Agile wing

Relatório

A close up of a logo

Description automatically generated

Outubro de 2023

TPSIP\_09.22

Inês Paiva, Rui Ruão, Sara Martins, Vasco Vitória

Índice

[1. Introdução 3](#_Toc147913428)

[1.1. Enquadramento 3](#_Toc147913429)

[1.2. Objetivos – Requisitos funcionais/ não funcionais 4](#_Toc147913430)

[1.2.1 Requisitos Funcionais: 5](#_Toc147913431)

[1.2.2. Requisitos Não Funcionais: 7](#_Toc147913432)

[1.3. Estrutura e planificação 8](#_Toc147913433)

[1.3.1 Gráfico de GANT 8](#_Toc147913434)

[1.4. Organização do relatório 8](#_Toc147913435)

[2. Arquitetura de Software 10](#_Toc147913436)

[2.1 Diagrama de casos de uso 10](#_Toc147913437)

[2.2. Diagramas de sequência – principais 11](#_Toc147913438)

[2.2.1 Gestão de horários 11](#_Toc147913439)

[2.2.2. Preenchimento de disponibilidade 12](#_Toc147913440)

[2.3. BPMN v2.0 - Modelação dos principais processos de negócio 13](#_Toc147913441)

[2.3.1. BPMN Disponibilidades 13](#_Toc147913442)

[2.3.2. BPMN Planeamento 13](#_Toc147913443)

[2.4. Modelo de dados 14](#_Toc147913444)

[2.5. Diagrama de classes 15](#_Toc147913445)

[2.6. User Interface (UI) Flow Diagram 16](#_Toc147913446)

[2.7. Dicionário de dados 17](#_Toc147913447)

[3. Desenvolvimento 18](#_Toc147913448)

[3.1. Arquitetura da aplicação 18](#_Toc147913449)

[3.2. Tecnologias 19](#_Toc147913450)

[3.3. Ferramentas de desenvolvimento 22](#_Toc147913451)

[3.3. Design de Interfaces 24](#_Toc147913452)

[3.4. Processo do Desenvolvimento 25](#_Toc147913453)

[3.4.1. Arquitetura e Prototipagem 25](#_Toc147913454)

[3.4.2. Environment Setup 26](#_Toc147913455)

[3.4.3. Sprint I 27](#_Toc147913456)

[3.4.4. Sprint II 28](#_Toc147913457)

[3.4.5. Sprint III 29](#_Toc147913458)

[3.4.6. Sprint IV 31](#_Toc147913459)

[Anexos 32](#_Toc147913460)

1. Introdução

Este relatório e o trabalho associado estão enquadrados no âmbito do módulo 5425 - Projeto de Tecnologia e Planeamento de Sistemas de Informação, como parte do curso de Técnico Especialista em Tecnologias e Programação de Sistemas de Informação.

O propósito deste projeto é criar uma aplicação que atenda a requisitos específicos definidos pelo cliente. O cliente em questão é a Academia de Formação ATEC que, através da sua representante, Carmen Teixeira, estipula requisitos para responder às suas necessidades de gestão.

O objetivo principal deste projeto é abordar as necessidades da empresa cliente desenvolvendo a referida ferramenta. Para alcançar este objetivo, foram criadas equipas compostas por alunos do curso, com a nomeação de *product owners* em cada grupo, selecionados pelos formadores do curso.

Os elementos constituintes deste grupo são: Inês Paiva (*Developer Back/Frontend*), Rui Ruão (*Developer Back/Frontend & TeamLeader*), Sara Martins (*Developer Back/Frontend & TeamLeader*), Vasco Vitória (*Developer Back/Frontend & Product Owner*).

**ADICIONAR CARMEN E ATEC resp e cliente**

## Enquadramento

O objetivo central deste projeto é desenvolver um sistema abrangente de gestão de disponibilidades e atribuições dos/aos formadores da ATEC, que seja eficaz na coordenação das suas atividades. O sistema contará com um componente de *backend e frontend*. O *backend* cuidará do armazenamento de dados dos formadores, dos horários e das atividades, enquanto o *frontend* proporcionará uma interface intuitiva para os formadores atualizarem as suas disponibilidades e para os administradores programarem, atribuírem e alterarem formadores a ufcds que integram uma turma de forma eficiente. O sistema visa otimizar a alocação dos formadores, melhorando a comunicação e a organização das atividades educacionais da ATEC.

## 1.2. Objetivos – Requisitos funcionais/ não funcionais

Para assegurar que todos os requisitos do projeto fossem respondidos, a nossa equipa adotou uma abordagem abrangente. Começamos por criar um questionário que se alinhava com o tema comunicado. Em seguida, realizamos uma reunião com Carmen Teixeira, representante da ATEC (cliente) e responsável pelo planeamento e gestão dos horários dos formadores. Durante essa reunião, não só obtivemos respostas detalhadas às nossas perguntas, mas também recebemos um enunciado que delineava as metas do projeto com mais algum detalhe.

Para além deste manancial de informação mais institucional abordamos alguns professores em conversas não-formais, aos quais mostramos o que já tínhamos desenvolvido pedindo opinião e feedback. Destacando a professora La-Salete Moreira, o professor Ruben Canelas, o professor Sérgio Nogueira e o professor Nelson Lima.

Com base nestes insights, iniciamos o processo de conceção da arquitetura de software, mantendo um canal de comunicação ativo com a cliente para esclarecer quaisquer dúvidas que surgissem. À medida que avançávamos no processo, desenvolvemos um protótipo no Figma, uma ferramenta de *user interface design/layout*, para validação junto da cliente. Essa colaboração estreita com a cliente e o uso de ferramentas visuais como o protótipo foram fundamentais para identificar e documentar os requisitos essenciais do projeto.

Com base nestes recursos e processo, bem como nas anotações das reuniões com a cliente, foram identificados os seguintes requisitos:

### 1.2.1 Requisitos Funcionais:

**Tipos de Utilizador - Formador:**

- 1º Acesso/Reposição palavra-passe, formador acede à sua área reservada através do link enviado por email e pode fazer login após a conclusão da definição da palavra-passe (este registo e consecutivo email é criado pelo planeamento).

- O formador realiza login com utilizador e palavra-passe.

-Um formador tem duas cores associadas para o distinguir nos mapas de planeamento, cores definidas pelo planeamento.

- Após login o formador pode preencher a sua disponibilidade (dentro dos blocos de horário definidos pela instituição), consultar as suas atribuições (horários) e alterar a sua palavra-passe.

- Notas informativas (por exemplo, disponibilidade total, mas apenas 2 dias por semana para marcações pela ATEC) podem ser adicionadas.

- O formador pode definir o tipo de disponibilidade: Online, Presencial, Disponível, Indisponível.

- O formador pode bloquear e submeter a sua disponibilidade (aparece no planeamento somente quando bloqueado).

- O formador pode alterar a palavra-passe.

- Pedidos de Alterações de disponibilidade bloqueada enviam notificação para o planeamento. (***extra***)

- Consulta de horários integrados com calendários do Outlook e Gmail. (***extra***)

- Integração google/Microsoft agenda. (***extra***)

**Tipos de Utilizador - Planeamento:**

- O planeamento realiza login com utilizador e palavra-passe.

- Possibilidade de criar, editar e apagar formadores no sistema, incluindo reset de palavra-passe a pedido do professor.

- Possibilidade de criar, editar e apagar turmas no sistema para atribuição aos horários dos formadores.

- Visualização geral por turma os blocos UFCD acrescentar nomes de formadores.

- Envio de e-mail para formadores com notificações de aprovação/alocação ou alterações de horário.

- Mudança do tipo de disponibilidade após alocação final, por defeito é presencial, salvo definido pelo formador.

- Inserção/edição de dados via folha Excel. (**Importação extra**)

- Controlo de dados: visualização e edição de todos os perfis de formadores, datas de última gravação e acesso.

- Preparação para detetar a turma alocada e associá-la aos agendamentos (turma, UFCD, professor).

- Inserção de nome e e-mail de novos formadores aciona envio de link de registo.

**Sistema:**

- Após criação em sistema este envia um link seguro por e-mail para o formador para realizar o primeiro login.

- O sistema permite a seleção de horários dentro do intervalo de disponibilidade do formador.

- Nas atribuições para uma dada turma, dia e bloco, apenas mostrar professores com disponibilidade, que possam lecionar as UFCDs daquele curso. Na listagem de UFCDs filtrar a partir desta lista de professores, para aquela seleção em concreto.

- Possibilitar/facilitar um sistema de troca de horários entre formadores do lado do planeamento.

- Registo da data de acesso do formador ao sistema no login.

- Registar a data do último bloquear de disponibilidade do formador.

- Calendário gerado com blocos de horas e dias.

- Capacidade de importar e exportar dados em formato Excel. (**Importar extra**)

- Filtros para visualização por turma nas atribuições.

**Requisitos de Cálculos:**

**- Agendamento e verificação de disponibilidades.**

**- Trocas de horários.**

**- Conceção de cronogramas/mapas de disponibilidade.**

### 1.2.2. Requisitos Não Funcionais:

**Design:**

- Design minimalista e curvilíneo recorrendo a uma palete em tons de azul derivados da imagem institucional da ATEC.

- Interface responsivo a múltiplas resoluções de ecrã.

## 1.3. Estrutura e planificação

Tendo já 3 elementos do nosso grupo experienciado a metodologia *Scrum* com *Kanban boards* (*Srumban*) optamos por aproveitar essa experiência anterior na gestão e acompanhamento das tarefas do projeto. A escolha assentou não apenas na eficácia demonstrada, mas também no facto de ser uma abordagem *Agil* que nos proporciona a flexibilidade necessária de adaptação rápida e redução dos tempos de resposta

Para a organização das nossas *boards*, iremos recorrer ao *Trello*, uma ferramenta conhecida pela sua eficiência e simplicidade na gestão de tarefas e colaboração em equipa. Além disso, para uma gestão mais detalhada do *roadmap* e das dependências entre tarefas, utilizaremos a ferramenta integrada *Team* *Gantt*, que nos permitirá uma visão clara e estruturada do progresso do projeto.

### 1.3.1 Gráfico de GANT

Optámos por utilizar o site *Team Gantt* para a elaboração do nosso gráfico de *Gantt*, pois esta plataforma permite a criação de dependências entre tarefas e integra-se de forma eficaz com o *Trello*. Esta escolha estratégica proporciona à nossa equipa uma visão clara e organizada do cronograma do projeto, permitindo-nos estabelecer relações de dependência entre as várias etapas do trabalho. Além disso, a integração com o *Trello* facilita a sincronização das informações entre as duas ferramentas, tornando a gestão do projeto mais eficiente e facilitando a identificação de eventuais desvios no plano. Vemos esta abordagem como fundamental para garantir que o projeto avance de acordo com as expectativas e que as tarefas sejam concluídas de forma coordenada e no prazo estabelecido.

