

Análise e Gestão de Horários da L.EIC

Relatório Final do Projeto Integrador

Bárbara Filipa da Silva Carvalho - up202004695

José Pedro Teixeira Ramos - up202005460

Lucas Ferreira de Sousa - up202004682

Luís Tiago Trindade Cabral - up202006464



Licenciatura em Engenharia Informática e Computação

Tutor na U.Porto: Daniel Castro Silva

Data: 17/06/2023

Conteúdo

1	Introdução	1
1.1	Enquadramento	1
1.2	Objetivos e resultados esperados	1
1.3	Estrutura do relatório	1
2	Metodologia utilizada e principais atividades desenvolvidas	3
2.1	Metodologia utilizada	3
2.2	Intervenientes, papéis e responsabilidades	3
3	Desenvolvimento da solução	5
3.1	Requisitos	5
3.1.1	Requisitos funcionais	5
3.1.2	Requisitos não funcionais	5
3.1.3	Restrições	6
3.2	Arquitetura e tecnologias	6
3.2.1	Plataforma	6
3.2.2	Frameworks	7
3.2.3	Dados	8
3.2.4	<i>Parsing</i> e <i>Scraping</i> de HTML	9
3.2.5	Lista de tecnologias utilizadas	9
3.3	Implementação da Solução	10
3.3.1	Web App	10
3.3.2	Gestão de utilizadores	12
3.3.3	Parsing e Base de Dados	12
3.3.4	Edição de Horários - Frontend	15
3.3.5	Edição de Horários - Backend	15
3.3.6	Exportação em Linguagem Natural	16
3.4	Solução desenvolvida	16
3.4.1	Autenticação	16
3.4.2	Criação de Novos Projetos	17
3.4.3	Controlo de Acesso	18
3.4.4	Edição de Horários	18
3.4.5	Exportação de Mudanças em Linguagem Natural	20
3.5	Validação	21
4	Conclusões	22
4.1	Resultados alcançados	22
4.2	Contribuições	22
4.3	Lições aprendidas	22
4.4	Trabalho futuro	23

1 Introdução

A geração de horários é um problema NP-Completo (*nondeterministic polynomial-time complete*) enfrentado por diversas instituições e organizações, maioritariamente a nível do ensino. Assim, encontrar uma configuração eficiente que atenda às necessidades de todos os envolvidos é uma tarefa árdua e complexa, pois envolve considerar uma série de restrições, como disponibilidade de docentes, salas de aula ou turmas. Para além disso, é também necessário salvaguardar o equilíbrio da carga horária dos alunos, evitar sobreposições de disciplinas e garantir a distribuição adequada dos períodos de aula ao longo da semana. Este problema é alvo de estudo académico a nível global, sendo, inclusive, matéria de teses e dissertações de discentes da FEUP (Faculdade de Engenharia da Universidade do Porto). [31] [18] Nesta secção encontra a descrição do problema da geração de horários na FEUP, a razão da necessidade de uma plataforma especializada, os objetivos e resultados da solução desenvolvida e a descrição da estrutura do relatório.

1.1 Enquadramento

A geração dos horários da FEUP é realizada semestralmente, sendo um processo complexo, semi-automatizado, mas que resulta em horários pouco otimizados, sendo depois necessário realizar manualmente um conjunto de alterações para aumentar a qualidade dos horários para alunos e docentes. Estas alterações são efetuadas pelas comissões de horários de cada curso, num processo longo e manual, o qual requer que sejam consideradas múltiplas dimensões, como turmas, docentes, salas, ou até compatibilidade temporal entre aulas. Este trabalho manual torna a identificação de conflitos mais difícil.

Conseguimos ver a utilidade de uma plataforma de gestão de horários, que torne a sua adaptação e a resolução de conflitos um processo mais eficiente. Visto que a gestão de horários nos afeta diretamente como estudantes da FEUP e tendo confiança nas nossas capacidades, sentimos confiantes no seu desenvolvimento.

1.2 Objetivos e resultados esperados

Pretende-se com este projeto melhorar a capacidade de análise e tornar mais célere o processo de ajustes dos horários gerados. Assim, o nosso objetivo seria criar uma plataforma capaz de simplificar as trocas de aulas, fornecendo diferentes perspetivas para a visualização dos horários e, ao mesmo tempo, mostrando os períodos de indisponibilidade dos docentes, turmas e salas. Além disso, procurámos também facilitar a cooperação entre docentes, permitindo uma edição em simultâneo. Com isto, esperamos construir as bases que permitam uma análise e melhoramento de horários mais automatizada e eficiente.

1.3 Estrutura do relatório

Este relatório está subdividido em 3 secções para além desta introdução:

Na primeira, procuramos descrever a metodologia e recursos utilizados durante o desenvolvimento, assim como identificar os membros da equipa, os seus papéis e responsabilidades, e os *stakeholders* do projeto.

Procuramos também sublinhar os requisitos e restrições relevantes, descrever a arquitetura implementada e enumerar todas as tecnologias envolvidas. Tentamos também apresentar a solução

desenvolvida a partir da ótica do utilizador, apresentando um fluxo lógico, e a sua implementação e validação, descrevendo as ações realizadas para validar a solução.

Por fim, sumariamos os resultados alcançados juntamente com as contribuições individuais, lições aprendidas e retrospectiva de todo o projeto.

2 Metodologia utilizada e principais atividades desenvolvidas

Esta secção relata o processo de desenvolvimento seguido, expondo a metodologia seguida. Descreve os intervenientes, os seus papéis e cargos. Finalmente, demonstra as atividades realizadas ao longo do tempo.

2.1 Metodologia utilizada

Por suspeitarmos, previamente, da dificuldade que dar vida a um projeto desta magnitude traria, o processo de desenvolvimento escolhido teve como sistemática o desenvolvimento *Agile* de software, tendo como método mais similar o *Scrum*. Optámos por esta metodologia pela sua abordagem ágil e colaborativa, que permitiu um desenvolvimento incremental e contínuo, adaptando-se a possíveis mudanças que aparecessem durante o progresso e priorizando a satisfação do "cliente". Desta forma, o acompanhamento semanal feito pelo professor Daniel Castro Silva serviu de *daily scrum*, e teve como principal propósito esclarecer dúvidas e necessidades do projeto. Destaca-se também o impacto nos *sprints* semanais, visto que foram dirigidos de acordo com críticas e sublinhados do tutor em mente. As reuniões ocorreram regularmente, à exceção de semanas sobrecarregadas com exames, ou em que o trabalho a apresentar não diferisse substancialmente do da semana anterior.

Deste modo, tornou-se imprescindível a utilização de recursos externos para permitir acelerar e controlar o progresso do projeto. Assim, utilizou-se maioritariamente: *GitHub* [14] para controlo de versão e hospedagem de repositório; *Hackmd* para fazer *brainstorming* e facilitar a edição coletiva; *Discord* para comunicar entre equipa, destacar apontamentos sobre as críticas/conselhos do tutor e perguntas pertinentes e também para partilha de recursos; *Overleaf* para a realização do relatório final; *ChatGPT* para perguntas sobre código.

2.2 Intervenientes, papéis e responsabilidades

A equipa de trabalho consistiu nos quatro elementos referidos na capa. Devido à complexidade do trabalho em mãos, a divisão de tarefas foi crucial para o bom desenvolvimento do projeto. De início, foi necessário desenvolver o *parsing* completo do HTML (*HyperText Markup Language*) fornecido de apoio da página de horários, sendo esta tarefa levada a cabo por todos os membros da equipa.

O *parsing* dividiu-se em quatro partes, docentes, turmas, salas e unidades curriculares. Para cada uma foi necessário extrair informação importante para a população dos horários, por exemplo, nos docentes foi extraído os seus blocos vermelhos, o seu nome, abreviação e número mecanográfico.

Após o término desta componente, foi possível avançar no desenvolvimento da base de dados, sendo mais notório o trabalho do Luís. Em paralelo, iniciou-se também o desenvolvimento de um módulo relativo ao *login*, registo e gestão de projetos, onde foi mais interveniente o José.

