Contents

[Introdução 3](#_Toc168852399)

[Capítulo 0 – Estrutura da equipa e metodologias de gestão aplicadas 4](#_Toc168852400)

[Trello 4](#_Toc168852401)

[Project 5](#_Toc168852402)

[GitHub 6](#_Toc168852403)

[Capítulo 1 – Funcionamento de um CMS 8](#_Toc168852404)

[Funcionamento de um CMS e o seu fluxo 8](#_Toc168852405)

[As várias etapas do fluxo de um CMS 9](#_Toc168852406)

[Exemplos de CMS no mercado 9](#_Toc168852407)

[EasyChair 9](#_Toc168852408)

[Capítulo 2 – Especificidades técnicas 10](#_Toc168852409)

[2.1 Requisitos funcionais 10](#_Toc168852410)

[2.1.1 Fase 1 - Inicialização 10](#_Toc168852411)

[2.1.2 Fase 2 - Submissão 11](#_Toc168852412)

[2.1.3 Fase 3 – *Bidding* 11](#_Toc168852413)

[2.1.4 Fase 4 – Revisões 12](#_Toc168852414)

[2.1.4 Fase 5 – Pré-Conferência 12](#_Toc168852415)

[2.2 Requisitos não funcionais 13](#_Toc168852416)

[2.2.1 - Configurabilidade 13](#_Toc168852417)

[2.2.2 – Modularidade (Not sure about this one) 13](#_Toc168852418)

[2.2.3 - Extensibilidade 13](#_Toc168852419)

[2.2.4 - Compatibilidade 13](#_Toc168852420)

[2.2.5 - Confiabilidade 13](#_Toc168852421)

[2.2.6 - Segurança 14](#_Toc168852422)

[2.2.7 - Usabilidade 14](#_Toc168852423)

[As nossas páginas por cada etapa do processo 15](#_Toc168852424)

[Fase de Inicialização 15](#_Toc168852425)

[Fase de Submissão 17](#_Toc168852426)

[Papéis de utilizador permitidos 18](#_Toc168852427)

[Owner 18](#_Toc168852428)

[Chair 18](#_Toc168852429)

[Author 20](#_Toc168852430)

[Capítulo 3 – Arquitetura e características da aplicação 21](#_Toc168852431)

[Arquitetura da aplicação 21](#_Toc168852432)

[Tecnologias utilizadas no Front-End 21](#_Toc168852433)

[Modularidade na Aplicação 22](#_Toc168852434)

[UI e UX 25](#_Toc168852435)

[Compreensão dos elementos visuais 27](#_Toc168852436)

[Utilização de *Cards* 27](#_Toc168852437)

[Estrtutura do projeto 35](#_Toc168852438)

[Contextos e gestão de comportamentos 37](#_Toc168852439)

[Descrição dos nossos componentes e layouts 40](#_Toc168852440)

[Backend 43](#_Toc168852441)

[Estrutura do backend: 43](#_Toc168852442)

[Dependências Principais do Projeto 46](#_Toc168852443)

[Routes 49](#_Toc168852444)

[PassportStrategies 54](#_Toc168852445)

[Módulo Utility 57](#_Toc168852446)

[Segurança 59](#_Toc168852447)

[Autenticação com Passport.js 59](#_Toc168852448)

[Encriptação de Palavras-Passe com Bcrypt 60](#_Toc168852449)

[Deploy do Frontend no Vercel 60](#_Toc168852450)

[Base de Dados em Neon.tech 60](#_Toc168852451)

[CORS (Cross-Origin Resource Sharing) 60](#_Toc168852452)

[Envio Seguro de Emails com Nodemailer 60](#_Toc168852453)

[Diagramas 61](#_Toc168852454)

[Base de dados 61](#_Toc168852455)

[Capítulo 4 – Implementação 62](#_Toc168852456)

[Deploy do nosso backend 62](#_Toc168852457)

# Introdução

Se formos à nossa história, a primeira vez que se deu vida à palavra “Conferência”, leva-nos para os tempos de 1527 e nestes tempos, esta palavra possuía um significa de “Uma reunião de duas ou mais pessoas sobre um determinado tópico”, claro que atualmente conseguimos perceber que esta palavra ganhou um significado mais específico e megalómano sendo agora mais caracterizado como “discurso ou palestra sobre temas literários, artísticos, científicos, políticos ou religiosos”, como tal e generalizando um pouco mais para os dias de hoje entenda-se uma conferência como uma reunião oficial que poderá ter um período de duração de alguns dias e que leva a cabo a capacidade da partilha e discussão de um determinado tema ou conjunto de temas relacionados com uma ou mais áreas especializadas.

Atualmente existem inúmeros formatos de conferência, como por exemplo: Painéis de discussão, Apresentações orais, Workshops, conferências virtuais, entre muitos outros formatos o que ainda permite que o conceito se torne mais abrangente e chegue a diversas áreas, desde áreas científicas como áreas da educação; literacia; cinema ou outros formatos de arte.