[Link para gráfico Gant Agile Wing](https://prod.teamgantt.com/gantt/schedule/?ids=3632617&public_keys=KuKPnztWmsxG&zoom=d100&font_size=12&estimated_hours=0&assigned_resources=0&percent_complete=0&documents=0&comments=0&col_width=355&hide_header_tabs=0&menu_view=1&resource_filter=1&name_in_bar=0&name_next_to_bar=0&resource_names=1&resource_hours=1#user=&company=&custom=&date_filter=&hide_completed=&color_filter=)

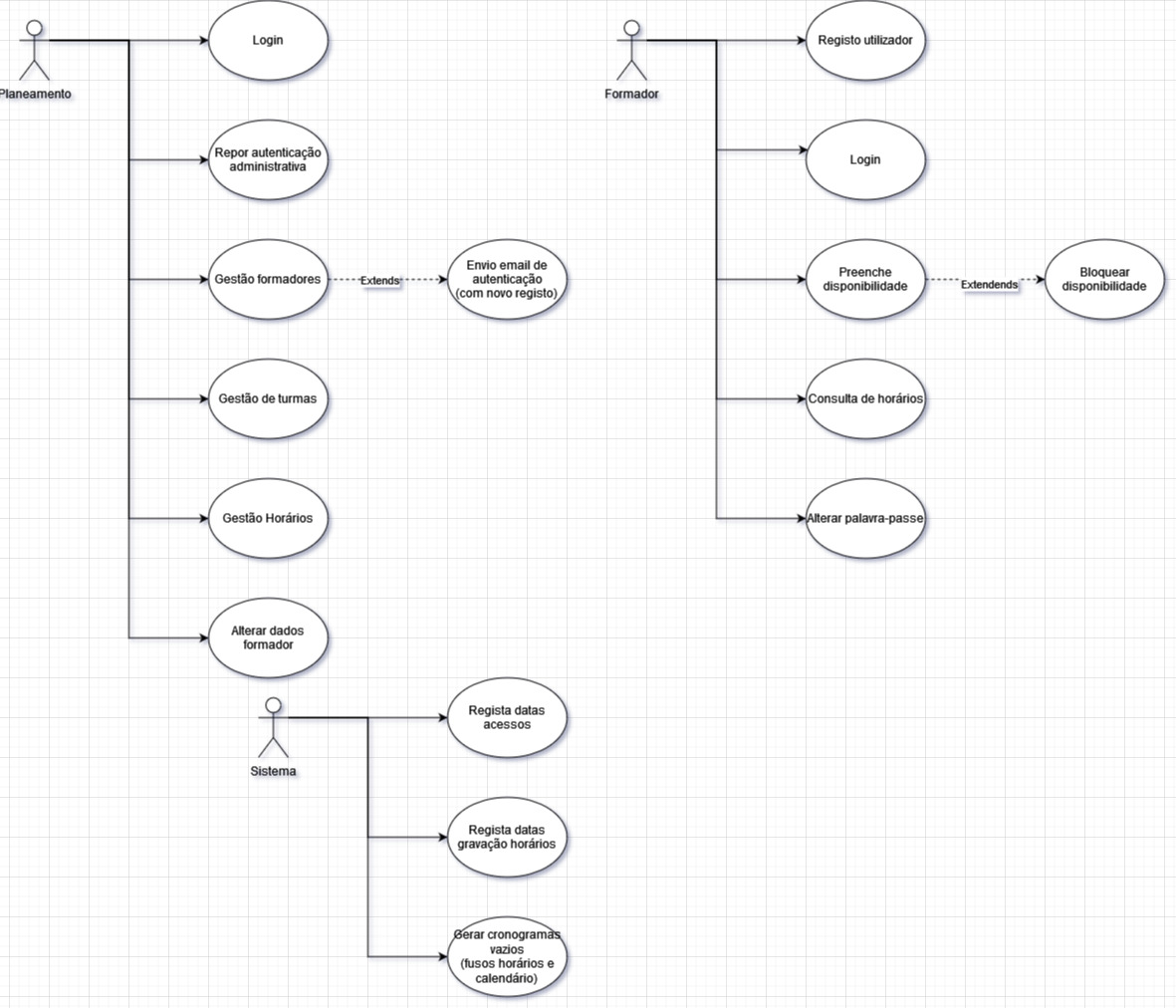
## 1.4. Organização do relatório

**A PREENCHER NO FINAL, QUANDO RELATÓRIO ESTIVER COMPOSTO E COM ORGANIZAÇÃO FINAL**

1. Arquitetura de Software

## 2.1 Diagrama de casos de uso

Um diagrama de caso de uso é a representação gráfica que descreve a interação entre atores (entidades externas, como utilizadores ou sistemas) e um software/aplicação. É utilizado na engenharia de software para modelar e documentar os requisitos funcionais do sistema, com foco nas ações que o sistema deve realizar em resposta às interações com os atores.



caso de uso

Serve o presente diagrama para ajudar a equipa a interpretar, através de uma visão macro, as necessidades a serem respondidas.

## 2.2. Diagramas de sequência – principais

### 2.2.1 Gestão de horários

Um diagrama de sequência é mais uma ferramenta de modelagem utilizada para representar a interação entre os intervenientes e o sistema. Representam o comportamento dinâmico de um sistema e são úteis para perceber a lógica do processo e fluxo de ações.

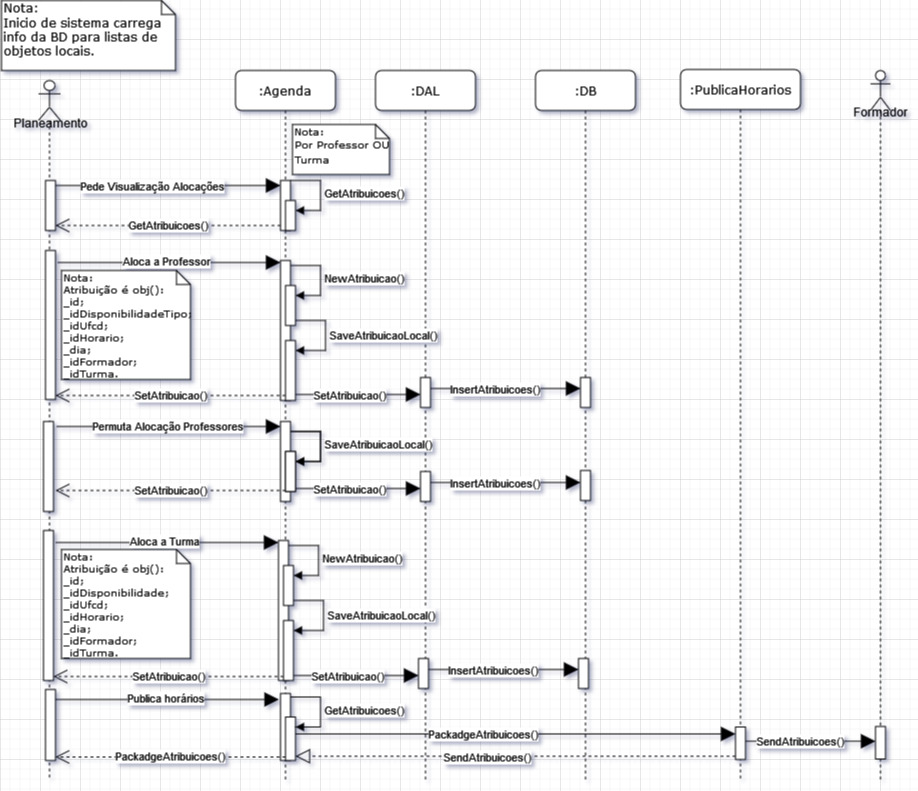


diagrama de sequência – gestão de horários

Através desta representação, apresentamos as ações e fluxos de informação por parte do utilizador que faz a gestão dos formadores, turmas e respetivas alocações.

### 2.2.2. Preenchimento de disponibilidade

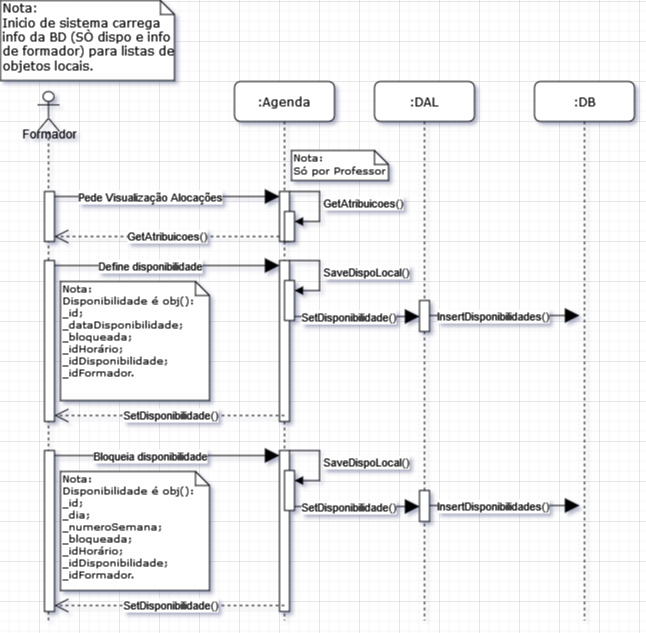


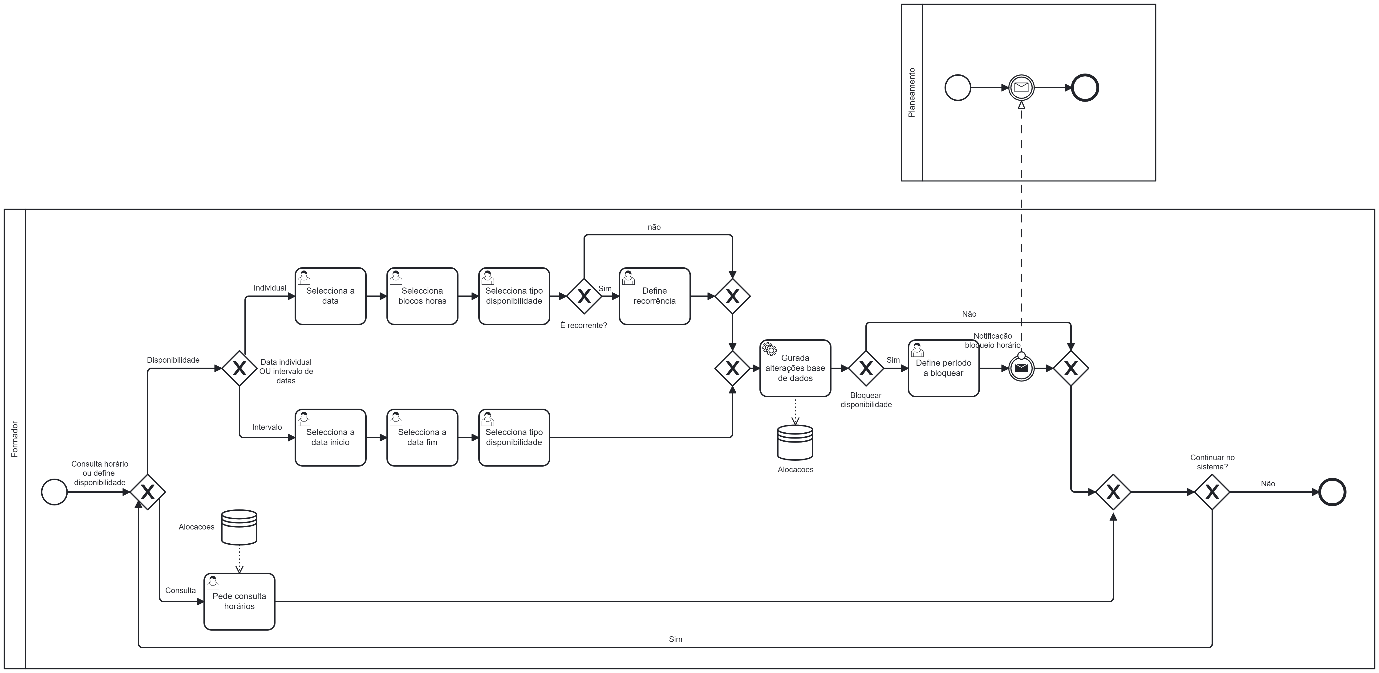
diagrama de sequência – preenchimento de disponibilidades

Através desta representação, apresentamos as ações e fluxos de informação por parte do formador que interage com aplicação para facultar as suas disponibilidades.