A autenticação e o controlo de acesso estão interligados entre si, por essa razão ficou apenas uma pessoa a fazer o seu desenvolvimento. Na autenticação os objetivos eram claros, criar um modelo para os utilizadores e fazer com que fosse possível iniciar sessão a partir de um nome de utilizador e palavra-passe. No controlo de acesso o seu objetivo é controlar quem pode ver e escrever nos projetos.

Após o povoamento da base de dados com a informação relativa ao *parsing*, a criação das páginas de navegação da *web app* tornou-se o principal foco, com destaque para a página de trabalho de um projeto. Uma vez que esta iria apresentar uma elevada complexidade, foi feita uma divisão em

componentes, mais concretamente, barra de navegação, barra de controlo, menu lateral e tabelas de visualização de horários, nas quais se destacam o trabalho da Bárbara e do Lucas.

A barra de navegação permite navegar entre as páginas de autenticação, a página inicial, a página de exporte de mudanças, as páginas de edição de projetos e permite criar novos projetos. A barra de controlo permite, durante a edição de um horário, alterar as vista, em curso, ano, turno, turma e semana. Permite, também mostrar a distribuição de aulas pela semana. A tabela de visualização permite ver e alterar as aulas num dado horário, permitindo movê-las e trocá-las. A barra lateral permite alterar os atributos de uma aula, sendo estes a UC(Unidade Curricular), o dia, a hora de inicio, a hora de fim, os seus docentes, salas e turmas. Mostra, também, depois de uma alteração, os conflitos gerados.

Na parte final do desenvolvimento, houve colaboração entre todos, pois seria necessário interligar todas as componentes previamente desenvolvidas. Ao longo deste processo de desenvolvimento, sublinhamos a supervisão e apoio do tutor Daniel Castro da Silva, crucial no esclarecimento de dúvidas sobre o tema em trabalho e direcionamento do projeto.

Estas etapas podem ser analisadas na figura 1.

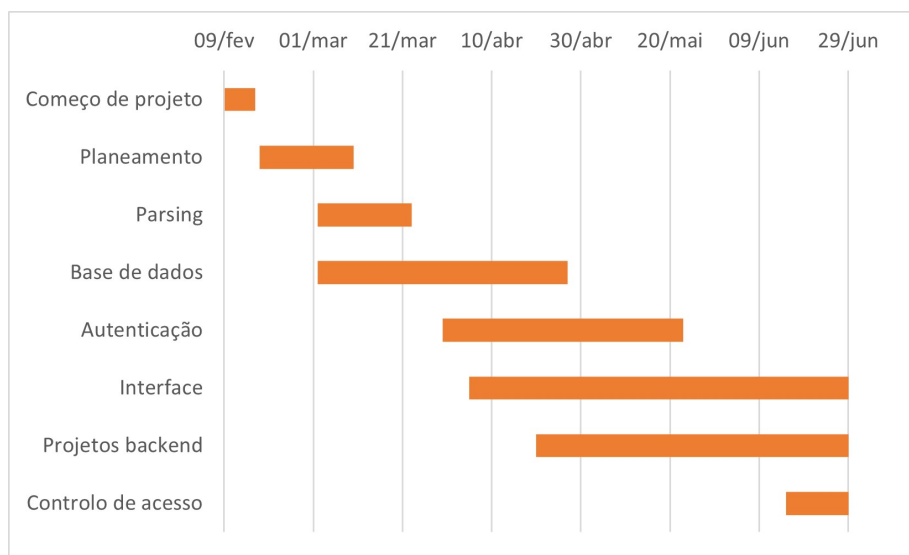


Figura 1: Diagrama de Gantt

Como mencionado anteriormente, decorreram reuniões semanais, nas quais estiveram presentes todos os membros da equipa e o tutor. Nestas reuniões foram discutidos requisitos, restrições, implementações e possíveis ideias a integrar no projeto. Além disso, era analisado o progresso semanal, nomeadamente o trabalho feito nessa semana, que posteriormente era registado numa ata. Estas reuniões ocorreram quase todas as quintas-feiras, na sala I121 ou *online*, às 14:00 ou 15:00 horas.

3 Desenvolvimento da solução

Esta secção descreve a solução desenvolvida, com ênfase nos seus requisitos. Relata as decisões tomadas em relação à arquitetura e às tecnologias utilizadas e descreve a implementação de cada componente, bem como a sua validação.

3.1 Requisitos

Para certificar que o projeto a desenvolver cumpria as necessidades de execução, tivemos de ter em conta diversos requisitos e restrições, de entre os quais destacamos alguns.

3.1.1 Requisitos funcionais

Utilizadores

- O sistema deve deixar os utilizadores editar e visualizar projetos de outras pessoas.
- O sistema deve restringir quem consegue editar e visualizar os projetos.
- O sistema deve enviar um email de confirmação ao utilizador quando ele cria uma conta.

Administradores

- O sistema deve deixar os administradores criarem outros utilizadores.

Projetos

- O sistema deve ser acedido através de um browser.
- O sistema deve permitir a criação de vários projetos de edição.
- O sistema deve guardar as mudanças que os utilizadores fazem nos horários.
- O sistema deve exportar as alterações em linguagem natural.

Horários

- O sistema deve permitir a visualização e edição de horários, por ano, curso e turnos.
- O sistema deve permitir editar aulas em relação à sua hora, duração, dia, UC, salas, docentes e turmas.
- O sistema deve detetar e avisar o utilizador de qualquer conflito.

3.1.2 Requisitos não funcionais

- **Usabilidade:** O sistema deve ser intuitivo e fácil de usar, com uma interface amigável e de fácil navegação.
- **Segurança:** O sistema deve ter medidas de segurança adequadas para proteger os dados e garantir a privacidade dos usuários. Isto inclui autenticação de dados e controlo de acesso.
- **Manutenibilidade:** O sistema deve ser fácil de manter e atualizar, com código limpo e bem documentado.
- **Fiabilidade:** O sistema deve ser confiável e estar disponível a maior parte do tempo.

3.1.3 Restrições

- **Restrições temporais:** O projeto tem data limite para a sua entrega.
- **Restrições de recursos humanos:** Devido a exames, frequências e outros trabalhos, surgiram alturas em que houve pouca disponibilidade para o desenvolvimento do projeto.
- **Restrições de segurança:** Medidas de segurança necessárias para proteger informações sensíveis ou mitigar riscos.
- **Restrições de qualidade:** Expectativas e critérios de qualidade específicos que devem ser atendidos pelo projeto.

3.2 Arquitetura e tecnologias

Para garantir que o projeto seria desenvolvido da maneira mais eficiente possível, houve necessidade de estudar detalhadamente a arquitetura e tecnologias a utilizar, estudo esse que explanamos no tópico seguinte.

3.2.1 Plataforma

Tentámos, de início, decidir face ao tipo de aplicação a desenvolver, estando divididos entre *Desktop App* e *Web App*. Para auxiliar, procurámos informar-nos de vantagens e desvantagens dos dois tipos referidos, tirando as seguintes conclusões.

- **Web app**

Vantagens:

- Acesso a partir de qualquer dispositivo com ligação à internet
- Não requer instalação ou atualizações nos dispositivos
- Possibilidade de implementar atualizações e correções de forma instantânea
- Possibilidade de uso de recursos de nuvem para armazenamento e processamento de dados

Desvantagens:

- Ligação constante à internet
- Desempenho mais lento e menor capacidade de processamento
- Dependência direta da compatibilidade de navegadores web
- Necessidade de *Web-only features*, ex: *login*, proteção contra XSS(*Cross Site Scripting*)

- **Desktop app**

Vantagens:

- Melhor desempenho e capacidade de processamento
- Possibilidade de funcionamento em ambientes *offline*
- Desenvolvimento independente de navegadores *web*

Desvantagens:

- Requer instalação e atualizações no dispositivo
- Dificuldade de compatibilidade entre diferentes sistemas operacionais e dispositivos
- Acesso remoto dificultado
- Impossibilidade de uso de recursos de nuvem
- Menor alcance

Contudo, um dos fatores importantes sublinhados pelo tutor do projeto foi a possível necessidade, futuramente, de desenvolver trabalho colaborativo na aplicação de gestão de horários, em dispositivos diferentes. Uma vez que tal funcionalidade é bastante mais fácil numa *WebApp*, essa acabou por ser a escolha tomada.