Tentámos encontrar alguns dados relativos ao número de conferências que foram dadas nos últimos anos, no entanto e devido à diversidade de eventos bem como devido à falta de uma fonte centralizada de dados não nos foi possível trazer para este trabalho números concretos, mas foi-nos possível obter que no ano 2012 foram feitas 1833200 conferências/congressos/reuniões apenas nos Estados Unidos da América (de acordo com o estudo feito, consideraram uma reunião como o ajuntamento de 10 ou mais pessoas durante um período de tempo).

Na era digital atual, a organização e a gestão de conferências e eventos académicos enfrentam desafios únicos relacionados com a eficiência, acessibilidade e automatização dos processos. A gestão eficaz de conferências exige uma abordagem sistemática que consiga lidar com a complexidade de organizar múltiplas tarefas. Neste contexto, surge a necessidade de desenvolver sistemas de gestão de conferências (CMS) que não só simplifiquem estas operações, mas também as tornem mais eficientes e menos sujeitas a erros.

Este projeto visa o desenvolvimento de um CMS, o qual intitulamos de “UAL Conf”, onde desenvolvemos a pensar em ser inovador, utilizando a framework React e Node.js, para auxiliar na organização de conferências. A escolha destas tecnologias baseia-se na sua capacidade de criar aplicações web robustas, escaláveis, acessíveis e também devido à sua capacidade de modularidade. O software proposto tem como objetivo principal automatizar e agilizar as diversas etapas envolvidas na organização de conferências, desde a fase pré-conferência até à definição do programa do evento, proporcionando uma base sólida para etapas subsequentes como o planeamento de infraestruturas e logística.

# Capítulo 0 – Estrutura da equipa e metodologias de gestão aplicadas

## Trello

Durante as primeiras fases do nosso projeto, fomos sentindo uma dificuldade acrescida para fazer a nossa gestão interna de forma que conseguíssemos atribuir de forma eficaz e justa as tarefas a cada membro do grupo, mas mais do que isso, começámos a sentir uma enorme entropia entre as ferramentas que iríamos utilizar e/ou a onde é que poderíamos deixar ficheiros/dados de forma que todos tivessem acesso e de uma forma organizada e estruturada. Por assim dizer, foi sentido que o nosso início foi um pouco atribulado. Em março decidimos reunir todos para podermos colmatar este problema de forma a termos mais rendimento no nosso trabalho.

Como solução para o nosso problema, decidimos aplicar a metodologia ágil *Kanban* para gerir o nosso projeto através da ferramenta *Trello*. No nosso *Board* criámos 6 colunas:

* *To Do*: Todos os pontos que fossem levantados durante as reuniões ou pequenas situações que cada um de nós fosse-se lembrando à medida dos desenvolvimentos, iam sendo adicionados nesta lista;
* *In Progress:* Sempre que um de nós fosse começar uma tarefa, colocávamos de imediato o card atribuído a nós nesta lista o que permitia toda a equipa estar a par do que é que cada um estava a fazer naquele momento;
* *In Analysis*: Esta lista mostra situações que ainda não existem grandes certezas e como tal estão em processo de ser analisadas e como tal debatidas entre todos nós, inclusive o nosso orientador;
* *Bugs*: Tal como o nome indica, sempre que fosse detetado um bug durante as vezes que estávamos a fazer testes ou simplesmente a mexer no nosso site e fosse encontrado algum bug, era logo aberto um card nesta lista. Desta forma não perdíamos o rastro aos bugs que iriam ser detetados e todos nós saberíamos quando é que os mesmos tinham sido resolvidos e por quem;
* *For Testing*: Sempre que uma página ou um desenvolvimento mais específico ia sendo desenvolvido como uma função muito específica, colocávamos um card nesta lista e atribuímos todos os elementos do grupo a este, desta forma poderíamos ir cumprindo com pequenos ciclos de vida de cada funcionalidade (requisitos; desenvolvimento; testes e implementação) para além de termos a vantagem de ser quatro pessoas a testar o que aumentava a probabilidade de encontrar bugs caso existissem;
* *Finished*: Esta lsita mostrava todos os cards implementados no nosso projeto. Quando um card chega aqui, significa que já passou por um ciclo de vida.

Screens screenshot of a phone

Description automatically generated

Para gestão interna, aproveitámos também todos os momentos oportunos para nos reunir caso sentíssemos essa necessidade, sendo que tínhamos sempre uma reunião semanal para ponto de situação do projeto, sendo esse dia no domingo às 14:00 da tarde.

Nesta reunião acabávamos sempre por debater alguns conceitos relevantes do contexto do nosso projeto, atribuir *cards* do Trello a cada um dos elementos, verificar se o nosso *Board* estava o mais atualizado possível para evitar entropias na nossa gestão e ainda discutir alguns prazos por cumprir, do projeto.