## 2.3. BPMN v2.0 - Modelação dos principais processos de negócio

### 2.3.1. BPMN Disponibilidades

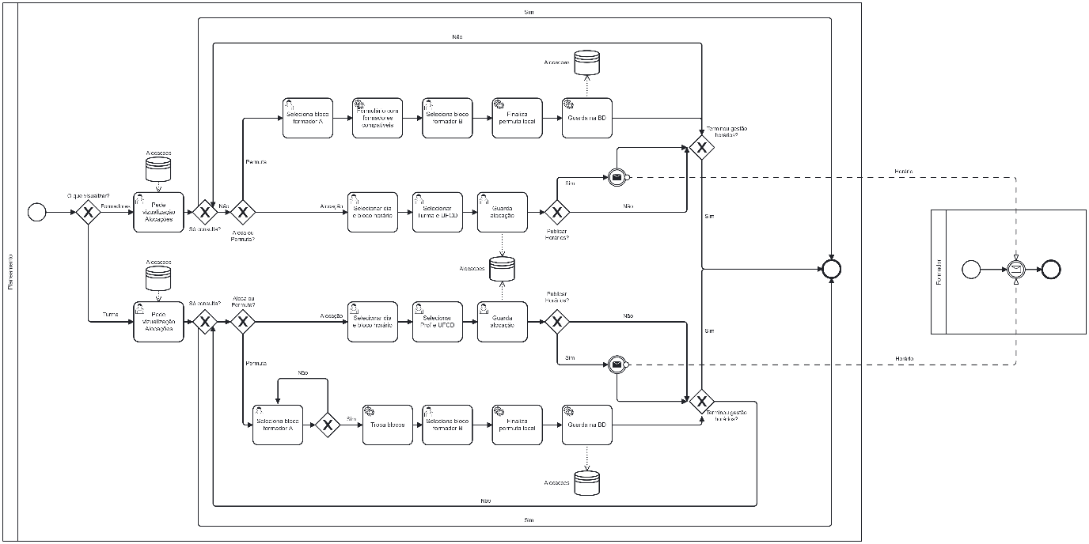
Trata-se de um modelo que representa os processos de negócio, permitindo a demonstração de forma detalhada de como um processo flui, incluindo as respetivas etapas, tarefas, decisões e fluxos de informação. Ajuda a evitar ambiguidades e garante que todas as pessoas envolvidas tenham uma compreensão comum dos processos.



Disponibilidades

A presente imagem documenta, através de uma visão macro, os processos/necessidades a que a aplicação tem de responder por parte do formador.

### 2.3.2. BPMN Planeamento

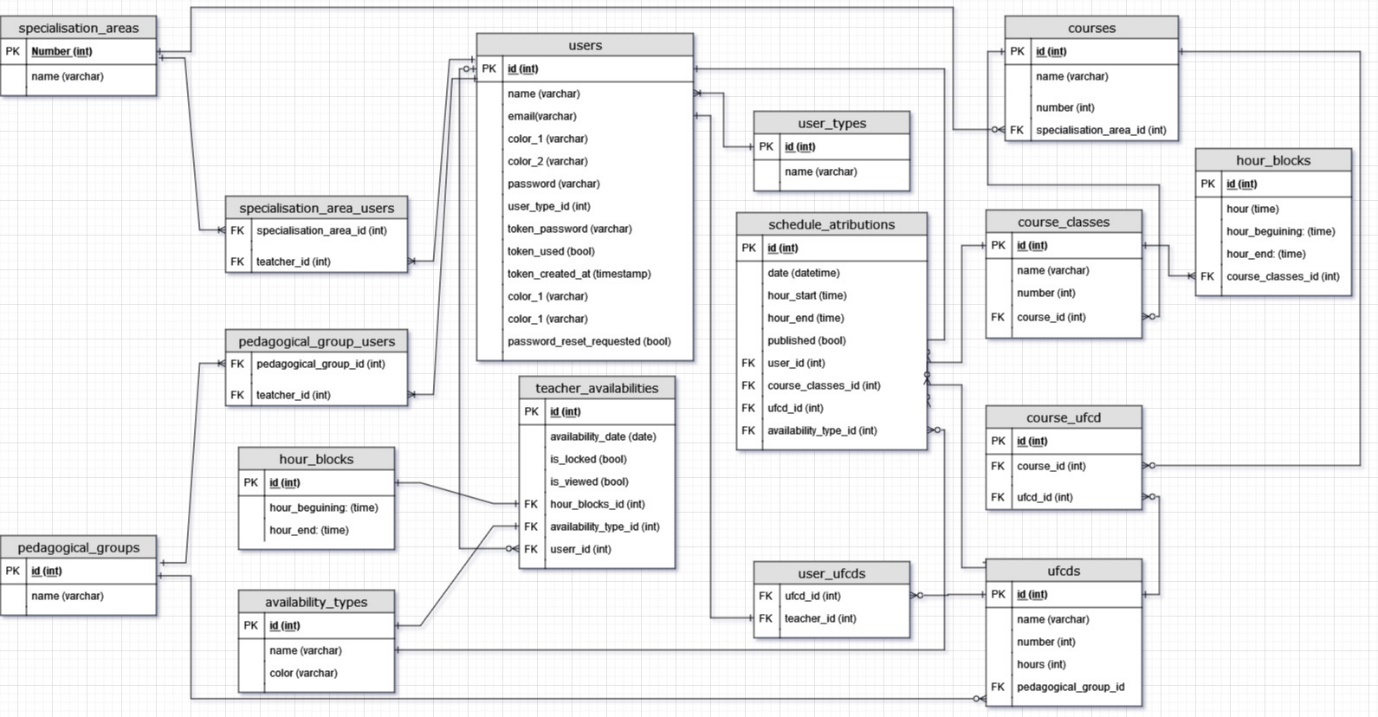


planeamento

Também através do mesmo modelo, desta vez, representamos os processos por parte de quem faz o planeamento de horários, turmas e formadores.

## 2.4. Modelo de dados

É a representação estruturada e abstrata de comos os dados são organizados e relacionados dentro do sistema. É um diagrama fundamental para a criação da base de dados da aplicação.



## 2.5. Diagrama de classes

O presente diagrama é a representação visual que descreve a estrutura de um sistema orientado a objetos. É utilizado para ilustrar as classes, os seus atributos, métodos e relacionamentos, demonstrando ser mais uma das ferramentas fundamentais na implementação de uma nova aplicação.

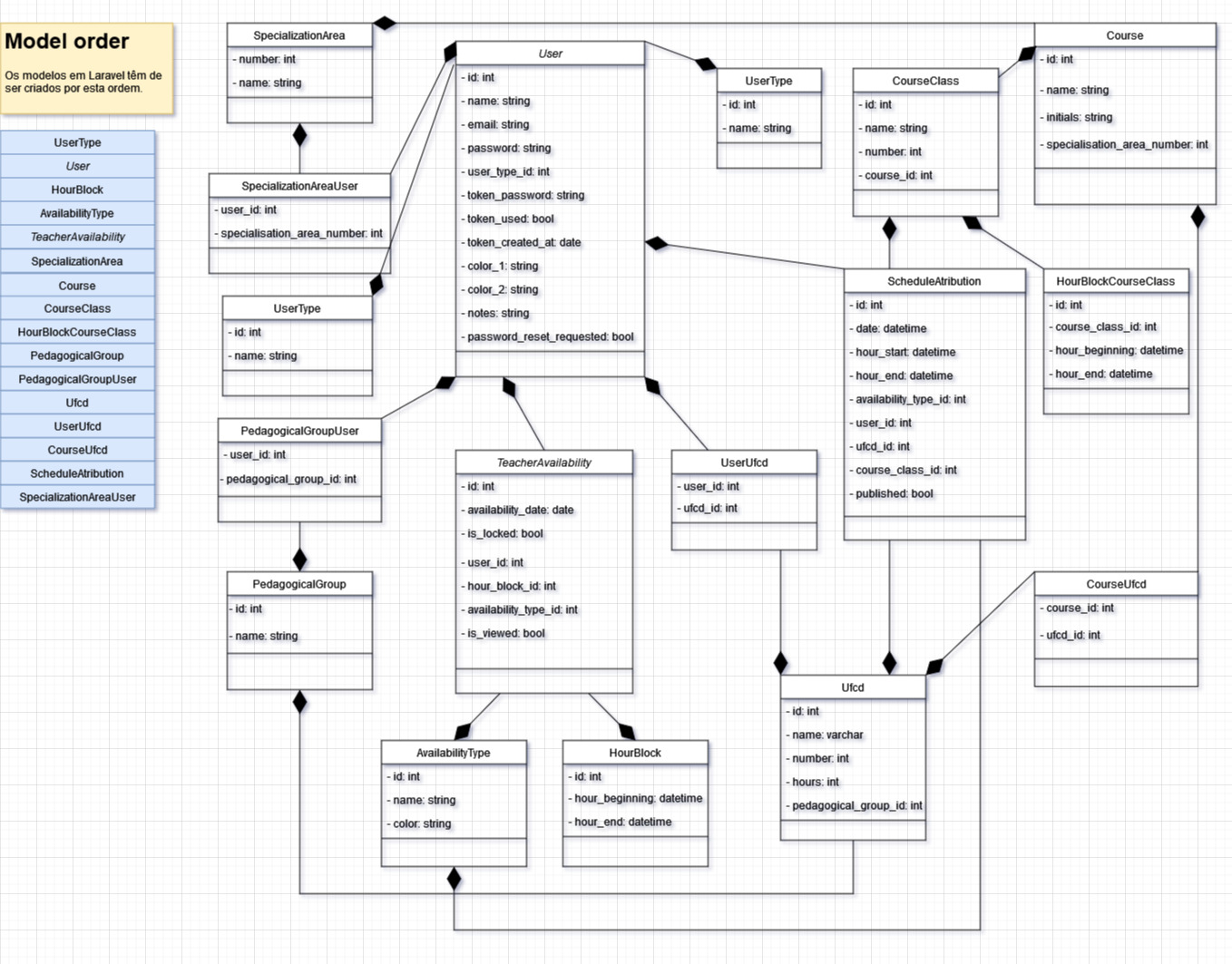


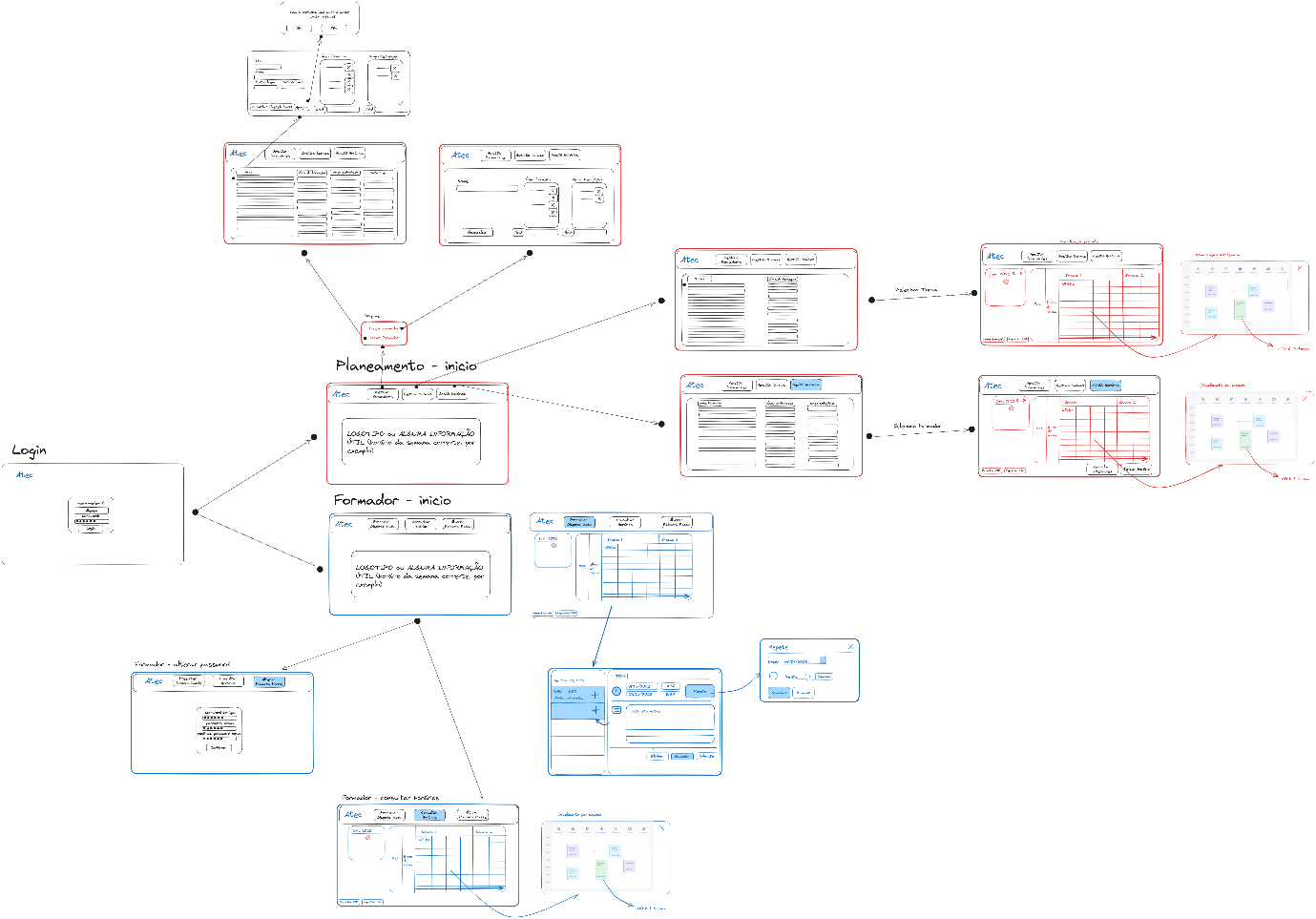
diagrama de classes

Representamos a estrutura estática do sistema, focada nas classes existentes, seus métodos e atributos e relações entre elas.