3.2.2 Frameworks

Neste seguimento, optámos por nos informar detalhadamente acerca de *frameworks* para *Web Apps*, para nos auxiliar a desenvolver o projeto. Estudámos, de forma análoga, as *frameworks* Django [3], Laravel [17], Ruby on Rails [29] e Flutter for Web [7], optando por utilizar Django, por nos oferecer mais possibilidades em relação às outras opções:

- **Django**

Vantagens:

- Estrutura abrangente
- Forte ênfase na segurança, (prevenção de ataques CSRF (*Cross-Site Request Forgery*) e sanitização de entrada de dados)
- Escalabilidade
- Sistema de administração integrado que permite a criação de interfaces de administração de utilizador de forma simples
- Possui uma grande comunidade de desenvolvedores ativos, o que facilita a procura de ajuda e recursos
- Oferece uma configuração fácil e recursos de desenvolvimento rápido
- Possui soluções pré-feitas para Autenticação, Administração, sessões...
- Arquitetura MVT (*Model-View-Template*)
- Django's Models: camada ORM (*Object Relation Mapping*) que simplifica a interação com a base de dados

Desvantagens:

- Curva de aprendizagem íngreme, especialmente para iniciantes
- Menos flexível do que outros frameworks (personalização)
- Menos eficiente que outros frameworks sobretudo em aplicações com grande volume de tráfego
- Complexidade de configuração e implementação

- Dependência de bibliotecas: depende de bibliotecas externas (módulos Python) para adicionar funcionalidades
- Tamanho de Aplicações Django tendem a ser maiores em comparação com outras *frameworks web*

Apesar de não detalharmos as vantagens e desvantagens das outras *frameworks* consideradas, ao compará-las com o Django, era bastante óbvio que eram menos adequadas às nossas necessidades.

3.2.3 Dados

Como ferramenta de armazenamento e acesso aos dados necessários, optámos por utilizar bases de dados Sqlite3 [10]. Esta escolha deveu-se ao facto de nos fornecer todas as funcionalidades necessárias para o desenvolvimento, de armazenar os dados necessários em ficheiros (facilitando a separação dos projetos de utilizadores), e de ter uma fácil integração em Python e Django. Decidimos, portanto, desenvolver um diagrama UML(*Unified Modeling Language*), aparente na figura 2, para facilitar a representação da base de dados e a construção do esquema relacional daí resultante.

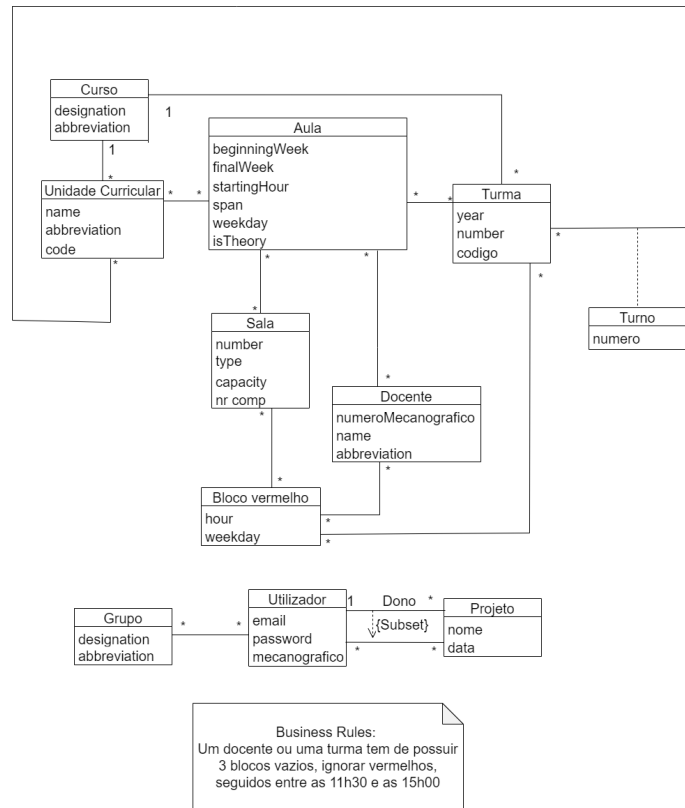


Figura 2: Diagrama UML da base de dados

3.2.4 *Parsing e Scraping de HTML*

Para lidar com o HTML existente na página dos horários optámos por utilizar BeautifulSoup e Requests, uma vez que possuem fácil integração com Python e são extremamente úteis para *web-scraping*.

3.2.5 *Lista de tecnologias utilizadas*

Tecnologias base:

- Django - Web Framework baseada em Python
- Python 3.10
- Pip - Python package manager [21]
- Pipenv - ambient virtual proporcionado por pip [22]

Parsing e Scraping:

- BS4(*Beautiful Soup 4*) - módulo python Beautiful Soup 4 para parsing de HTML em python [19] [2]
- requests - módulo python para fazer pedidos web [24]
- unicode - módulo python com suporte para caracteres unicode [25]

Front end:

- Bootstrap5 - toolkit frontend [1]
- jQuery - Biblioteca de funções JavaScript que interagem com páginas em HTML [9]
- jQuery - Dropdown - plugin para jQuery com soluções para multiselecs em formato dropdown [15]

Armazenamento de Dados:

- Sqlite3 - sistema de base de dados sql [10]
- readerwriterlock - módulo python para controlo de acesso paralelo [23]
- networkx - módulo python para representação de grafos [26]

Deployment:

- ASGI(*Asynchronous Server Gateway Interface*) - interface entre servidores Python com capacidades assíncronas, frameworks e aplicações [4]
- Daphne - servidor ASGI para UNIX, baseado em Python [20]

3.3 Implementação da Solução

Em seguida está descrita implementação de cada componente da aplicação, bem como as decisões que as influenciaram.

3.3.1 Web App

Como referido anteriormente, a *Web App* tem como base a framework Django. Segue um modelo MVT (*Model-View-Template*), semelhante ao MVC (*Model-View-Controller*), é baseada em Python, usando-o para gerir rotas URL(*Uniform Resource Locator*) a funções (*Views*), que retornam dados em JSON(*JavaScript Object Notation*), ou páginas completas, usando *templates*, onde se pode misturar HTML com Python.

A aplicação está subdividida em módulos, representados na figura 3, cada um responsável pelas suas componentes, tais como *routing* de pedidos, acesso à base de dados, lógica a efetuar e páginas ou JSON a retornar. Temos módulos para a autenticação, para o *parsing* e para a páginas da aplicação em si.

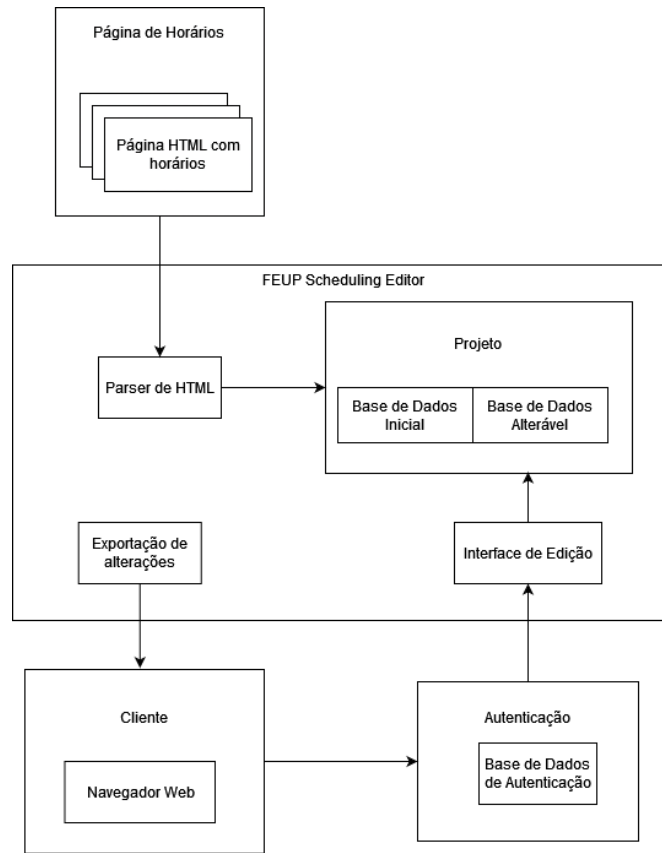


Figura 3: Diagrama da Arquitetura da Aplicação

Devido à capacidade da aplicação poder guardar vários projetos de edição de horários, foi decidido criar, no diretório *database*, um diretório por projeto criado. Aqui são guardadas dois ficheiros de base de dados Sqlite e um ficheiro JSON.