## Project

Uma outra ferramenta que utilizámos para a nossa gestão interna foi o “Project”. Com esta ferramenta foi nos possível criar uma linha do tempo de todas as tarefas bem como antever o que cada membro iria realizar no próprio projeto. Uma das grandes vantagens desta ferramenta foi a capacidade de permitir criar uma ideia geral dos prazos a cumprir e gerir as expetativas do nosso desenvolvimento (que páginas conseguimos criar no tempo determinado? Quantas horas precisamos para uma determinada funcionalidade e/ou quantas pessoas são realmente necessárias para desenvolver essa funcionalidade? Entre outras questões).

Para além da nossa reunião semanal, aproveitávamos também as aulas de Laboratório de projeto para nos reunirmos presencialmente de forma a podermos avançar em qualquer ponto que pudesse ser avançado.

## GitHub

Uma terceira ferramenta utilizada não só para gestão da equipa, como dos nossos códigos foi o GitHub. Desde o início do projeto que abrimos um repositório no GitHub e foi neste que fizemos toda a nossa gestão.

No branch main é onde se encontra todo o nosso código atualizado. Sempre era feito um novo desenvolvimento ou alguma modificação para testar algum comportamento e que não saberíamos se faria sentido de incluir no main, abríamos um novo branch e fazíamos os nossos commits nesses mesmos branchs desta forma tínhamos uma “camada” que nos protegia o código que efetivamente estava a funcionar como o esperado do código que poderia conter anomalias ou comportamentos indevidos. Caso os branchs fizessem sentido de passar para o branch main, então a dada altura o mesmo era merged com o main e de seguida apagado.

Uma outra decisão que nos levou a utilizar o github, prende-se com a capacidade de termos o controlo de versões do nosso projeto. Houve várias vezes a necessidade de revertemos certos commits ou fazermos checkout para commits mais antigos para comparar código e comportamentos o que nos possibilitou muita das vezes resolver muitos bugs e garantir a implementação de código com melhor qualidade.

Uma outra ferramenta que utilizámos muito e que nos ajudou a garantir a “saúde” do nosso código, foi a criação de local *Staches* que são no fundo branchs locais que apenas são visíveis ao utilizador que o criou e mais ninguém. Normalmente quando queríamos fazer alguma experiência, mas não sabíamos se seria catastrófica para o código principal, criávamos um *stash* para experimentar, caso a experiência fizesse sentido de ser implementada, faríamos então o commit para o main das mesmas alterações e de seguida apagaríamos o stash.

Apesar do front-end apelativo e intuitivo do próprio GitHub tal como as diversas ferramentas que o mesmo oferece, acabamos por utilizar um outro software chamado de Fork que serve precisamente como cliente de Git onde apresenta uma interface altamente direcionada para a gestão de repositórios. O Fork é um projeto de open source criado por dois programadores Dan Pristupov e Tanya Pristupova.

A seguinte imagem mostra a interface do Fork, onde no ecrã principal temos todos os commits feitos pelos elementos dos nossos grupos, bem como os diversos branchs e merges entre os mesmos.

A screenshot of a computer

Description automatically generated

# Capítulo 1 – Funcionamento de um CMS

## Funcionamento de um CMS e o seu fluxo

Existem algumas formas mais utilizadas e conhecidas para se gerir uma conferência, tais como:

1. “In-house Management” *(Gestão dentro de casa)*: Tipo de gestão onde uma equipa interna de uma organização, fica totalmente responsável por fazer toda a gestão da conferência, o que significa que a organização tem controlo total desta gestão. Tal como é expectado, este tipo de gestão consome bastante tempo e recursos, dado que é necessária uma equipa a tempo inteiro a lidar com todas as variáveis envolventes.
2. Gestão conjunta: Este tipo de gestão, tem uma abordagem mais colaborativa, uma vez que já existe uma entidade externa à organização, normalmente costumam ser empresas contratadas e especializadas em gestão de conferências/eventos, também chamada de: “professional conference organizer“(PCO). Para este modelo, as responsabilidades são partilhadas entre ambos o que acaba por ter um por existir uma partilha de controlo e também uma menos sobrecarga.
3. “Third-party Organizers” *(Organizadores externos)*: Este tipo de gestão, é o tipo de gestão que o nosso projeto poderá permitir, ou seja, existe um *software* especializado ou plataformas especializadas em gestão de eventos e toda a gestão acaba por ser feita nestes elementos. O responsável da conferência acaba por ter de confiar nas capacidades destas ferramentas e apenas faz o controlo de qualidade dos dados introduzidos bem como também toma certas decisões. Este tipo de gestão acaba por facilitar a carga de trabalho que uma organização tem, no entanto perde também o controlo absoluto permitindo assim haver um foco no conteúdo das conferências.
4. Gestão conduzida por voluntários: Tipo de gestão normalmente associada a fins não lucrativos/comunitárias e gerida por pessoas voluntárias. São estes que fazem toda a gestão e decisões envolventes no processo. Este tipo de gestão atribui uma enorme carga sobre os voluntários, mas mais uma vez o poder de decisão poderá ser total ou quase total absoluto uma vez que poderá também envolver algum envolvimento de uma entidade de gestão de conferências.
5. Gestão conduzida por patrocinadores: Normalmente estas conferências acabam por ser patrocinadas e financiadas por determinados patrocinadores. São estes os responsáveis por definir o conteúdo principalmente das conferências, bem como a tomada de decisão.