## 2.6. User Interface (UI) Flow Diagram

Numa primeira fase do projeto, optamos por utilizar o Excalidraw para criar um esboço rápido do layout da aplicação. Esta ferramenta proporcionou uma maneira ágil e intuitiva de esboçar as ideias iniciais de design, permitindo-nos visualizar a disposição geral dos elementos da interface. Este passo foi essencial para alinhar a nossa visão sobre o design/layout da aplicação antes de avançarmos para a fase de prototipagem.

Posteriormente, na fase de desenvolvimento do protótipo, escolhemos o Figma como a nossa ferramenta principal. O Figma possibilitou a criação de um protótipo mais detalhado e interativo, onde pudemos não apenas definir o layout, mas também simular a interação do utilizador com a aplicação. Isto incluiu a definição de transições, botões clicáveis e fluxos de navegação. Ao usar o Figma, conseguimos criar uma representação mais precisa e funcional da aplicação, o que foi fundamental para validarmos as nossas ideias com a equipa e o cliente, garantindo que o design respondesse aos requisitos e expectativas estabelecidos. [Link para protótipo FIGMA](https://www.figma.com/file/WKhB6OiF7OPIwbisUInLzv/Pr%C3%B3totipo-Agile-Wing?type=design&node-id=0%3A1&mode=design&t=OZ91ixCFSjWL2rNY-1)



UI

Permitiu desenvolver o primeiro esboço daquilo que viria a ser aplicação, representando também a navegação entre as suas páginas.

## 2.7. Dicionário de dados

**User Types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific user type | Numeric |  | Primary Key |
| Name | A name to indicate the type of user type | String |  |  |

Tabela 1 – User Types

A tabela 1 guarda os dados dos tipos de utilizadores. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará o *user type*;
* Nome;

**Users**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific user | Numeric |  | Primary Key |
| Name | The name of the user | String |  |  |
| Email | The email of the user | String |  | Is unique |
| Password | The password of the user | Number |  |  |
| Color\_1 | The primary color used to identify the user in schedules | RGB |  | Is nullable |
| Color\_2 | The secondary color used to identify the user in schedules | RGB |  | Is nullable |
| User\_type\_id | Number to identify the type of user | Numeric |  | Foreign id |
| Email\_verified\_at | Timestamp to indicate what time the user’s email was verified | Timestamp |  | Is nullable |
| Token\_password | A secondary one-time password | String |  | Is nullable |
| Token\_created\_at | Date that the token password was created | Date |  | Is nullable |
| Password\_reset\_requested | Indicates whether or not a user has requested a password reset before | Boolean |  | Default false |
| Notes | Notes about the user | Text |  | Is nullable |
| Last\_login | When the user last logged in | Date |  | Is nullable |

Tabela 2 – Users

A tabela 2 guarda os dados dos utilizadores. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará o *user*;
* Nome;
* Email;
* Palavra-passe;
* Color\_1 - a cor primaria usada para rapidamente identificar o formador em certas tabelas
* Color\_2 - a cor segundaria usada para rapidamente identificar o formador em certas tabelas
* User\_type\_id - os valores encontram-se na tabela roles e poderão ser:
  + Planeamento;
  + formador;
* Email\_verified\_at – data de quando o email do utilizador foi verificado;
* Token\_password - palavra-passe utilizada para o primeiro início de sessão ou para a reposição da palavra-passe;
* Token\_created\_at - um carimbo de data/hora utilizado para indicar um período de vida de 24 horas após a criação da palavra-passe simbólica;
* Password\_reset\_requested - variável interna ainda não utilizada para um futuro sistema interno de reposição da palavra-passe;
* Notes – caso necessário adicionar informações sobre o utilizador;
* Lat\_login – a data do ultimo login efetuado pelo utilizador;

**Password Resets**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Email | Email associated with password resets | String |  | Index |
| Token | Token of the password | String |  |  |

Tabela 3 – Password Resets

A tabela 3 guarda os dados dos projetos. A estrutura desta tabela é a seguinte:

* Email;
* Token – o valor de uma password temporária usada para o pedido de uma nova palavra-passe;

**Hour Blocks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific hour block | Numeric |  | Primary Key |
| Hour\_beginning | The starting time of the hour block | Time |  |  |
| Hour\_end | The ending time of the hour block | Time |  |  |

Tabela 4 – Hour Blocks

A tabela 4 guarda os dados dos blocos de hora. A estrutura desta tabela é a seguinte:

* ID - irá ser único e identificará o *bloco de hora*;
* Hour\_beginning – a hora de início de um bloco de hora;
* Hour\_end – a hora de fim de um bloco de hora;

**Availability Types**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific availability type | Numeric |  | Primary Key |
| Name | Name of the availability type | String |  |  |
| Color | Color of the ability type | String |  |  |

Tabela 5 – Availability Types

A tabela 5 guarda os dados dos tipos de disponibilidade. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará o *tipo de disponibilidade*;
* Nome;
* Color – uma cor usada para facilmente identificar o tipo de disponibilidade em certas tabelas;

**Teacher Availabilities**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific teacher availability | Numeric |  | Primary Key |
| Availability\_date | Date of the teacher availability | Date |  |  |
| Is\_locked | Indicates whether this teacher availability has been locked or not | Boolean |  | Default false |
| Is\_viewed | Indicates whether this teacher availability has been seen or not | Boolean |  | Default false |
| User\_id | Id of the user this teacher availability belongs to | Numeric |  | Foreign id |
| Hour\_block\_id | Id of the hour block that was selected | Numeric |  | Constrained |
| Availability\_type\_id | Id of the availability type that was selected | Numeric |  | Constrained |

Tabela 6 – Teacher Availabilities

A tabela 6 guarda os dados das disponibilidades dos formadores. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará as *disponibilidade*s *dos formadores*;
* Availability\_date – a data bloqueada pelo formador com uma certa disponibilidade;
* Is\_locked – indica se este bloco foi bloqueado ou não. Quando um formador publica uma disponibilidade, essa fica bloqueada no lado do formador;
* Is\_viewed – indica se um utilizador de planeamento viu aquela disponibilidade;
* User\_id – armazena o utilizador que forneceu esta disponibilidade;
* Hour\_block\_id – armazena o bloco de hora bloqueado pelo formador com uma certa disponibilidade;
* Availability\_type\_id – armazena o tipo de disponibilidade bloqueada pelo formador;

**Specialization Areas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific specialization area | Numeric |  | Primary Key |
| Number | The number of the specialization area | Integer |  | Is unique |
| Name | The name of the specialization area | String |  |  |

Tabela 7 – Specialization Areas

A tabela 7 guarda os dados das áreas de disponibilização. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará as *áreas de especialização*;
* Number – um numero único para representar aquela área de especialização;
* Nome;

**Courses**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific course | Numeric |  | Primary Key |
| Name | Name of the course | string |  |  |
| Initials | Initials of the course | String |  |  |
| Specialization\_area\_id | Unique identifier of the associated specialization area | Numeric |  | Foreign id  Constrained  On delete cascade |

Tabela 8 – Courses

A tabela 8 guarda os dados dos cursos. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará os *cursos*;
* Nome;
* Initials – string the inicias usadas para rapidamente identificar o curso em certas tabelas;
* Specialization\_area\_id – armazena as areas de especialização associadas com aquele curso;

**Course Classes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific course class | Numeric |  | Primary Key |
| Name | Name of the course class | String |  |  |
| Number | Number of the course class | String |  |  |
| Course\_id | Unique identifier for the course associated with the course class | Numeric |  | Foreign id constrained  On delete cascade |

Tabela 9 – Course Classes

A tabela 9 guarda os dados das aulas do curso. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará as *aulas do curso*;
* Nome;
* Number – Numero usado para identificar o curso;
* Course\_id – armazena dados sobre o curso associado;

**Pedagogical Groups**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific pedagogical group | Numeric |  | Primary Key |
| Name | Name of the pedagogical group | String |  |  |

Tabela 10 – Pedagogical Groups

A tabela 10 guarda os dados das aulas dos grupos pedagógicos. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará os *grupos pedagógicos*;
* Nome;

**UFCDs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific UFCD | Numeric |  | Primary Key |
| Name | Name of the UFCD | String |  |  |
| Pedagogical\_group\_id | Unique identifier of the associated pedagogical group | Numeric |  | Constrained |
| Number | Number of the UFCD | String |  |  |
| Hours | Hours of the UFCD | Integer |  |  |

Tabela 11 – UFCDs

A tabela 11 guarda os dados das aulas das UFCDs. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará as *UFCDs*;
* Nome;
* Pedagogical\_group\_id – armazena dados sobre o grupo pedagógico associado;
* Numero – numero usado para rapidamente identificar a UFCD em certas tabelas;
* Hours – total horas de formação de cada UFCD;

**Hour Block Course Classes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific hour block course class | Numeric |  | Primary Key |
| Course\_class\_id | Unique identifier of the associated UFCD | Numeric |  | Foreign id  Constrained  On delete cascade |
| Hour\_beginning | The starting time of the hour block | Time |  |  |
| Hour\_end | The ending time of the hour block | Time |  |  |

Tabela 12 – Hour Block Course Classes

A tabela 12 guarda os dados das aulas dos blocos de hora de aulas de curso. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará os *blocos de hora de aulas de curso*;
* Course\_class\_id – armazena dados do curso associado a certos blocos de hora;
* Hour\_beginning – a hora de início de um bloco de hora;
* Hpir\_end – a hora de fim de um bloco de hora;

**Schedule Attributions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Description** | **Type** | **Length** | **Notes** |
| Id | Unique number used to identify a specific schedule attribution | Numeric |  | Primary Key |
| Date | Date of the schedule attribution | Date |  |  |
| Published | Published status of the schedule attribution | Boolean |  | Default false |
| Hour\_block\_course\_class\_id | Unique identifier of the associated hour block course class | Numeric |  | Foreign id  Constrained |
| Course\_class\_id | Unique identifier of the associated course class | Numeric |  | Foreign id  Constrained  On delete cascade |
| Ufcd\_id | Unique identifier of the associated UFCD | Numeric |  | Foreign id  Constrained  On delete cascade |
| User\_id | Unique identifier of the associated user | Numeric |  | Foreign id  Constrained  On delete cascade |

Tabela 13 – Course Classes

A tabela 13 guarda os dados das aulas das atribuições de disponibilidade. A estrutura desta tabela é a seguinte:

* Id - irá ser único e identificará as *atribuições de disponibilidade*;
* Date – a data onde uma disponibilidade foi atribuída;
* Published – um valor que indica se a disponibilidade foi publicada ou não;
* Ufcd\_id – armazena para qual ufcd a atribuição de disponibilidade foi feita;
* User\_id – identifica qual o utilizador que esta disponibilidade pertence;

1. Desenvolvimento

## 3.1. Arquitetura da aplicação

Decidimos adotar o modelo de arquitetura *Layered n-Tier*, por se apresentar como o que melhor se adequa às necessidades do projeto. Este modelo permite-nos trabalhar com o padrão de design MVC (*Model-View-Controller*), o que proporciona um potencial significativo para manutenção, expansão e escalabilidade da aplicação. Optar por esta abordagem foi uma escolha estratégica, pois permitiu-nos separar claramente as responsabilidades entre as diferentes camadas da aplicação e utilizar ferramentas com as quais a equipa já tem alguma familiaridade.

