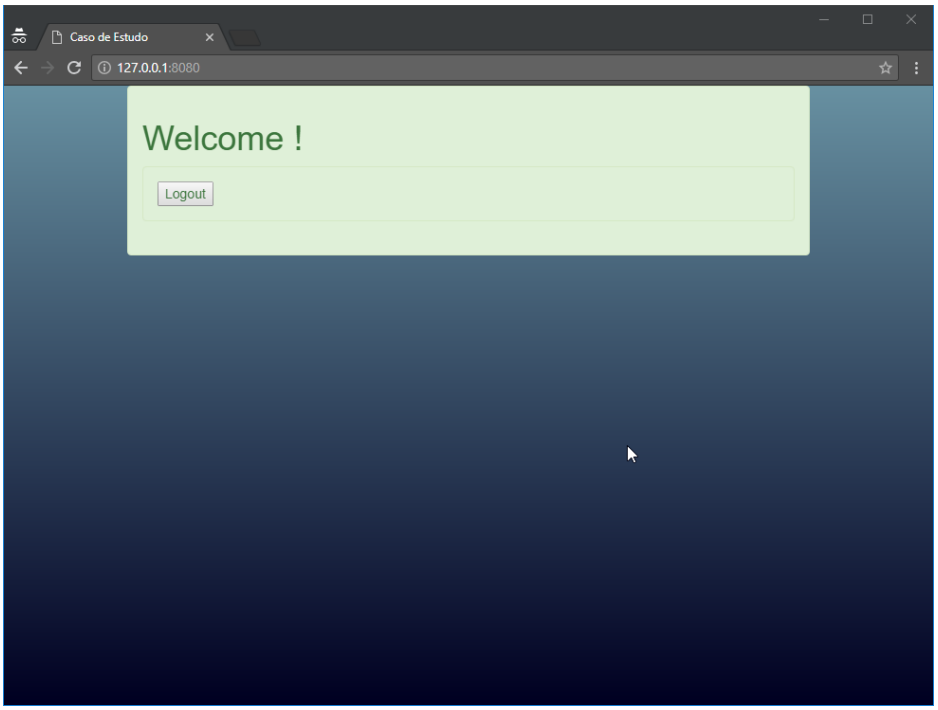


Figura C.23: Resultado (Parte 6).

A figura (C.24) apresenta a área de sessão de um utilizador autenticado com sucesso, quando este clica no botão logout *são apresentas duas possibilidades. A de realizar efetivamente o* logout *e voltar para o formulário de autenticação ou então permanecer na sua área de utilizador.*