Ambas as bases de dados são povoadas, identicamente, pelo parsing. Um mantém os dados originais, sem nunca serem alterados, enquanto o outro vai sendo alterado durante a utilização da aplicação pelo utilizador. Desta forma, conseguimos compará-las e obter as diferenças efetuadas. O ficheiro JSON guarda um mapa que contém os conflitos criados durante a edição pelo utilizador.

No diretório *templates*, encontram-se ficheiros que descrevem as páginas a serem geradas e as suas componentes, divididas em:

- **admin**, para as páginas relacionadas com os administradores
- **common**, para componentes de páginas partilhadas, como a base e a *header*
- **login**, para as páginas de *login* e *logout*
- **starter**, para a página inicial onde se interage com os projetos

- **editTurnos**, para a página de edição

A aplicação é hospedada numa máquina virtual Ubuntu na FEUP, corre no ambiente virtual *pipenv* fornecido pelo Django e usa Daphne, um servidor ASGI baseado em Python, tal como é aconselhado na documentação Django.

3.3.2 Gestão de utilizadores

Com a ajuda da *framework* do Django, a implementação de um sistema simples de autenticação foi relativamente fácil. A *framework* já tinha quase tudo pré implementado, mas para a nossa *web app* a autenticação do Django não era suficiente para alguns dos nossos objetivos, por exemplo, criar utilizadores com um email já pré definido, que não era possível com a autenticação da *framework*. A solução a que nós chegamos foi recriar o utilizador e o seu formulário na página do administrador, isto foi possível através do *Custom user*, que é praticamente igual a autenticação original do Django.

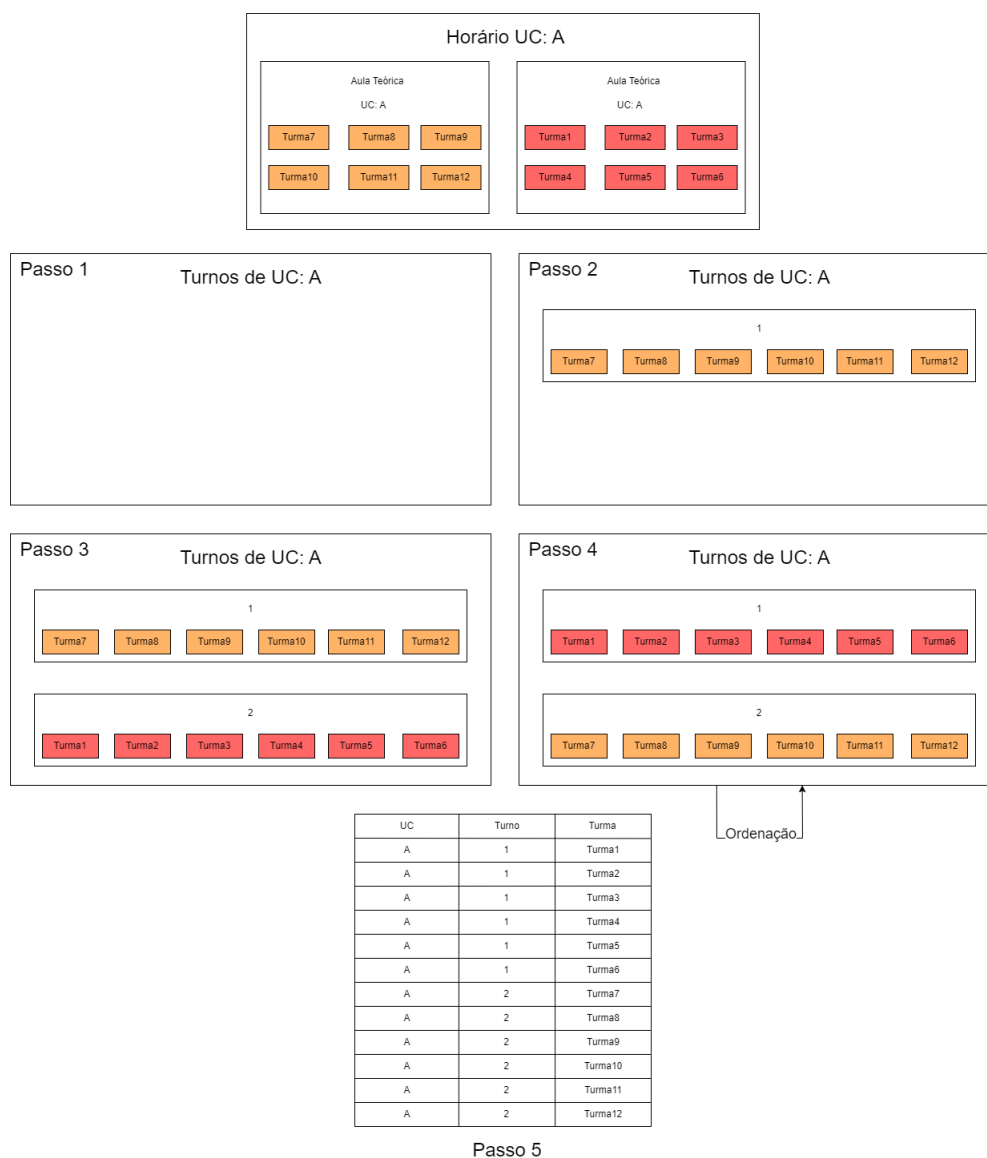
A base de dados dos utilizadores é armazenada internamente no Django e ela pode ser dividida em três partes, Grupos, Pessoas e Projetos. As Pessoas são os utilizadores, os Grupos são grupos de Pessoas e os Projetos têm um nome, que é usado para identificar os projetos, e o controlo de acesso, que são referencias a Pessoas e Grupos.

Tal como foi dito anteriormente, a nossa *web app* tem uma página dedicada apenas para o administrador. Nesta página é possível gerir os utilizadores e mandar emails de confirmação. Mandar o email de confirmação é obrigatório, pois não há maneira de um utilizador comum criar um utilizador. Portanto, o que acontece quando o administrador manda um email a alguém, a primeira coisa é que a palavra-passe é redefinida para uma aleatória temporária, depois é mandado o email com o nome de utilizador e a palavra-passe do utilizador, em conjunto com um link de ativação de conta, no qual usa um *token* para que seja possível identificar o utilizador, após fazer isso a conta do utilizador passa a ser confirmada e é-lhe atribuído uma pessoa com referência ao seu utilizador. Para ajudar o administrador a mandar os emails de confirmação, foi adicionada uma confirmação se o utilizador já recebeu um email, portanto mesmo que se selecione alguém que já recebeu email, não vai ser enviado email para essa pessoa.