Na sequência da escolha desta arquitetura, para reduzir o tráfego de pedidos ao servidor e, ao mesmo tempo, manter a consistência e a integridade dos dados, que o padrão MVC em Laravel promete, planeamos utilizar JavaScript para manipulação e apresentação de dados do lado do cliente (*client-side*). Isto permitir-nos-á realizar algumas (simples) operações e atualizações de dados diretamente no navegador do utilizador, sem a necessidade de enviar um novo pedido ao servidor a cada interação. Esta abordagem é valiosa para melhorar a experiência do utilizador e otimizar o desempenho da aplicação, mantendo a coesão e a integridade do padrão MVC no servidor.

## 3.2. Tecnologias

Continuando com a escolha de uma arquitetura em camadas n-Tier e o padrão de design MVC, optamos por uma tipologia MPA (Multiple Page Application) para manter a nossa intenção de facilitar a manutenção e escalabilidade da aplicação. A escolha da framework Laravel foi uma decisão natural, considerando os pontos mencionados até este ponto e a familiaridade prévia de todos os membros da equipa com a Framework Laravel. Além disto, Laravel demonstra ser uma escolha adequada, uma vez que oferece uma arquitetura compatível (*out-of-the-box)* com o modelo de dados apresentado anteriormente.



Uma das vantagens significativas da utilização do *Laravel* é a integração com a *framework* Vue.js, o que simplifica a criação de componentes de interface do utilizador interativos e dinâmicos. Além disso, o *Laravel* também oferece um sistema de login integrado, tornando a gestão de autenticação e autorização uma tarefa mais eficiente em termos de desenvolvimento.

A ferramenta *Laravel Mix* também foi um fator determinante na escolha, uma vez que oferece várias vantagens importantes, incluindo a granularidade (por permitir a integração de Css, Sass e JS) necessária para reduzir o número de pedidos ao servidor com JavaScript, bem como a capacidade de transpilação de JavaScript ES6+ para ES5. Isso é particularmente útil para garantir a retrocompatibilidade com navegadores mais antigos.

O método de encriptação que optámos por utilizar foi o ‘bcrypt’ que é uma função de *hash* de *passwords* projetada para construir uma *hash* criptograficamente segura. Pela investigação feita, percebemos que é um método bastante utilizado numa ampla variedade de plataformas e já com uma longa história comprovada. Além disso, por ser o sistema já utilizado por defeito na *framework* utilizada, garante suporte nativo, tornando a sua integração e manutenção mais simples.

Em suma, a escolha do *Laravel* como *framework* para o desenvolvimento da nossa aplicação ofereceu uma série de benefícios que se alinham perfeitamente com as necessidades do projeto, garantindo eficiência, compatibilidade e escalabilidade.



A escolha do MySQL como base de dados para o nosso projeto foi baseada em vários fatores que surgiram das nossas conversas e considerações até o momento:

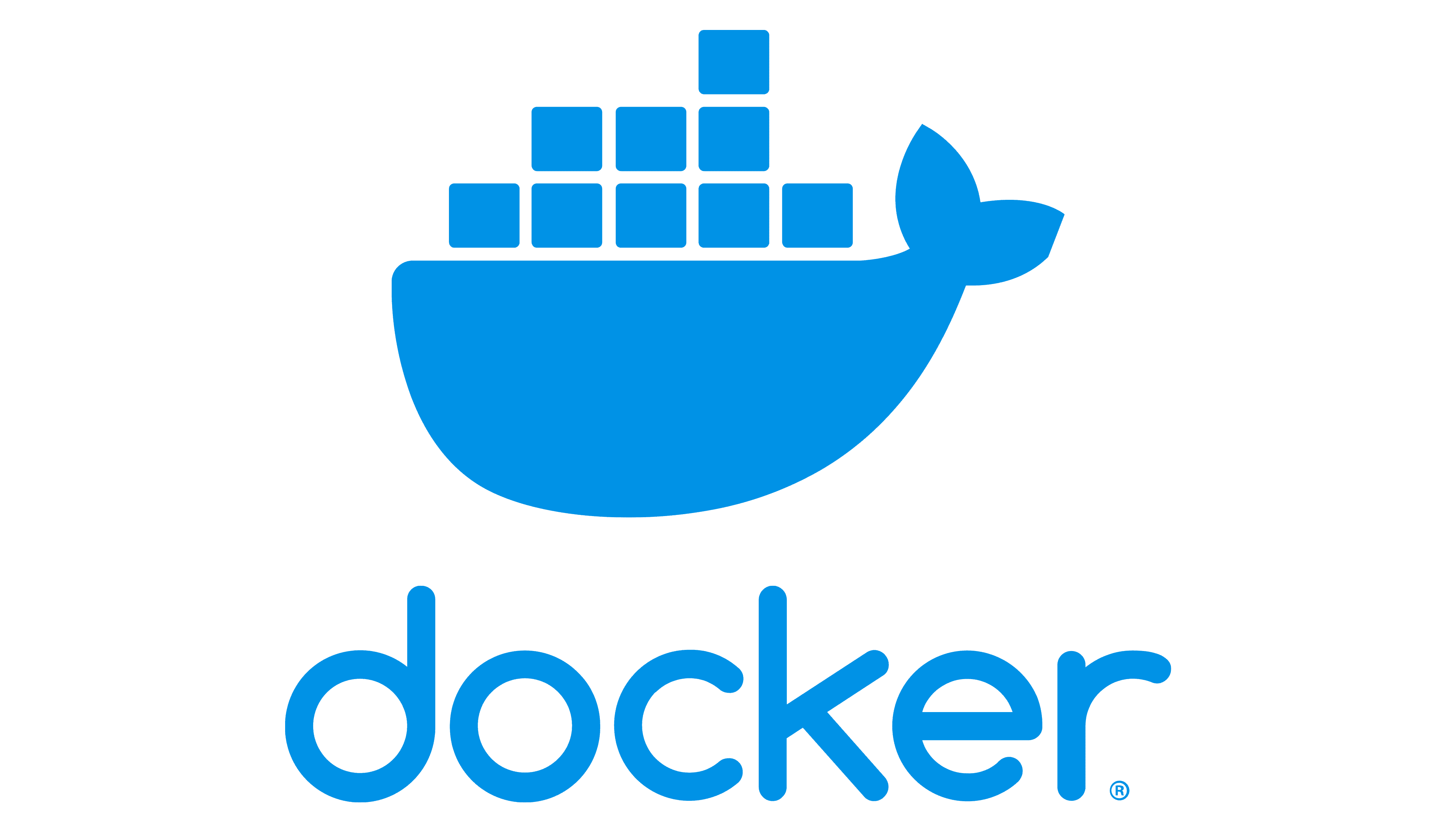
**Compatibilidade com Laravel:** por possuir integração nativa e bem suportada com MySQL. Esta compatibilidade simplifica a configuração e a interação com a base de dados, tornando o desenvolvimento mais eficiente.

**Desempenho e Escalabilidade:** O MySQL é conhecido pelo seu desempenho sólido e capacidade de escalabilidade. Isto é importante para garantir que nossa aplicação possa lidar com um aumento de utilizadores e volume de dados à medida que cresce, *multi-thread processing*.

**Segurança e Confiabilidade:** O MySQL oferece recursos robustos de segurança e é amplamente utilizado em aplicações críticas em todo o mundo. Isso dá-nos confiança na segurança e integridade dos dados dos utilizadores.

**Comunidade e Suporte:** O MySQL possui uma grande comunidade de utilizadores e é amplamente adotado na indústria. Isto significa que há uma abundância de recursos, documentação e suporte disponíveis para resolver problemas e otimizar o desempenho, caso surjam desafios durante o desenvolvimento e manutenção.

Em resumo, a escolha do MySQL como base de dados para o nosso projeto foi guiada pela sua compatibilidade com o Laravel, desempenho, escalabilidade, segurança, confiabilidade e suporte da comunidade, todos eles fatores que se alinham com os requisitos e objetivos da nossa aplicação.



A utilização do Docker neste projeto é vantajosa por várias razões:

**Isolamento e Portabilidade:** O Docker permite criar contentores isolados que empacotam (*containerize*) a aplicação, juntamente com todas as suas dependências e configurações necessárias. Isso garante que a aplicação seja executada consistentemente em qualquer ambiente de desenvolvimento, teste ou produção. A portabilidade facilita a implantação em diferentes servidores ou na cloud.

**Gestão de Dependências:** Com o Docker, podemos definir todas as dependências da aplicação, incluindo versões específicas de bibliotecas, servidores web, bases de dados, etc., num arquivo de configuração (*Dockerfile*). Isto elimina problemas de compatibilidade e garante que todos os membros da equipe tenham as mesmas configurações.

**Facilitação do Desenvolvimento Colaborativo:** O Docker simplifica o desenvolvimento colaborativo, pois todos os membros da equipe podem usar o mesmo ambiente Docker. Isto reduz a ocorrência de problemas relacionados com diferenças no ambiente de desenvolvimento local.

**Testes e Integração Contínua:** Os contentores Docker podem ser facilmente integrados em pipelines de integração contínua (CI/CD), facilitando a execução de testes automatizados em ambientes idênticos aos de produção.

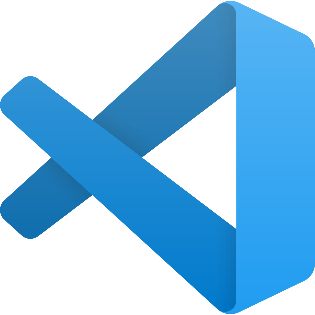
**Manutenção Simplificada:** Com o Docker Compose, é possível orquestrar vários contentores, como a aplicação e a base de dados, num único ambiente. Isto simplifica a gestão e manutenção da infraestrutura do projeto.

**Flexibilidade de Tecnologia:** O Docker é compatível com uma variedade de tecnologias, o que permite a escolha de ferramentas e serviços que melhor atendam às necessidades do projeto.

**Reutilização de Contentores:** Se a equipa desenvolver outros projetos futuros, os contentores Docker podem ser reutilizados, economizando tempo e esforço no *Environment Setup*.

Em resumo, o Docker oferece uma solução eficaz para padronizar, gerir e escalar a infraestrutura do projeto, facilitando o desenvolvimento colaborativo e garantindo a consistência em diferentes ambientes. Isto pode contribuir significativamente para o sucesso do projeto.