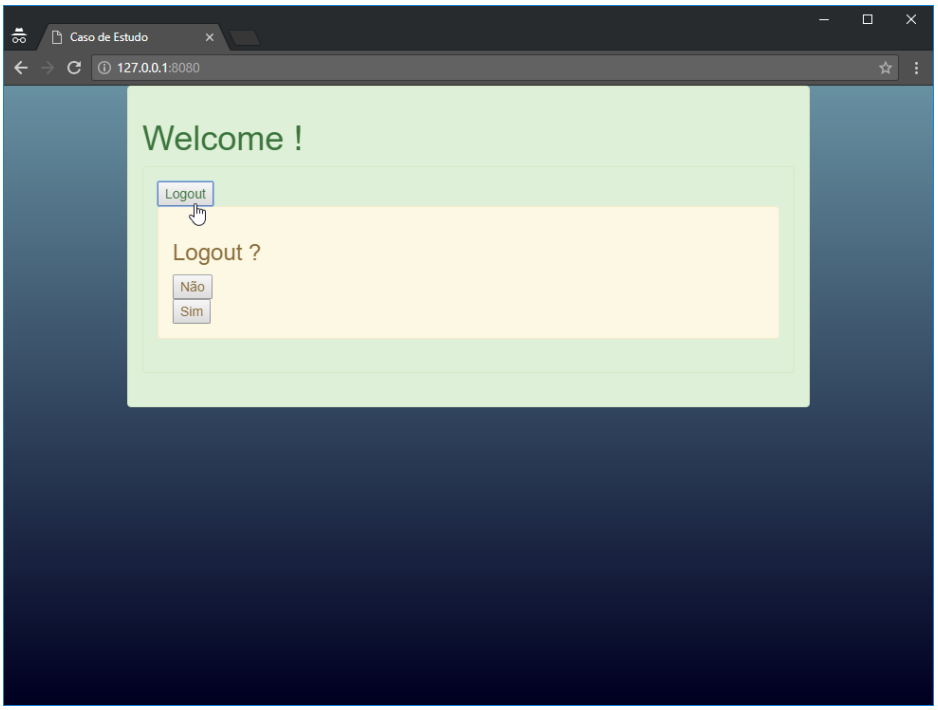


Figura C.24: Resultado (Parte 7).

A figura (C.25) apresenta a página warning_page*, a qual só é acessível caso p utilizador coloque no URL um endereço que não existe. Esta página apresenta também uma ligação que pode ser utilizada para voltar à página inicial.*

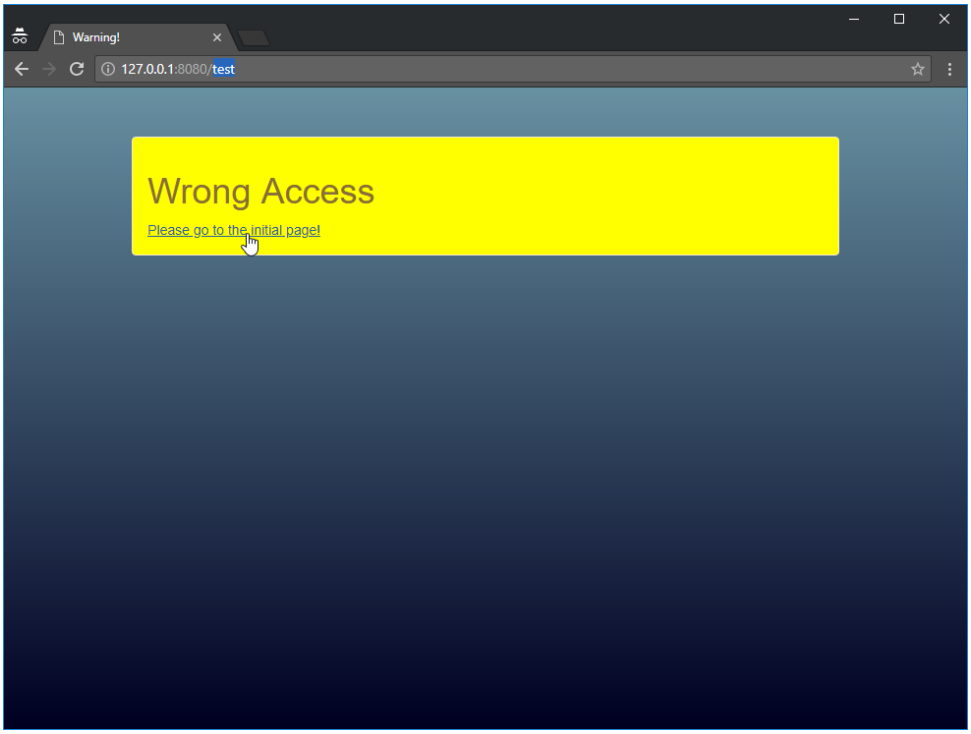


Figura C.25: Resultado (Parte 8).

C.5 Conclusão

Após uma análise sobre a anatomia de uma aplicação Ocsigen*, a sua forma de desenvolvimento e as suas particularidades conclui-se o objetivo deste documento. Agora é possível utilizar a plataforma* Ocsigen *juntamente com* Ocsi_simple Framework *proporcionando uma mais valia relevante ao desenvolvimento seguro com estas tecnologias.*