3.3.3 Parsing e Base de Dados

Concluimos que seria necessário armazenar a informação proveniente do *parsing* numa base de dados. Para isso, começámos por analisar a informação no HTML fornecido, notando que o nível de organização da mesma nos dificultaria imenso o processo de *scraping*. Um dos primeiros problemas detetados foi a existência de *nested lists*, algo que forçou um aumento do número de ciclos de pesquisa e, consequentemente, do tempo necessário para o *parsing*. Para isto, optámos por dividir o *parse* em etapas, para facilitar o povoamento de cada tabela. Tomámos a decisão de povoar cada tabela principal com uma função distinta (Docentes, Turmas, Unidades Curriculares), sendo a função utilizada para as Unidades Curriculares também usada para povoar a tabela onde se armazenaria a informação das aulas. Desta forma, de cada vez que, por exemplo, um docente fosse analisado, a sua informação era guardada na tabela correspondente de imediato, otimizando a velocidade da aplicação. Relativamente ao armazenamento da informação das salas, o processo foi um pouco diferente, uma vez que o HTML não possuía toda a informação necessária. Neste seguimento, informámo-nos junto do tutor sobre o tipo de salas existente na FEUP, tendo-nos sido facultadas informações que convertimos num ficheiro de texto, presente no diretório "*parser*". Deste modo, confrontámos a informação obtida pela análise do HTML com a informação no ficheiro

de texto para conseguirmos armazenar corretamente o tipo e/ou lotação de cada sala, pois seriam atributos importantes no decorrer do desenvolvimento. O processo de guardar informação relativa aos turnos de uma dada Unidade Curricular revelou-se mais trabalhoso, e complexo, uma vez que não existia nenhuma informação diretamente relacionada com esta entidade. Tendo em conta que o surgimento de turnos é diretamente causado pela existência de aulas teóricas, a forma de obter essa informação foi substancialmente diferente das explicadas anteriormente. Achámos necessário criar uma função especificamente para o povoamento da tabela Turno, dadas as circunstâncias referidas. Começámos por, para cada aula teórica encontrada no *parsing* de uma dada Unidade Curricular, guardar informação referente às turmas presentes nessa dada aula e atribuir a esse conjunto de turmas, um número provisório. Quando a próxima aula teórica fosse encontrada, caso o conjunto de turmas (turno temporário) fosse diferente do já registado, ser-lhe-ia atribuído um número provisório diferente (geralmente o anterior, incrementado por um). Assim que o *parse* de toda a informação e aulas de uma dada Unidade Curricular fosse terminado, seriam analisados todos os turnos temporários dessa mesma Unidade, e colocados numa estrutura de dados comum, onde cada um seria um item. Após tal, seriam ordenados com base no número das turmas que deles fizessem parte, tal que um turno com as turmas 1.LEIC01, 1.LEIC02, 1.LEIC03, fosse colocado primeiro, relativamente a um turno com as turmas 1.LEIC04, 1.LEIC05, 1.LEIC06, sendo reatribuídos os números previamente colocados. Por último, para cada uma das turmas em cada turno, seria inserida na tabela uma entrada com o nome dessa turma, o número atribuído ao turno ao qual pertence, e o código referente à Unidade Curricular correspondente, conforme exemplificado na seguinte figura 4.



Passo 1: Inicialização da estrutura comum aos turnos
 Passo 2 e 3: Colocação dos conjuntos de turmas na estrutura
 Passo 4: Ordenação dos turnos com base nas turmas que os compõem
 Passo 5: Povoamento da Tabela

Figura 4: Diagrama de explicação do algoritmo de povoamento da tabela turnos

Apesar deste esforço, continuou a persistir um problema: A existência de turmas sem qualquer aula teórica no seu horário impedia-as de pertencer a um qualquer turno, impedindo a visão dos

horários por turnos, algo fulcral na aplicação proposta e necessária para da exibição da informação. Para isto, optámos por criar uma outra função, executada após todo o processo de *parsing*, que analisasse todas as turmas não pertencentes a qualquer turno, e lhes atribuisse um número de turno igual a 0, (assim escolhido para evitar causar conflitos com turmas do mesmo curso já presentes em turnos). Deste modo, continuaria a ser possível apresentar na interface toda a informação existente na base de dados, sem prejudicar as diferentes visões, tão importantes neste projeto.

3.3.4 Edição de Horários - Frontend

Para o carregamento da página de edição de um projeto, a quantidade de informação a recolher da base de dados é muito elevada, uma vez que envolve todas as tabelas existentes. De forma a facilitar este processo, foram criados modelos de objetos em Python, nomeadamente, Curso, Ano, Docente, UC, Aula, Sala, Bloco, os quais continham estruturas de dados como listas e dicionários que armazenariam dados mais específicos a cada objeto. Posteriormente, estes objetos são convertidos em objetos do tipo JSON capazes de serem enviados em resposta a pedidos. Para o acesso à base de dados neste módulo, foram criadas várias funções, organizadas em ficheiros no diretório `getHorariosFromDB`, que recolhem e retornam as informações pretendidas, consequentemente armazenadas nos objetos referidos anteriormente. A página de edição de horários é carregada juntamente com duas páginas parciais: `schedule.HTML` e `vista_ucs_salas.HTML`. Deste modo, é efetuado um pedido AJAX que obtém como resposta o HTML relativo às páginas referidas bem como um objeto Curso convertido em JSON, que encapsula todos os dados necessários ao preenchimento das mesmas. Sempre que é escolhido um curso ou ano através do respetivo botão existente na barra de controlo expansível, é feito um novo pedido AJAX (*Asynchronous JavaScript and XML (Extensible Markup Language)*) de forma análoga à referida anteriormente. Em contraste, os restantes botões não exigem a realização de pedidos, uma vez que não exigem a alteração de informação presente em nenhuma das páginas parciais. Quando obtidos todos os dados, é realizado o processo de preenchimento da tabela principal, da tabela da distribuição e das opções da sidebar, destacando-se 4 funções: `fillUcs`, `fillDocentes` e `fillSalas`, `fillTable`, sendo as três primeiras responsáveis por preencher cada uma das células da tabela principal com informações das aulas. A função `fillTable` é responsável pelo preenchimento da tabela da distribuição, juntamente com a informação necessária à mesma.

3.3.5 Edição de Horários - Backend

Para garantir que as alterações a efetuar no horário executam da forma esperada, foram desenvolvidas funções específicas para cada tipo. Optámos, inicialmente, por desenvolver funções de propósito simples, tendo como objetivo, por exemplo, a adição de um docente ou de uma sala numa dada aula, assim como a troca de docentes ou salas de duas aulas distintas. Estas funções operam sobre a base de dados geral do projeto, através de simples instruções SQLite. Neste seguimento, foi também necessário desenvolver funções que aferissem a existência de conflitos nas alterações efetuadas, tendo por base a análise das indisponibilidades de uma dada entidade (docente, turma, sala) no horário para onde seria alterada. Os conflitos são armazenados e reenviados sempre que existe uma alteração, pois podem ser transversais a várias entidades. Para estas funções foi necessário obter o horário de uma dada entidade, assim como todos os períodos de indisponibilidade, sendo desenvolvidas funções auxiliares com esse fim específico.

3.3.6 Exportação em Linguagem Natural

Para as alterações serem exportadas para o utilizador em linguagem natural, é feita, inicialmente, uma comparação entre a base de dados após as alterações e a base de dados antes das mesmas, sendo estas organizadas por tabela. Após obter a informação organizada, são enviados os argumentos necessários para a formatação da mensagem, referindo as principais alterações efetuadas numa dada entidade.

3.4 Solução desenvolvida

3.4.1 Autenticação

Para o utilizador a primeira página (Figura 5) que ele consegue aceder é a página do *login*, esta página é composta por duas partes, uma dedicada à autenticação, e outra caso o utilizador tenha se esquecido da sua palavra-passe ou nome de utilizador, ele pode clicar no link "Esqueci-me da palavra-passe".

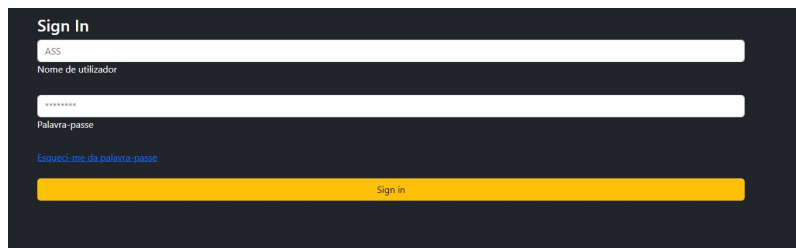


Figura 5: Página de *Login*

Não existe maneira de um utilizador comum criar uma conta, esta tarefa cabe ao administrador, através da página de administração, visível na figura 6, no qual ela mostra vários elementos da base de dados criada para os utilizadores.