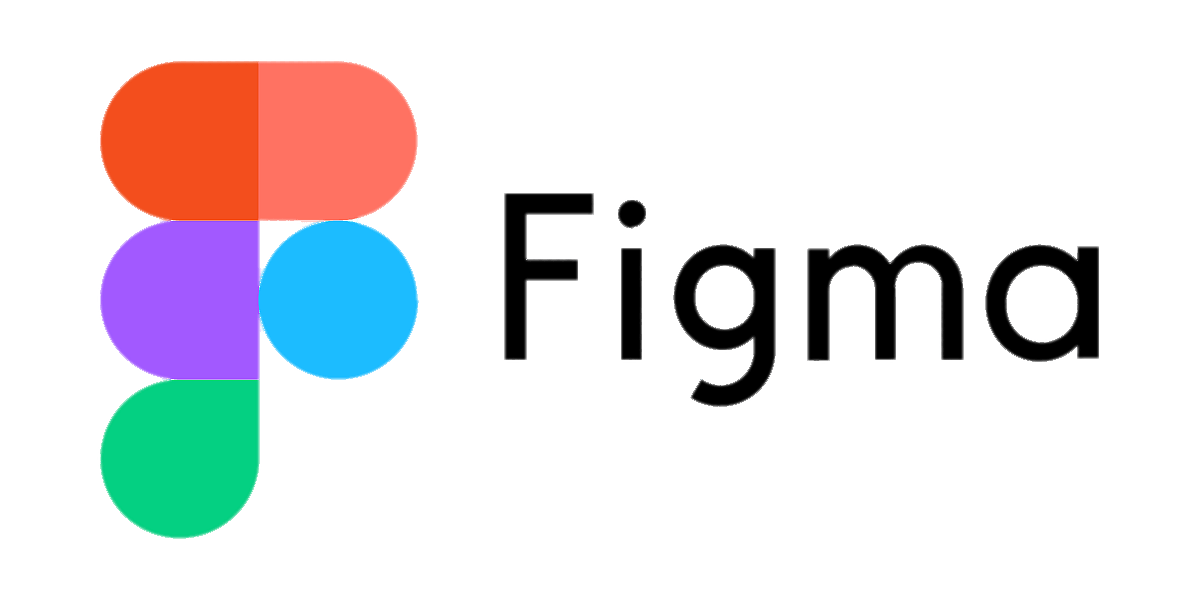
## 3.3. Ferramentas de desenvolvimento



Como editor de código para o presente projeto, foi utilizado o *Visual Studio Code*, por ser o editor com que o grupo já tinha trabalhado durante todo o curso e se encontrava mais familiarizado.



Como ferramenta de gestão de versões recorremos ao *GIT* com repositório criado no *Github*, e para auxiliar na gestão local utilizamos o *Giuthub desktop*. Todas as ferramentas foram selecionadas por familiaridade dos elementos do grupo com as mesmas.

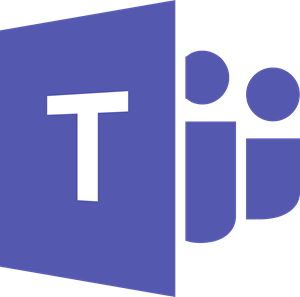
Foi desenvolvido no Excalidraw e Figmaum layout que traduzisse uma forma possível de apresentação e manipulação dos dados que se alinhasse com o pretendido pelo cliente**.**

Todos os restantes diagramas, gráficos e fluxogramas foram desenvolvidos em Draw.io e no caso particular do BPMN recorremos a versão 2 deste tipo de notação tendo a nossa equipa recorrido ao Camunda.

Como referenciado anteriormente ferramentas para gestão e de distribuição de tarefas, optámos por uma combinação do *Team Gant* e *Trello*.



Para coordenação, comunicação, pedidos de ajuda e partilha de informação foi criado um grupo na aplicação teams.

## 3.3. Design de Interfaces

## 3.4. Processo do Desenvolvimento

### 3.4.1. Arquitetura e Prototipagem

|  |  |
| --- | --- |
| Arquitetura e prototipagem 14/7/23 – 4/8/23 (3 semanas) | |
| Objectivo Principal | * Desenho/Arquitetura do sistema e Prototipagem. |
| Descrição | * 1ª reunião com cliente; * Levantamento de requisitos funcionais/não-funcionais; * Diagrama de casos de uso; * Diagramas de Sequência, casos de destaque; * BPMN v2; * Modelo de dados; * Diagrama de Classes; * UI flow diagram; * Protótipagem; * 2ª reunião com cliente Apresentação protótipo. |
| Dificuldades | * Decomposição e análise do problema de forma analítica com base na informação fornecida pela cliente. |
| Conquistas | * Objectivo principal alcançado; * Através de dialogo com a cliente e alguns formadores (utilizadores finais) conseguimos desambiguar alguns bloqueios. |

### 3.4.2. Environment Setup

|  |  |
| --- | --- |
| Environment Setup 5/8/23 – 13/8/23 (1 semana) | |
| Objectivo Principal | * Criação e setup do ambiente de desenvolvimento. |
| Descrição | * Docker *containerize* do projecto; * Implementação do modelo de dados; * Dummy data com faker; * Implementação de login base através de token, componente VUE auth; * Master page com header & footer; * Criação do repositório Git para o projeto. |
| Dificuldades | * Tecnologia nova para o grupo que gerou alguma confusão no seu setup inicial, na definição e configuração de dependências e containerização do servidor de MySql. Demos prioridade a operacionalização do projeto, sendo que posteriormente revisitaremos esta implementação. |
| Conquistas | * Objetivo principal alcançado; |

### 3.4.3. Sprint I

|  |  |
| --- | --- |
| Sprint I 14/8/23 – 20/8/23 & 28/8/23 – 3/9/23 (2 semanas) | |
| Objectivo Principal | * Criação dos componentes centrais do projeto. |
| Descrição | * Scheduler geral(agenda e date picker); * Content table geral; * Button geral; * Warning message modal; * Master page (distinção Formador e Planeamento); * Implementação envio emails. |
| Dificuldades | * Criação de componentes generalizados que se adaptassem aos vários use-cases; * Concretização da implementação de emails; * Distinção de autenticação Formador/Planeamento. |
| Conquistas | * Conclusão retirada de que overhead necessário para conseguir implementar esta solução não compensaria o investimento de tempo. * Reestruturação/redefinição de estratégia, de componentes para páginas. Melhor leitura, melhor escalabilidade, mas fácil manutenção e troubleshooting. |

### 3.4.4. Sprint II

|  |  |
| --- | --- |
| Sprint II 4/9/23 – 10/9/23 (1 semanas) | |
| Objectivo Principal | * Configuração e testes de CRUD |
| Descrição | * Reestruturação do processo de desenvolvimento; * Implementação das operações de CRUD necessárias para todas as entidades do nosso modelo de dados; * Páginas operacionais de teste para operações CRUD. |
| Dificuldades | * Dificuldades manifestadas por alguns membros com a framework; * Primeiras experiências com Laravel-Mix. |
| Conquistas | * Objectivo principal conseguido; * Primeira conquista do grupo com Laravel-Mix e JS/CSS. |

### 3.4.5. Sprint III

|  |  |
| --- | --- |
| Sprint III 14/9/23 – 01/10/23 (3 semanas) | |
| Objectivo Principal | * Desenvolvimento de BL e aplicação de Use Cases |
| Descrição | * Implementação do layout; * Desenvolvimento da lógica e verificações necessárias à realização das operações necessárias. |
| Dificuldades | * Inclusão de JS com Laravel na criação de uma página única para as operações de *gestão gerais;* * Gestão de tempo entre responsabilidades de gestão, desenvolvimento e prestação de apoio técnico à equipa. O que levou a um atraso das tarefas consignadas ao Product Owner; * Durante os vários encontros/*daylies* da equipa ajuda era disponibilizada e sempre que detetadas duvidas ou a possibilidade de interpretações erróneas as mesmas eram mitigadas com as devidas explicações, partilha de código, documentação até feedback positivo da equipa em relação a estas intervenções. * Aquando da deteção de múltiplos atrasos e falhas de comunicação ou a não procura de ajuda/exposição de problemas nas *daylies*, tentou-se 1 on 1’s com os elementos da equipa. Houve elementos que manifestaram que não precisavam de ajuda nesta fase. * Falhas de comunicação continuas levaram à necessidade de uma reunião com a supervisão do Coordenador de turma de forma a tentar entender como se poderia ultrapassar este desafio. Conclui-se a necessidade de uma antecipação do trabalho de *Frontend* de forma a reposicionar os colegas que manifestaram, nesta reunião, mais dificuldades com *Backend.* Mas que para isso seria necessário terminar as tarefas pendentes para criar as condições mínimas de arranque de aprimoramento e desenvolvimento do *Frontend.* * Sprint concluído com tarefas em incompletas que terão de ser reintegradas em Backlog para desenvolvimento em futuro sprint. |
| Conquistas | * Conseguimos detetar e começar a trabalhar os bloqueios e falhas de comunicação; * Reorganização e priorização de tarefas. |

### 3.4.6. Sprint IV

|  |  |
| --- | --- |
| Sprint IV 02/10/23 – 08/10/23 (1 semana) | |
| Objectivo Principal | * Passagem de *Backend* para *Frontend* |
| Descrição | * Implementação de sistema de login com distinção de users/rotas; * Conclusão da aplicação de BL e use case a páginas de *scheduller*; * Ecxel/PDF export system; * Notificações de Email. |
| Dificuldades | * Na conclusão do sprint os *schedullers* ficaram por fazer, devido a necessidades de acompanhamento e pedidos de ajuda de outros elementos da equipa e continuação do desenvolvimento do JS necessário a implementação de uma lógica SPA para as páginas de gestão geral, o que para além de aliviar carga sobre o servidor, reduz pontos de falha, *debug*, manutenção e futuras implementações melhorias de *frontend;* * Sistema de login, houve necessidade de realocacao da tarefa a outro membro do grupo, por indisponibilidade do primeiro; * Notificações de email , houve necessidade de realocacao da tarefa a outro membro do grupo, por indisponibilidade do primeiro; * Terefas em falta de sprint III e IV e ainda não mencionadas realocadas a outros elementos de *backend* de forma a libertar elementos para desenvolvimento de *frontend*; |
| Conquistas | * Sistema de exportação para PDF (de acordo com reunião com a cliente), aguarda que *schedullers* sejam finalizados para testes e conclusão de implementação; * Restruturação de equipa para direcionar colegas para áreas distintas de intervenção. |

### 3.4.6. Sprint V

|  |  |
| --- | --- |
| Sprint IV 08/10/23 - (3 semana) | |
| Objectivo Principal | Encerramento projeto |
| Descrição | Fionalização Backend  Finalização Frontend |
| Dificuldades | Algumas falhas de comunicação, mas em muito menor gravidade e com a devida atenção resolvidas na altura.  Alguns atrasos e reajustes na calendarização das tarefas. |
| Conquistas | Requesitos funcionais respondidos; |