Tal como foi mencionado no ponto 3, o nosso projeto permite a uma pessoa/organizador, através da ferramenta que desenvolvemos, fazer a gestão de uma conferência. Onde oferecemos a gestão simplista das primeiras etapas da conferência, podendo retirar a carga de cima da organização responsável da conferência. É uma solução que já existe no mercado, no entanto procuramos oferecer simplicidade e elegância.

## As várias etapas do fluxo de um CMS

O CMS facilita o processo desde a criação de uma conferência até a publicação dos resultados. Abaixo estão as várias etapas do fluxo de um CMS:

### Configuração Inicial da Conferência

* Configuração das definições da conferência: nome da conferência, *webpage* da conferência (caso tenha), descrição, etc.
* Determinar o comité: Organização das funções dos utilizadores (administradores, revisores, autores, etc.).
* Personalização do Sistema: Ajustes na interface e configurações específicas da conferência (datas de cada fase, a possibilidade de atualizar as submissões, etc.).

### Submissão de Artigos

* Registo do Autor: O autor que quiser criar uma submissão necessita de se registar no sistema.
* Submissão: Os autores submetem os seus artigos, incluindo dados como título, resumo, palavras-chave e o ficheiro.
* Confirmação de Receção: Notificação de que a submissão foi recebida.

Distribuição para Revisão

* Convite para Revisão: Revisores são convidados para o processo de revisão.
* *Bidding*: Os revisores indicam as suas preferências sobre quais as submissões que pretendem fazer revisão.
* Atribuição de Revisores: Os organizadores atribuem os artigos aos revisores baseando-se nas suas áreas de *expertise*.
* Distribuição de Artigos: As submissões são disponibilizadas aos revisores.

Processo de Revisão

* Leitura e Avaliação: Revisores leem e avaliam as submissões.
* Comentários e Sugestões: Revisores fornecem *feedback* detalhado e recomendam aceitação, revisão ou rejeição da submissão.
* Submissão de Revisões: Revisores submetem as suas revisões no sistema.

Decisão sobre Artigos

* Compilação de *Feedback*: Os organizadores compilam as revisões e tomam a decisão sobre as submissões baseados no *feedback* dos revisores.
* Notificação de Decisão: Os autores são notificados sobre a aceitação, necessidade de revisão ou rejeição das suas submissões.
* Revisão de Artigos: Os autores submetem versões revistas se necessário, seguindo as sugestões dos revisores.

Estas são as etapas típicas do fluxo que a maioria dos CMS seguem. Nos requisitos funcionais, iremos detalhar cada etapa específica do fluxo da UAL CONF.

## Exemplos de CMS no mercado

### EasyChair

O EasyChair é uma plataforma web de gestão de conferências, criada com o objetivo de simplificar o processo de revisão e organização de conferências, em diversas áreas, desde a computação até às ciências da saúde.   
Pode ser categorizado como uma solução de "Third-party Organizers" (Organizadores externos) de acordo com a descrição feita acima.

O EasyChair permite lidar com todas as etapas do processo de gestão da conferência, desde a submissão de artigos até ao agendamento e pagamento da mesma.   
Embora os responsáveis pela conferência tenham o controlo sobre a qualidade dos dados introduzidos e tomada de decisões, a gestão da mesma é feita através da plataforma EasyChair.   
Isso permite que a organização se concentre no conteúdo da conferência, e o EasyChair nos aspetos logísticos e administrativos.   
  
 Principais funcionalidades e carateristicas do EasyChair:

Criação da Conferência, e submissão de documentos necessários;

Revisão: facilita o processo de revisão, permitindo que os interessados avaliem os artigos submetidos e forneçam comentários aos autores;

Gestão de revisores: gestão e acompanhamento do Comitê e dos conflitos de interesse;

Sistema de Notificação: atualização dos participantes, através de notificações automáticas sobre datas importantes, prazos de submissão e notícias relacionadas à conferência.

Personalização: oferece opções de personalização para a configuração do processo de revisão, os critérios de avaliação e outros aspetos da conferência de acordo com suas necessidades específicas;

Relatórios e Estatísticas: fornece relatórios detalhados e estatísticas sobre o processo de revisão, incluindo taxas de aceitação, médias de revisão e outras métricas relevantes;

Acesso Seguro: Garante um ambiente seguro e protegido para todas as atividades relacionadas à conferência, incluindo a submissão de artigos e o acesso aos conteúdos confidenciais.