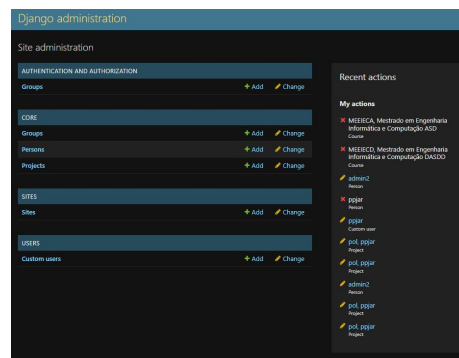


Figura 6: Página inicial do administrador

Ao clicar no *Custom users* aparece uma tabela com todos os utilizadores, e um botão "*add custom user*", que redireciona para um formulário para criar um novo utilizador. Depois de criar o utilizador é obrigatório mandar um email de confirmação para o utilizador, para fazer isto basta seleccionar *Send confirmation email* no input action e clicar no *go*.

Quando o utilizador abre o email e clica no *link* a sua conta é ativada e é redirecionado para mudar a sua palavra-passe. Caso o utilizador se esqueça da palavra-passe enquanto tenta fazer o *login*, ele tem a opção de recuperar/mudar a palavra-passe clicando no "Esqueci-me da palavra-passe", ao clicar no botão o utilizador é redirecionado para outra página, em que lhe é pedido o email. Se o email que foi submetido existir, o utilizador recebe um email, com o link para mudar a palavra-passe. Caso o utilizador queira mudar a sua palavra-passe enquanto tem a sessão iniciada, ele tem de clicar no botão de "mudar palavra-passe".

Quando o utilizador acabar de usar o sistema ele pode fazer *logout*, para sair da sua conta.

3.4.2 Criação de Novos Projetos

Para criar um novo projeto, o utilizador necessita de clicar no botão "Novo Projeto", na barra do topo, ou no botão com um "+", na página principal. Estes ativam um *pop-up* onde o utilizador terá de fornecer um nome e o *link* da página onde se encontram os horários, visível na figura 7.

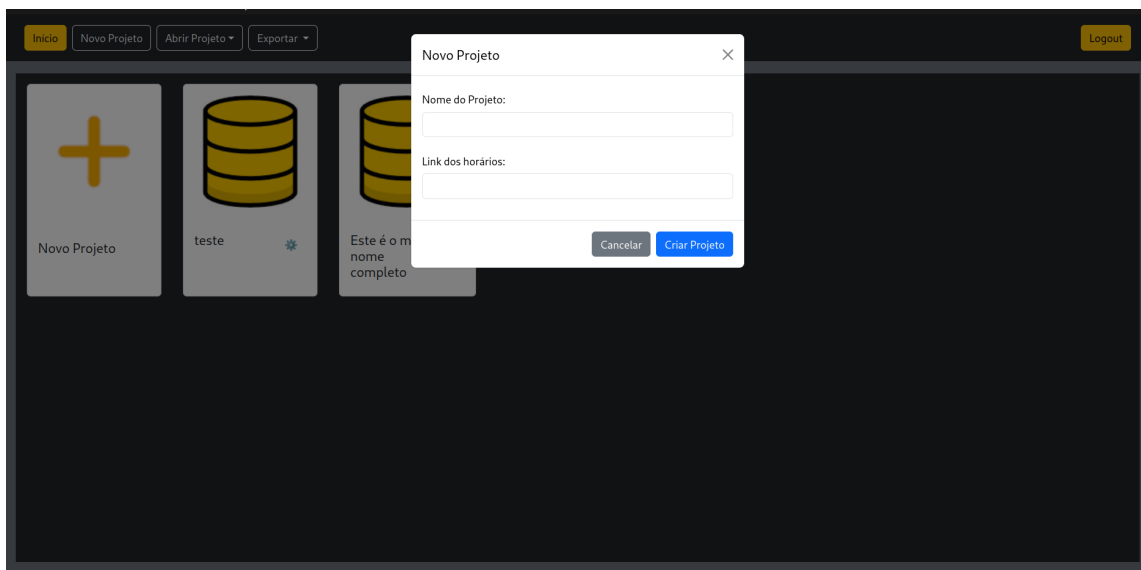


Figura 7: Novo Projeto *pop-up*

Após confirmar, o processo de *parsing* começa, o que se pode confirmar pelo novo projeto mostrado na página principal, com um símbolo de *loading*. Este processo demora vários minutos.

Por *default*, este projeto pode ser acedido apenas pelo criador e pelos administradores.

3.4.3 Controlo de Acesso

Cada projeto tem um criador, grupos a que pertence e pessoas convidadas. Portanto para controlar quem pode ler e escrever nos projetos, pode-se adicionar grupos ao projeto, ou pode-se adicionar diretamente utilizadores ao projeto.

Para adicionar utilizadores ou grupos ao projeto, pode-se clicar nas definições do projeto e adicionar aqueles que se querem. (mostrar definições)

Para gerir os grupos, o utilizador clica no botão "grupos", nesta página o utilizador pode criar grupos ou gerir grupos no qual ele pertence, o utilizador só pode gerir um grupo de cada vez, e ele pode convidar ou remover utilizadores do grupo (incluindo si mesmo).(pagina dos grupos)

3.4.4 Edição de Horários

Para editar um horário, um utilizador clica num dos projetos para o abrir redirecionando-o para uma página composta dos seguintes elementos:

1. Barra de Controlo Expansível
2. Horário
3. Barra Lateral Expansível

Abaixo está descrito cada um.

1. Barra de Controlo Expansível

Esta barra, visível na figura 8, localizada no topo do horário, expansível ao clicar no botão debaixo da *header*, permite selecionar o horário a ser mostrado. Tem vários *dropdowns* que o utilizador pode usar para escolher o curso, o ano, a turma, o turno e a semana do horário desejado. Aqui também se encontra um botão que mostra a distribuição de aulas por turno, distinguidas entre aulas Práticas (P) ou Teórico-Práticas (TP).

Junto ao botão de expansão da barra de controlo também se encontra o botão de expansão da barra lateral.

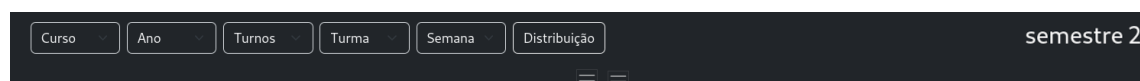


Figura 8: Barra de Controlo Expansível

2. Horário

O horário apresentado na figura 9, contém informações sobre as aulas de cada turma de um dado curso para um dado ano. Quando a página é carregada pela primeira vez, o horário visível é referente ao primeiro curso e primeiro ano listados nos respetivos botões, não exigindo, desta forma, a imediata seleção de qualquer opção. Analisando o cabeçalho da tabela, denota-se que para cada dia da semana, são apresentadas as turmas existentes do curso e ano selecionados. Inicialmente, o utilizador observa todos os turnos a que estas turmas pertencem, podendo escolher, através do botão turno, qual informação pretende visualizar. Relativamente ao conteúdo da tabela, para cada

uma das aulas presentes, é visível a informação relativa à unidade curricular a que pertence, bem como os docentes que a lecionam e a sala associada. Cada aula encontra-se sombreada com uma cor, específica à Unidade Curricular à qual corresponde, sendo as aulas teóricas sombreadas com um tom mais leve que as práticas da mesma Unidade Curricular. Estas aulas estão contidas em células que podem ser trocadas entre si. Caso o objetivo do utilizador seja mudar o conteúdo de uma célula, é possível selecioná-la e editá-la diretamente a partir da barra lateral expansível. Tendo em conta que nem todos os docentes e salas têm disponibilidade em qualquer horário, ao clicar num destes, ficam visíveis os respetivos blocos vermelhos, isto é, células sombreadas a vermelho que correspondem a períodos de indisponibilidade. Paralelamente, pode ser visualizada uma segunda tabela na parte inferior do ecrã, que permite a análise do número de aulas de uma unidade curricular em cada dia da semana por tipo de sala (salas de computadores ou práticas) e turno.