Anexos

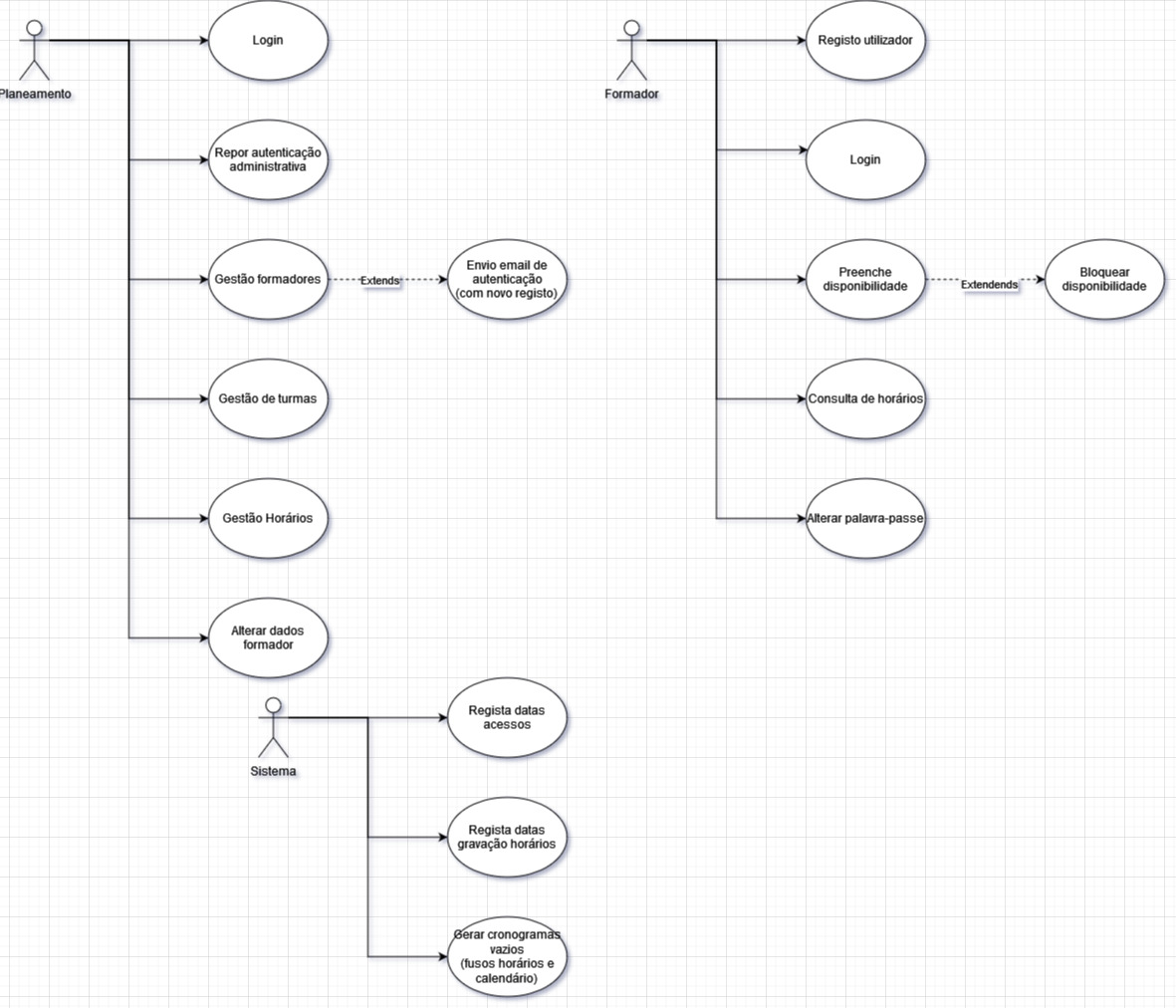


diagrama de casos de uso

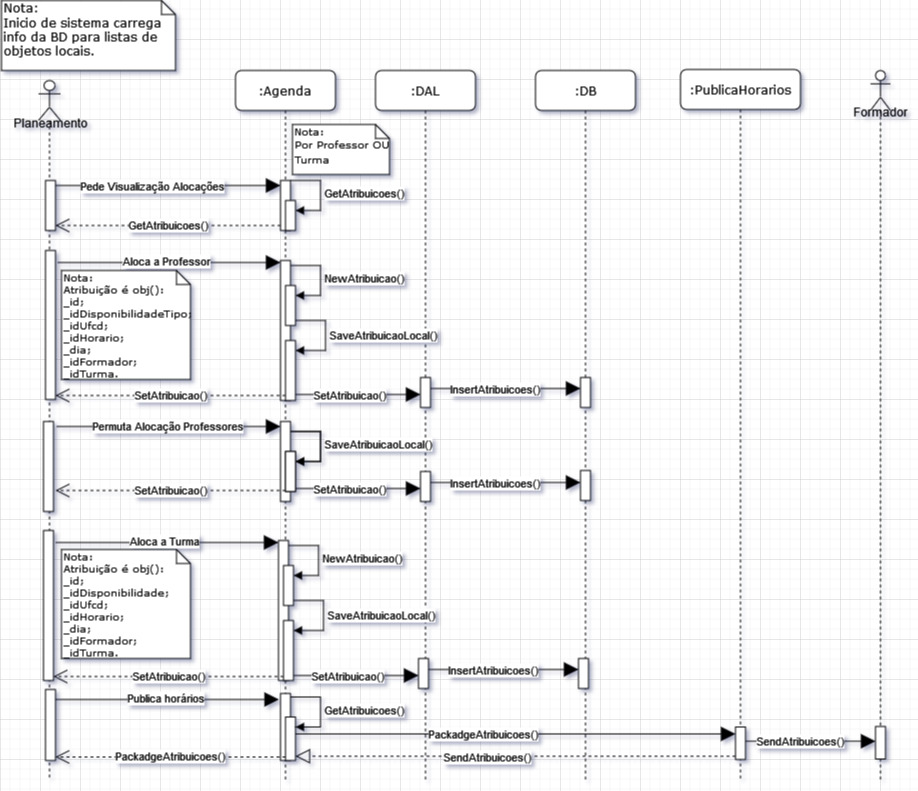


diagrama de sequência – gestão

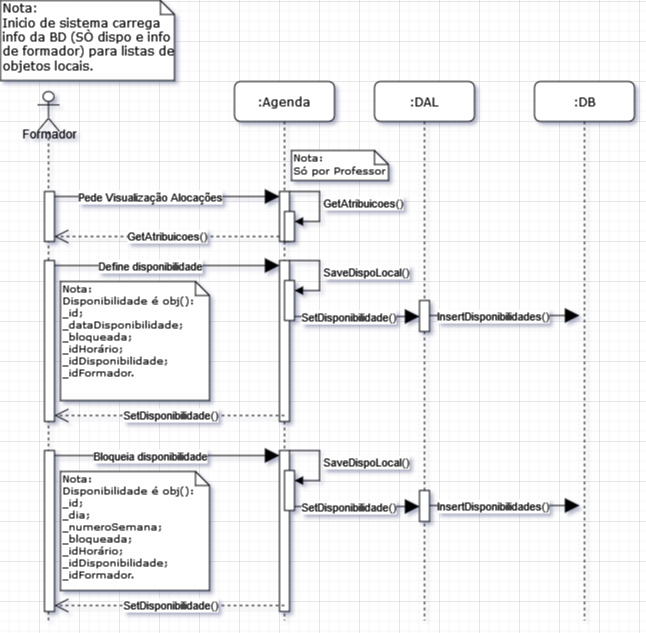
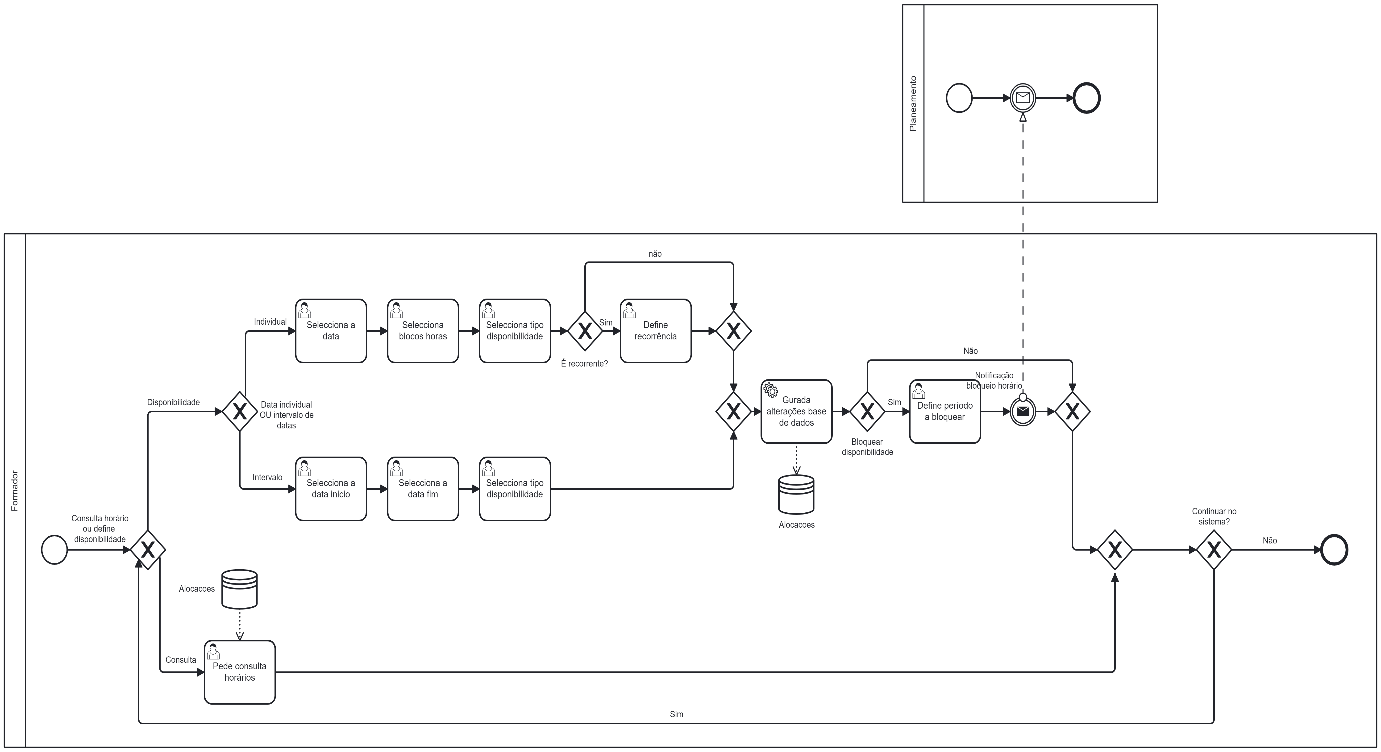
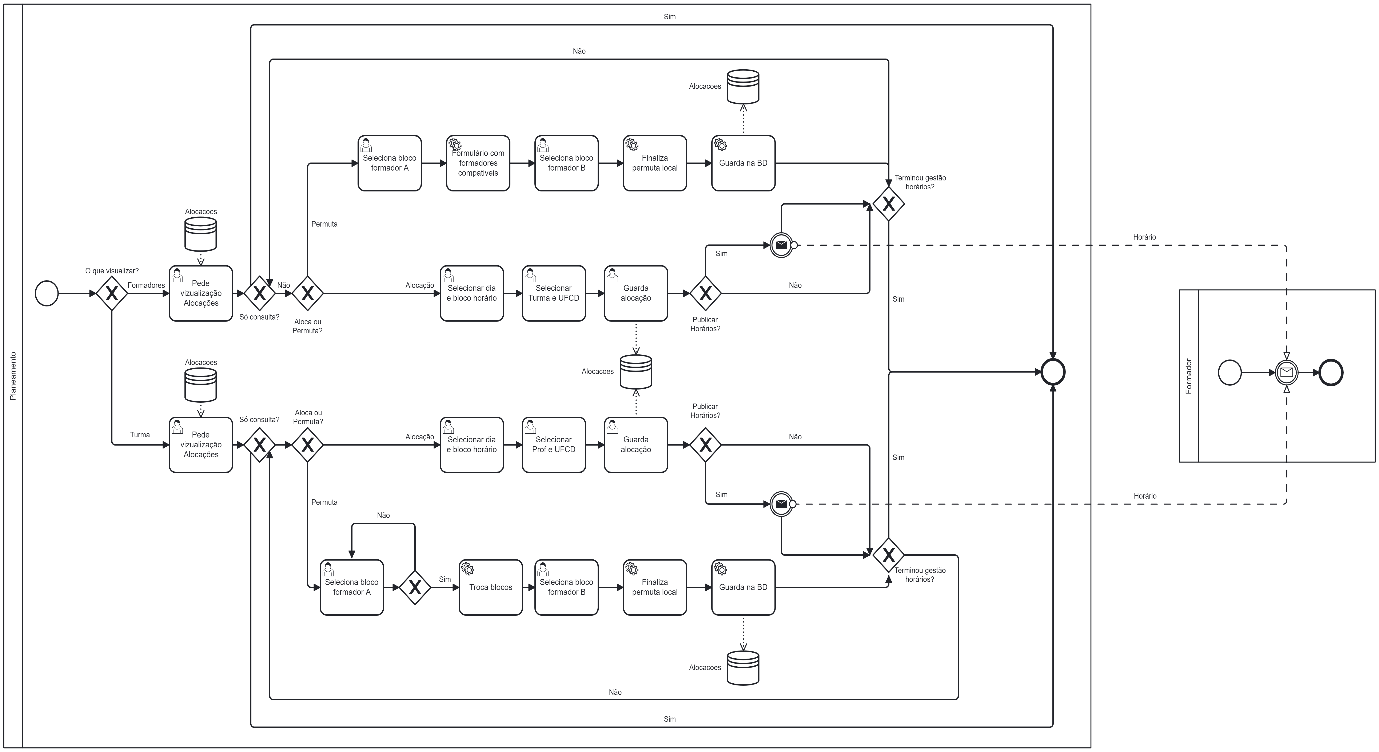