O EasyChair é uma ferramenta abrangente para a gestão de conferências, permitindo a simplificação de tarefas complexas e a melhoria da experiência dos organizadores e dos participantes.

# Capítulo 2 – Especificidades técnicas

## 2.1 Requisitos funcionais

Os requisitos funcionais da UAL Conf baseiam-se num conjunto de funcionalidades presentes em *Conference Management Systems* atuais. Para uma melhor organização e implementação, dividimos os requisitos por fases.

### 2.1.1 Fase 1 - Inicialização

Numa primeira fase, a organização ou individuo responsável, terá de se registar na plataforma e depois criar e definir os detalhes iniciais da conferência, tais como:

* Nome (Obrigatório),
* Tipo de conferência (Obrigatório):
  + Conferência,
  + Congresso,
  + *Workshop,*
  + Painel,
  + Simpósio,
  + Seminário,
* Área de Especialidade (Obrigatório),
* País (Obrigatório),
* Cidade (Obrigatório),
* Descrição (Obrigatório),
* Página *web* referente à conferência (caso exista)(campo não obrigatório),
* Data de início e fim da fase de submissões (Obrigatório),
* Data de início e fim da fase de *bidding* (Obrigatório),
* Data de início e fim da fase de *reviews* (Obrigatório),
* Data de início e fim da conferência (fisicamente) (Obrigatório.

Na definição da conferência existem certos campos que tem de ser validados antes de ser criada (maioritariamente as validações envolvem as datas definidas):

* A data de início de submissões tem de ter uma diferença mínima de 3 dias da data atual, para que nesse período possa ser avaliada e aceite, ou não, pelos *admins*,
* As datas de Inicio de uma fase não podem ser superiores à data de fim,
* Cada fase deve seguir uma sequência específica (Submissões -> *Bidding* -> *Reviews* -> Conferência), ou seja, as datas não podem ser entrelaçadas entre fases,
* Todo o campo obrigatório tem de ser devidamente preenchido,
* O número mínimo de *reviewers* tem de ser sempre superior ao número máximo de *reviewers.*

Após a correta validação do *form* e aprovação, por parte de um *admin*, a conferência será então criada e, a partir deste momento, o responsável poderá entrar na página da mesma e começar a sua gestão (envio de convites para o comité, alteração de definições …). É também a partir deste momento que as próximas fases são desbloqueadas.

### 2.1.2 Fase 2 - Submissão

Nesta fase, que tem começo na data de início de submissões, os utilizadores registados, poderão ver a conferência na página *Call for Papers* (com o intuito de atrair potenciais autores), dando assim indicação que a mesma já está a aceitar submissões.

Na criação de uma submissão, é necessário identificar o(s) autor(es), título, *abstract* e o ficheiro inicial.

O autor tem, também, acesso a uma página (“*My Submissions”)*, onde poderá ver os detalhes da submissão, fazer download do ficheiro, editar a submissão (alterar autores, título, *abstract* e o ficheiro associado) e apagar a submissão.

O autor não está limitado no número de submissões que poderá fazer.

O comité, terá ainda a opção de aceder a página (“*All Submissions*”), onde é possível ver todas as submissões submetidas até ao momento, ver os detalhes de cada uma, fazer download do ficheiro e, caso necessário, apagar a mesma.

Os detalhes de cada submissão são guardados numa tabela do *Neon Tech*, enquanto o ficheiro é armazenado num *object storage* do *Supabase*.

## 

### 2.1.3 Fase 3 – *Bidding*

Após o término do período de submissões, o comité da conferência terá de distribuir os trabalhos entre os seus membros para uma revisão adequada. Este processo, conhecido como "*bidding*", desempenha um papel fundamental na garantia da equidade e eficácia do processo de revisão.

O *bidding* é uma prática onde os membros do comité expressam as suas preferências em relação aos trabalhos que desejam avaliar e o seu nível de conforto em relação ao tema do mesmo.

O processo de bidding começa com a disponibilização de uma lista de todos os trabalhos submetidos, onde será possível fazer o *download* da mesma, caso o *reviewer* ainda não tenha tido oportunidade de a ler. É também nesta tabela que o utilizador assinala as submissões às quais quer fazer *bidding* e indicar o seu nível de confiança, dentro do tema.

Com base nestas informações, os membros do comité podem fazer escolhas informadas sobre os trabalhos que desejam avaliar.

O *reviewer* terá também uma página (“*My Biddings*”) onde é possível alterar a *bid* (nível de confiança) e apagar a mesma, caso já não queira fazer *review* dessa submissão.

Após o início do período de *bidding*, o algoritmo que determina conflitos de interesse será executado de forma automática, de modo, a declarar logo de início os conflitos de interesse entre os membros do comité e as submissões presentes na conferência.