Início

Novo Projeto

Abrir Projeto ▾

Exportar ▾

Logout

HORÁRIO	SEGUNDA					TERÇA					QUARTA					QUINTA						
TURMA	3LEIC01	3LEIC02	3LEIC03	3LEIC04	3LEIC05	3LEIC01	3LEIC02	3LEIC03	3LEIC04	3LEIC05	3LEIC01	3LEIC02	3LEIC03	3LEIC04	3LEIC05	3LEIC01	3LEIC02	3LEIC03	3LEIC04	3LEIC05	3LEIC01	3
8:00																						
8:30																						
9:00			IA LPR			CG AVC	CG TOCM	CPD DCC - PMAADO S04	C TDRC	IA HLC			C PNFRCD			CPD VGRMR	C PMGP	IA RMNUG	CG DPMTM	C LGRC		
9:30			HLC B015			B314	B107	B313	B307				B015			B204	B305	B111	B303	B313		
10:00																						
10:30								CPD					C	IA	CG	CPD						PI BMCL B339
11:00	IA ICD	CPD DCC - SC5	C JBRpo	IA TFB	CG RPR			PFS JGR			C SMEL	IA LPR	CG DPMTM	CPD AIAAC	CPD CB			CG AAS				
11:30	B111	B202	B308	B204	B103			CB B015			B206	B305	B110	B302	B205			B014				
12:00																						
12:30																						
13:00																						
13:30																						
14:00																						
TIPO	UC	SEGUNDA	TERÇA	QUARTA	QUINTA	SEXTA	SOMA	SEGUNDA	TERÇA	QUARTA	QUINTA	SEXTA	SOMA									
PCs	C	1	1	1	2	0	5	1	1	2	2	0	6									
PCs	CG	1	2	1	1	0	5	2	2	1	1	0	6									
PCs	CPD	1	1	2	1	0	5	1	1	2	2	0	6									
PCs	IA	2	1	1	1	0	5	2	2	1	1	0	6									

Figura 9: Página principal

3. Barra Lateral Expansível

A barra lateral expansível, visível na figura 10, que pode ser expandida ao clicar no botão referido no ponto 1 ou com o atalho "CTRL + e", permite ao utilizador alterar todos os aspetos de uma aula e ver os conflitos criados.

A barra está dividida em 2 secções: a secção de edição e a secção dos conflitos.

A secção dos conflitos mostra uma lista com todos os conflitos encontrados no projeto. Esta lista é atualizada sempre que se faz uma alteração. Cada conflito tem uma breve descrição do conflito encontrado.

Exemplos:

- "Salas B301, B204 estão ocupadas"
- "Docentes DCS estão ocupados"

A secção de edição é onde um utilizador pode alterar todos os aspetos de uma aula. Inicialmente esta secção está escondida e, assim que uma aula seja selecionada no horário, as opções são mostradas com as opções atuais da aula.

Para mudar a Unidade Curricular, a hora inicial, hora final ou o dia da semana basta escolher a opção.

Para mudar os docentes, as salas ou as turmas basta escolher as opções desejadas de cada lista, ou pesquisar por elas.

Basta apenas submeter as mudanças para que estas fiquem em efeito e para ver os conflitos gerados.

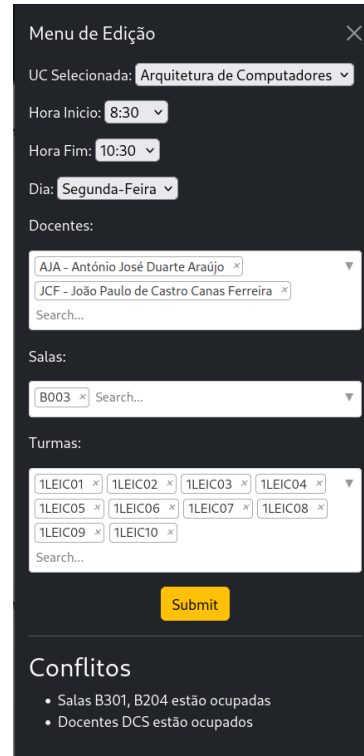


Figura 10: Barra Lateral Expansível

3.4.5 Exportação de Mudanças em Linguagem Natural

Quando o utilizador se encontrar satisfeito com as alterações efetuadas na plataforma, deve clicar em "Exportar", onde lhe serão apresentadas as alterações que efetuou desde o início da edição, num formato perceptível e intuitivo.

Ex: Aula CP01 (Sexta - 10:30 [Turma : turmaCTCPT12]) : Sala B214 ->Sala B331

3.5 Validação

Para validação do trabalho desenvolvido, foi analisado o desempenho da *web app* com a utilização de diferentes cursos, dado que assim conseguiríamos uma variância nos dados relativos a unidades curriculares, docentes, turmas e horários. Como critérios de avaliação do desempenho, foram tidos em conta alguns fatores como o tempo na etapa de *parsing* bem como no carregamento da informação da base de dados para uma página, dados esses que apresentamos nas tabelas abaixo. Além dos referidos, um outro fator a que atribuímos extrema importância, foi a validade dos dados obtidos no final das duas etapas referidas anteriormente. De forma a obter os melhores resultados possíveis com base nestes critérios, esta avaliação foi sendo realizada de forma regular e em paralelo com o desenvolvimento de cada uma das componentes do projeto. De cada vez que obtínhamos tempos que não eram satisfatórios à utilização da aplicação, como por exemplo uma página demorar demasiado tempo a carregar, ou dados que não estavam em concordância com os fornecidos no HTML de apoio, a prioridade tornava-se resolver estes problemas de forma a melhorar a experiência do utilizador final e garantir a menor probabilidade de ocorrência de erros. Além do referido, também era obtido *feedback* proveniente das reuniões semanais com o tutor Daniel Castro Silva, o que permitia avaliar o direcionamento do desenvolvimento do projeto.

Etapa de <i>Parsing</i>	Tempo (segundos)
Docentes	148.79
Turmas	554.55
Salas	67.47
UCs	211.91
Turnos	4.01
Total	986.73

Tabela 1: Tempo de *parsing* para as diversas etapas

Nota: Os tempos de *parsing* referidos na tabela 1 são o resultado da média de 10 medições isoladas e foram medidos num ambiente específico, podendo variar dependendo das condições de execução.

Turma do L.EIC	Tempo de Carregamento (segundos)
1	8.667
2	3.572
3	1.538

Tabela 2: Tempo de carregamento para o curso L.EIC

Nota: Os tempos de carregamento referidos na tabela 2 são o resultado da média de 10 medições isoladas e foram medidos num ambiente específico, podendo variar dependendo das condições de execução.

4 Conclusões

Esta secção sumariza os resultados alcançados, apresenta as lições aprendidas em retrospectiva e dá ideias de possíveis melhorias futuras.

4.1 Resultados alcançados

Em relação aos objetivos, a aplicação:

- Efetua parsing dos horários fornecidos.
- Sustém uma base de dados funcional.
- Possui um sistema de autenticação.
- Apresenta uma interface *user friendly*.
- Certifica o controlo de acesso nos projetos.
- Apresenta uma página de administração.

4.2 Contribuições

Tabela 3: Contribuições

Aluno	Planeamento	Parsing	Base de Dados	Autenticação	Interface	Projetos Backend	Controlo de Acesso	Total
Bárbara	1/4	1/4	1/3	0	4/9	0	0	1/4
Lucas	1/4	1/4	1/9	0	4/9	0	2/5	1/4
Luís	1/4	1/4	1/3	0	0	1	0	1/4
José	1/4	1/4	2/9	1	1/9	0	3/5	1/4

A tabela 3 mostra a distribuição de trabalho de cada membro, por cada componente do projeto.

4.3 Lições aprendidas

Durante o desenvolvimento do projeto, adquirimos conhecimento, aprendemos a usar tecnologias que nunca tínhamos usado e aprendemos várias lições.