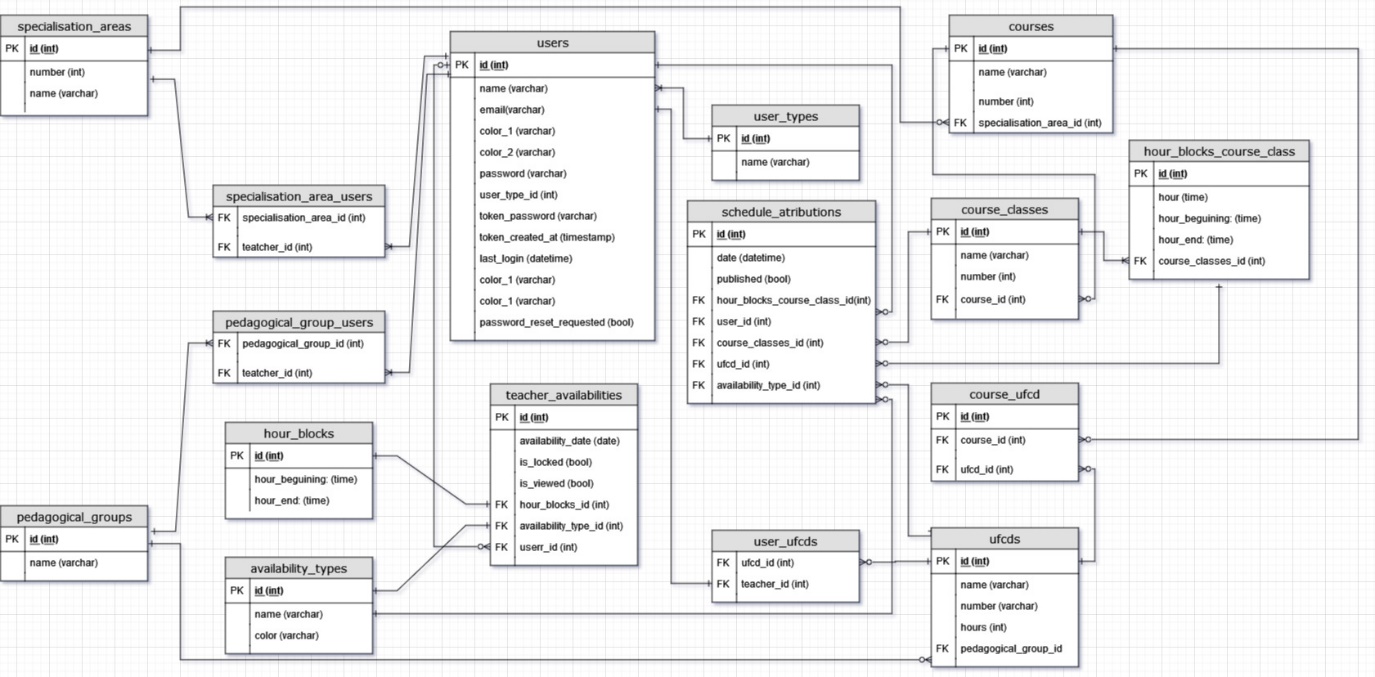
diagrama de sequência – formador



bpmn – formador



bpmn – planeamento



modelo de dados

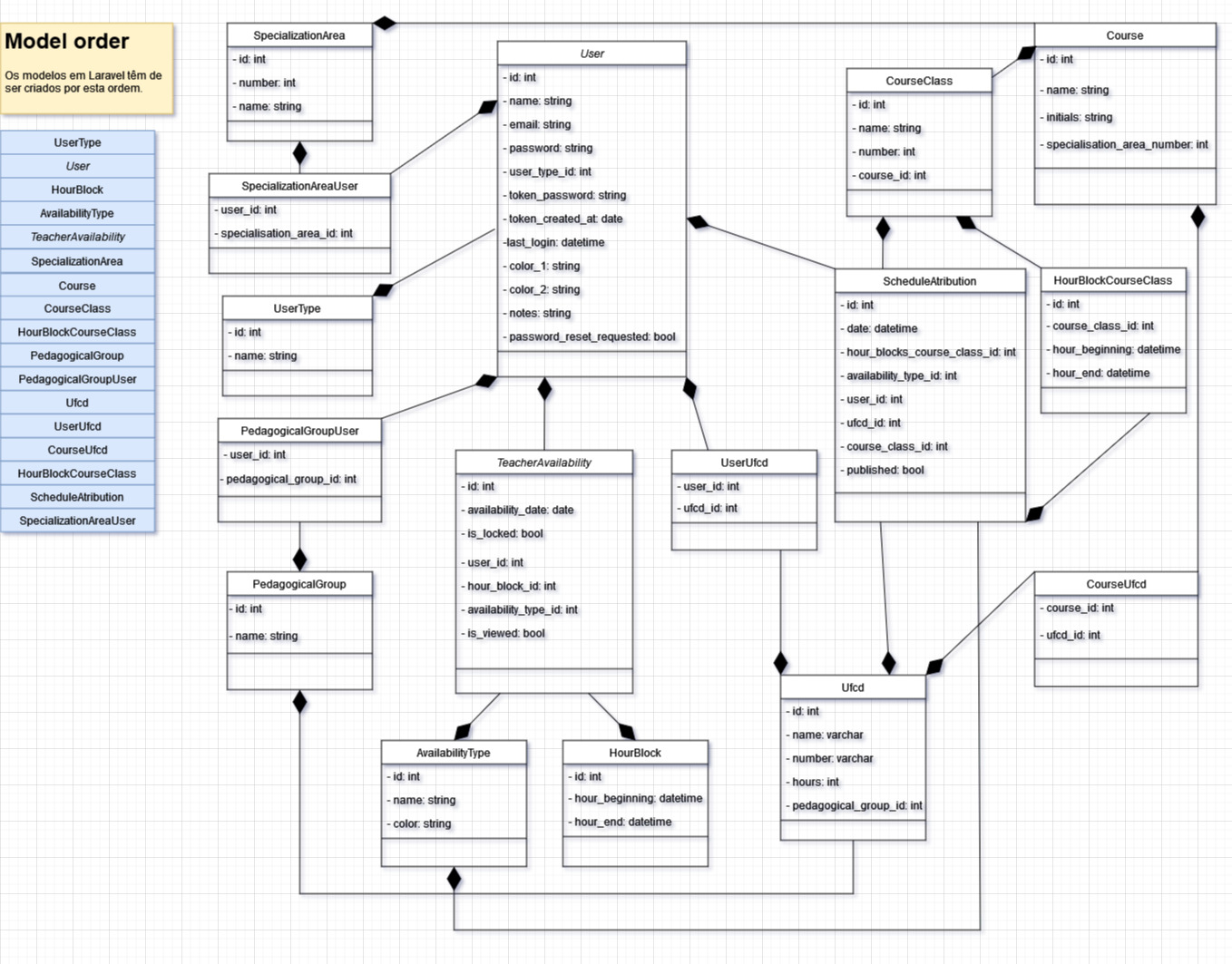
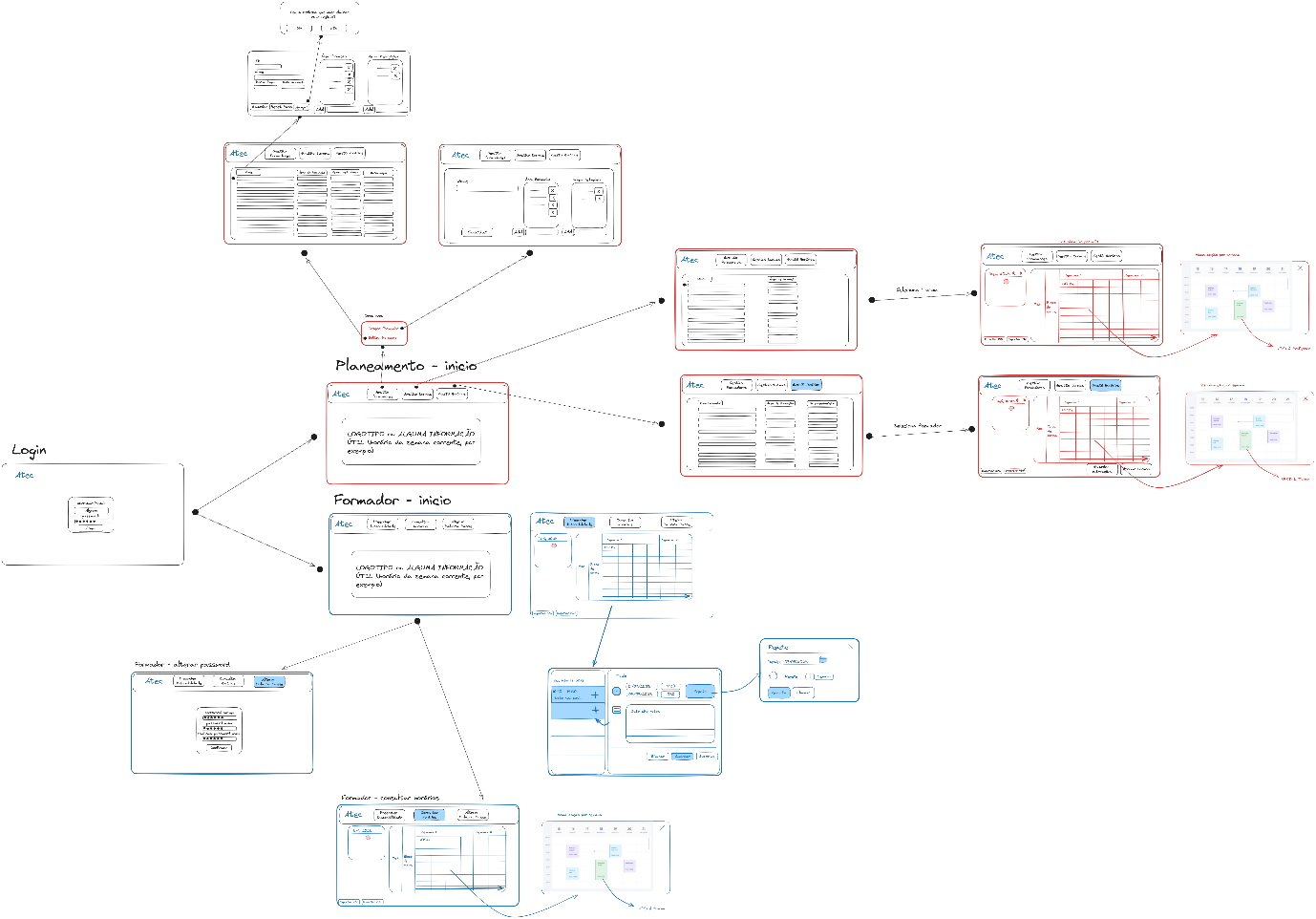


diagrama de classes



user interface