Durante o decorrer da fase, é possível, por parte do comité correr o algoritmo (na página de definições da conferencia), pois é possível haver adições de membros do comité durante o bidding.

O algoritmo irá verificar se um ou mais membros do comité faz parte dos autores (através do email) e, se necessário, se têm a mesma afiliação que 1 ou mais dos autores. Se um dos casos se comprovar então é declarado o conflito de interesses. Existe ainda a possibilidade de já terem sido feitos *bidding*, onde existem conflitos de interesses, nesses casos o algoritmo irá eliminar a *bidding* correspondente.

O(s) *chair(s)* e o *owner* da conferência terão ainda a possibilidade de fazer a declaração de conflitos manualmente, caso, aja conflitos que o algoritmo não consiga detetar (por exemplo, um autor e um membro do comité serem familiares). Tal como o algoritmo, se o conflito se aplicar a *biddings* já feitas, então, estas serão eliminadas.

Durante a fase de *bidding*, tanto o(s) *chair(s)*, como o *owner*, terão a possibilidade de, na página das definições da conferência, executar um algoritmo para atribuição de submissões a *reviewers*.  
Esta primeira atribuição ficará registada na base de dados, no entanto, poderá sofrer alterações, se o algoritmo voltar a ser executado manualmente ou automaticamente (no dia após a data final de *bidding)*.

(Explicar o algoritmo de atribuição quando tiver feito)

O(s) *chair(s)* e o *owner* terão ainda, uma página (“*Nome da Página*”) para fazer a atribuição manualmente. Caso contrário ao algoritmo, estas atribuições serão consideradas finais, ou seja, quando o algoritmo é executado, não terá estas submissões em conta. A atribuição manual poderá ser alterada, ou até mesmo apagada, até ao dia final de *bidding*.

### 2.1.4 Fase 4 – Revisões

(Não mudei pois temos de discutir como vamos fazer com as revisões)

Durante o período de revisões, o comité desempenha um papel crucial na avaliação da qualidade e relevância de cada trabalho.

Quando o período de revisões tem início, cada membro do comité recebe as submissões que lhe foram atribuídas. Cada avaliador deve atribuir uma nota à submissão com base nos critérios estabelecidos, refletindo assim a sua avaliação da qualidade e contribuição do trabalho. Além da nota, os avaliadores são também solicitados a indicar o seu nível de concordância com o conteúdo da submissão. Esta informação fornece uma perspetiva adicional sobre a sua opinião em relação ao trabalho e pode ajudar a identificar áreas de consenso ou discordância entre os revisores.

Ao completar o processo de revisão, os avaliadores submetem as suas avaliações através do sistema de gestão da conferência. Este permite aos organizadores reunir e analisar as avaliações de forma eficiente, garantindo uma revisão completa e equitativa de todas as submissões.

### 2.1.4 Fase 5 – Pré-Conferência

Após o término da fase de revisões, é disponibilizada à organização uma lista das submissões que foram avaliadas pelos membros do comité. Esta informação é crucial para ajudar os organizadores que têm a responsabilidade de escolher quais as submissões que irão avançar para serem apresentadas, posteriormente, na conferência. Podem definir critérios específicos para esta seleção, como o número de submissões a aceitar, seja por um valor absoluto ou uma percentagem do total. Tanto o número definido como a pontuação das submissões são disponibilizados aos organizadores como guia para auxiliar na seleção, embora não sejam obrigados a seguir estritamente esses valores.

Os autores cujas submissões forem aceites são notificados e informados das avaliações recebidas. Têm então a oportunidade de submeter a versão final dos seus trabalhos para a conferência.

Este processo de preparação para a conferência garante que apenas trabalhos de qualidade e relevância adequadas sejam incluídos no programa, enquanto oferece aos autores a oportunidade de finalizarem os seus trabalhos para apresentação durante o evento.

## 2.2 Requisitos não funcionais

O sucesso da UAL Conf dependerá não só da sua funcionalidade, mas também da sua capacidade de se adaptar e responder às diversas necessidades das conferências que serve. Assim, os requisitos não funcionais desempenham um papel crucial na garantia da eficácia e flexibilidade deste sistema de gestão de conferências. Com base nas necessidades identificadas, os seguintes requisitos não funcionais foram estabelecidos:

### 2.2.1 - Configurabilidade

A UAL Conf deve ser altamente configurável, permitindo ajustes e adaptações para atender às especificidades de cada conferência e das suas diferentes fases. Esta capacidade de configuração permite uma personalização completa do sistema para atender às necessidades específicas de cada organizador e evento.

### 2.2.2 – Modularidade (Not sure about this one)

O sistema deve ser modular, com funcionalidades separadas em módulos distintos, cada um com responsabilidades próprias. Esta abordagem simplifica o sistema e facilita atualizações ou reimplementações de módulos individuais.