Uma dessas lições foi a importância de longas considerações sobre a arquitetura das soluções a implementar e sobre a escolha das ferramentas a usar. Até agora, em outros projetos, nunca tivemos de escolher a *Framework* e as ferramentas a utilizar. O nosso tutor aconselhou-nos a usar algum tempo, no início, para analisar bem as nossas alternativas, compilar vantagens e desvantagens, pensar como é que nos podem beneficiar, como se integram com o restante projeto e como podem ser mantidas no futuro.

Uma outra lição aprendida, foi a importância de *feedback* constante ao longo do desenvolvimento. Todas as semanas mostramos ao professor tutor o trabalho desenvolvido e este deu-nos *feedback*. O facto que tivemos o *feedback* sempre cedo, permitiu-nos fazer melhorias de forma mais fácil, com mínima reescrita de código e sem gastar muito tempo.

Finalmente, o desenvolvimento do projeto com várias componentes essenciais, mostrou-nos a importância do planeamento do tempo atribuído a cada. Deu-nos a capacidade de perceber quando foi necessário dedicar mais ou menos tempo a uma dada componente e tornou-nos capazes de adaptar os nossos planos conforme os obstáculos encontrados.

4.4 Trabalho futuro

Algumas *features* discutidas que não foram implementadas, quer por necessidade de foco do projeto, quer por estarem fora dos objetivos pretendidos, quer por serem *nice to have's*:

- **Mais vistas possíveis:** foi necessário focar o projeto na vista por turnos, sendo esta a vista mais importante de acordo com os *stakeholders*. Faria sentido, no futuro implementar mais vistas, como uma vista semelhante às do site dos horários.
- **Funcionalidade Colaborativa:** não faz parte dos objetivos pretendidos para o projeto, mas é algo em que o professor tutor mostrou interesse em ter no futuro. Algo semelhante ao que se pode encontrar ao usar *Google Docs* [13], de forma a ter vários utilizadores a editarem um projeto em sintonia.
- **Capacidade *Multi-Threaded* do *Parsing*:** o fator limitante na velocidade do *parsing* é o número significativo de pedidos que necessitam de ser feitos. Neste momento estes pedidos são feitos sequencialmente e, como tal, ao reescrever o *parsing* para que possa fazer vários pedidos em paralelo, a sua *performance* pode ser melhorada.
- **Anotações em Conflitos:** dar ao utilizador a capacidade de escrever notas, associadas a um conflito, para que este possa lê-las mais tarde.
- **Ordenação Dinâmica de Docentes e Salas na Barra Lateral:** mudar a ordenação das opções na barra lateral dinamicamente, para que, por exemplo, docentes da UC selecionada apareçam antes do resto e salas do mesmo tipo da sala da aula selecionada apareçam antes do resto.
- **Filtros de Informação na Edição de Horários:** de forma a filtrar a informação referente aos docentes, salas, turmas ou UCs, diminuindo o espaço ocupado por cada aula na página.
- **Zoom Variável nos Horários:** opções de diferentes nível de zoom, que redimensionem as aulas, para ter uma visão mais alto-nível do horário.

Referências

- [1] @fat @mdo. *Bootstrap*. <https://getbootstrap.com/>. [Online; acessado 20 de junho de 2023]. 2010.
- [2] BeautifulSoup. *Beautiful Soup Documentation*. <https://beautiful-soup-4.readthedocs.io/en/latest/>. [Online; acessado 20 de junho de 2023]. 2023.
- [3] Django. *DjangoProject*. <https://www.djangoproject.com/>. [Online; acessado 20 de junho de 2023]. 2023.
- [4] Django. *How to deploy with ASGI*. <https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/>. [Online; acessado 20 de junho de 2023]. 2023.
- [5] Django. *How to use Django with Daphne*. <https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/daphne/>. [Online; acessado 20 de junho de 2023]. 2023.
- [6] Flutter. *Flutter*. <https://flutter.dev/>. [Online; acessado 20 de junho de 2023]. 2023.
- [7] Flutter. *FlutterForWeb*. <https://flutter.dev/multi-platform/web>. [Online; acessado 20 de junho de 2023]. 2023.
- [8] Flutter. *Sqlite*. <https://docs.flutter.dev/cookbook/persistence/sqlite>. [Online; acessado 20 de junho de 2023]. 2023.
- [9] OpenJS Foundation. *jQuery*. <https://jquery.com/>. [Online; acessado 20 de junho de 2023]. 2023.
- [10] GeeksForGeeks. *Python SQLite - Connecting to Database*. <https://www.geeksforgeeks.org/python-sqlite-connecting-to-database/>. [Online; acessado 20 de junho de 2023]. 2023.
- [11] GeeksForGeeks. *Python Web Scraping Tutorial*. <https://www.geeksforgeeks.org/python-web-scraping-tutorial/>. [Online; acessado 20 de junho de 2023]. 2023.
- [12] GeeksForGeeks. *WebScraping in Flutter*. <https://www.geeksforgeeks.org/web-scraping-in-flutter/>. [Online; acessado 20 de junho de 2023]. 2023.
- [13] Google. *Google Docs*. <https://docs.google.com/>. [Online; acessado 20 de junho de 2023]. 2023.
- [14] GitHub Inc. *Repositório do Projeto*. <https://github.com/13luiscabral13/Projeto-Integrador>. 2023.
- [15] Janking. *dropdown*. <https://github.com/Janking/dropdown>. [Online; acessado 20 de junho de 2023]. 2019.
- [16] Kivy. *Kivy*. <https://kivy.org/>. [Online; acessado 20 de junho de 2023]. 2023.
- [17] Laravel. *Laravel*. <https://laravel.com/>. [Online; acessado 20 de junho de 2023]. 2023.
- [18] Neves. “GPU-based timetable generation”. Tese de mestrado. R. Dr. Roberto Frias, 4200-465 Porto: Faculdade de Engenharia da Universidade do Porto, 2016, p. 120.
- [19] PyPI. *beautifulsoup4*. <https://pypi.org/project/beautifulsoup4/>. [Online; acessado 20 de junho de 2023]. 2023.
- [20] PyPI. *daphne*. <https://pypi.org/project/daphne/>. [Online; acessado 20 de junho de 2023]. 2023.

- [21] PyPI. *Pip*. <https://pypi.org/project/pip/>. [Online; acedido 20 de junho de 2023]. 2023.
- [22] PyPI. *Pipenv*. <https://pypi.org/project/pipenv/>. [Online; acedido 20 de junho de 2023]. 2023.
- [23] PyPI. *readerwriterlock*. <https://pypi.org/project/readerwriterlock/>. [Online; acedido 20 de junho de 2023]. 2023.
- [24] PyPI. *requests*. <https://pypi.org/project/requests/>. [Online; acedido 20 de junho de 2023]. 2023.
- [25] PyPI. *unicode*. <https://pypi.org/project/unicode/>. [Online; acedido 20 de junho de 2023]. 2023.
- [26] PyPi. *networkx*. <https://pypi.org/project/networkx/>. [Online; acedido 27 de junho de 2023]. 2023.
- [27] Pypi. *PyQT5*. <https://pypi.org/project/PyQt5/>. [Online; acedido 20 de junho de 2023]. 2023.
- [28] PythonBasics. *Tkinter*. <https://pythonbasics.org/tkinter/>. [Online; acedido 20 de junho de 2023]. 2023.
- [29] RubyOnRails. *RubyOnRails*. <https://rubyonrails.org/>. [Online; acedido 20 de junho de 2023]. 2023.
- [30] jQuery Script.net. *Searchable Multi-select Dropdown*. <https://www.jqueryscript.net/form/Searchable-Multi-select-jQuery-Dropdown.html>. [Online; acedido 20 de junho de 2023]. 2023.
- [31] Silva. “Sistema multi-agente para geração de horários no ensino secundário”. Tese de mestrado. R. Dr. Roberto Frias, 4200-465 Porto: Faculdade de Engenharia da Universidade do Porto, 2000, p. 133.