### 2.2.3 - Extensibilidade

A UAL Conf deve ser extensível, permitindo a incorporação de novas funcionalidades na forma de novos módulos. Se o sistema não possuir uma funcionalidade necessária para uma conferência, um administrador deve ser capaz de desenvolvê-la e adicioná-la à sua instalação com pouco ou nenhum impacto nas outras funcionalidades. Além disso, o sistema deve ser extensível de forma administrativa, permitindo a aplicação de novas regras ou políticas.

### 2.2.4 - Compatibilidade

Dada a natureza de um CMS (Sistema de Gestão de Conteúdos), a UAL Conf deve ser facilmente acessível a partir de diversas plataformas e dispositivos, independentemente do tamanho da tela ou do sistema operacional utilizado pelo utilizador.

### 2.2.5 - Confiabilidade

O sistema deve ser altamente confiável, minimizando o tempo de inatividade e prevenindo a perda de dados a todo o custo. Qualquer interrupção ou perda de dados pode ter impactos críticos no funcionamento da UAL Conf e na experiência dos utilizadores.

### 2.2.6 - Segurança

Devido ao acesso de diferentes utilizadores com diferentes níveis de autorização, o sistema deve garantir uma segurança robusta, com autenticação e autorização adequadas para cada tipo de utilizador. Esta medida visa proteger os dados sensíveis e garantir a integridade do sistema.

### 2.2.7 - Usabilidade

O sistema deve ser intuitivo e fácil de usar, não requerendo formação prévia por parte dos utilizadores. Este requisito é especialmente relevante para os autores, que podem usar o sistema apenas uma ou duas vezes e devem ser capazes de submeter os seus trabalhos de forma simples e direta.

## As nossas páginas por cada etapa do processo

### Fase de Inicialização

|  |  |
| --- | --- |
| Nome da página | O que faz? |
| Criar conta | Página dedicada para o utilizador ter a possibilidade de criar uma conta no nosso *website.* |
| Entrar | Nesta página, o utilizador pode fazer o *login* com a sua conta |
| O meu perfil | Utilizador tem a capacidade de alterar os seus dados pessoais bem como alterar a sua *password* ou ainda de submeter um código de convite para ser *reviewer* de uma conferência. |
| Criar uma conferência | Nesta página, o utilizador pode fazer um pedido para criar uma conferência. Após preencher todos os campos necessários, o mesmo terá de aguardar no máximo de dois dias até que os administradores da página aprovem ou rejeitem a conferência criada. Desta forma é possível fazermos filtragem entre o que é ou não uma verdadeira conferência.  Após a tomada de decisão sobre a mesma, o utilizador será informado através de um envio de um email |
| Conferências pendentes | Esta página só aparece se e só se o utilizador em questão for administrador da página. Nesta página é possível visualizar os detalhes das conferências acabadas de serem criadas e também nesta onde se pode aceitar ou rejeitar uma conferência |
| As minhas conferências | Nesta página o utilizador poderá aceder a uma tabela onde está disponível todas as conferências onde ele participa independentemente do seu papel na respetiva conferência.  É também nesta página onde o utilizador poderá entrar em cada uma das conferências para entrar no modo de edição da conferência (explicaremos mais adiante o que faz este modo) |
| Gestão do comité | Página onde é possível encontrar todos os membros do comité. É possível alterar o papel destes, ver alguns detalhes dos mesmos e ainda remover membros para os utilizadores com o papel de *owner* ou *chair*. |
| Definições da conferência | Nesta página é possível alterar alguns dados da conferência, tal como as datas registadas no momento da criação da mesma, ou alguns dados informativos tais como cidade da conferência, página web entre outros.Nesta página é também possível “disparar” algumas ações como correr os conflitos de interesse ou o algoritmo de *bidding*. |
| Detalhes da conferência | Esta página é a primeira página a ser aberta após um utilizador entrar numa conferência a partir da página das minhas conferências. Esta página possui todos os dados informativos sobre a conferência em questão, tais como em que percentagem se encontra, em que estado se encontra e ainda quais os seus dados registos no momento da criação. |
| Composição de emails | Página destinada a enviar emails por parte do comité para os diferentes papéis de utilizadores. Pode ser utilizada sempre que for necessário fazer algum comunicado. |
| Envio de convites | Esta página serve para convidar novos membros para o comité onde depois do convite ser realizado é possível indicar se os mesmos deverão ser eliminados ou não e ainda em que estado é que se encontram |
| Eventos | Esta página destina-se a ser meramente informativa, logo não possui a criação de qualquer ação para além a de consulta. Em determinados momentos no ciclo de vida de uma conferência, existem algumas ações que são registadas para questão de *logs*, como por exemplo, a criação da conferência, uma submissão criada ou até mesmo a alteração na estrutura do comité, sendo então possível consultar nesta página esses mesmos eventos onde são registados a data e hora e quem os realizou |

### Fase de Submissão

Todas as seguintes páginas encontram-se funcionais enquanto a o período de submissão estiver ativo.

|  |  |
| --- | --- |
| Nome da página | O que faz? |
| Criar uma submissão | É possível criar uma submissão para uma conferência em questão. Nesta página o utilizador deverá indicar os diversos autores, um título e um resumo do que se trata a sua submissão bem como um único ficheiro que represente o eu trabalho submetido. |
| As minhas submissões | Após ter sido efetuado pelo menos uma submissão, o autor da mesma poderá consultar as suas submissões. Os autores poderão também fazer o download do seu ficheiro de forma a ser possível consultar o mesmo; pode editar a sua submissão onde é-lhe possível adicionar/remover autores, alterar título ou o resumo e/ou alterar o seu ficheiro inicial por um outro. É também dada a possibilidade de visualizar alguns detalhes da submissão e ainda apagar por completo a mesma. |
| Todas as submissões | Esta opção apenas se encontra disponível para os membros do comité, *chair* ou *owner* e é lhes possível consultar todas as submissões realizadas incluindo a visualização de alguns detalhes e/ou o download do ficheiro submetido nas respetivas submissões. |

## Papéis de utilizador permitidos

No UalConf, consoante o tipo de interação que um utilizador tenha com uma conferência, poderá apresentar tipos de papéis a desempenhar e mais do que isto, pode também apresentar múltiplos papéis, por exemplo, um utilizador que faça uma submissão a uma “call for papers” de uma conferência passa a ser um autor para essa conferência, mas se este mesmo autor já fazia parte do comité, então não só tem o papel de autor como também tem o papel de pertencer ao comité.

Uma das formas de um utilizador ter acesso a que papel pertence é entrar no “My Conferences” e verificar o que indica a coluna “Your Role” tal como se pode observar na seguinte imagem:

A screenshot of a phone

Description automatically generated

Nesta imagem podemos também constatar um exemplo de um utilizador que apresenta mais do que um papel e nestes casos aparece no formato de texto: “papel 1, papel 2, …, papel n”.

Posto isto os papéis que disponibilizamos são:

### Owner

Este tipo de papel, é atribuído a um utilizador assim que este cria uma conferência.

Apenas pode existir um Owner por conferência e este tem os mesmos privilégios que um Chair, o que significa que se um utilizador tiver este papel não irá ter também a indicação de ser um chair.

Uma outra característica deste papel é que nunca pode ser removido de uma conferência enquanto os chairs e os committee podem, princisamente porque este é o responsável principal da mesma.

### Chair

Este papel é um papel que um utilizador “ganha” quando é convidado para entrar para o comité da conferência e assim que este aceita o convite entra com este papel. Esta opção é possível ser feita dentro do menu da conferência em “Emails” e de seguida “Send Invitation” onde depois poderá selecionar-se o papel de Chair e indicar os emails para o convite, tal como mostra a seguinte imagem:

A screenshot of a computer

Description automatically generated

Uma outra forma de um utilizador passar a ser um Chair é através de uma atualização direta ao seu papel na opção da gestão do comité tal como ilustramos na seguinte imagem:

A screenshot of a computer

Description automatically generated

Esta atualização só poderá ser feita por outros Chairs.

Committee

Da mesma forma que um utilizador pode passar a ser um chair, o mesmo coloca-se para este papel de Committee sendo que a única diferença é que este papel tem menos permissões que um owner e um chair, mais concretamente, este não tem a autoridade de aceder á administração da conferência, não podendo alterar papeis de outros utilizadores, alterar definições da conferência e ainda entrar na tomada de decisão dos papers que irão entrar numa determinada conferência, no entanto terão a capacidade de ver todas as submissões, entrar no processo de bidding e ainda fazer as reviews.

### Author

O autor, é o último papel que disponibilizamos aos nossos utilizadores e este é atribuído a partir do momento que um determinado utilizador faz uma submissão numa conferência.

Este papel é o que tem menos permissões em comparação com os restantes, onde atem acesso aos detalhes da conferência; criar uma submissão; consultar e/ou editar as suas submissões passadas.

# Capítulo 3 – Arquitetura e características da aplicação

## Arquitetura da aplicação

Uma imagem com texto, diagrama, captura de ecrã, Paralelo

Descrição gerada automaticamente

A nossa aplicação foi desenhada com o modelo de sistemas distribuídos com uma arquitetura de 3 camadas (ver seguinte imagem) onde temos a camada de apresentação que se traduz na nossa camada responsável pelo *front-end*, ou seja, a camada que interage com o cliente; a camada de negócio, onde temos a camada que processa os pedidos enviados pela camada anterior e é nesta onde temos a nossa API e por último a camada de dados que se traduz na nossa base de dados.

Como característica destes tipos de modelos, o fluxo de dados é sempre o mesmo e não há desvios, ou seja, se o cliente precisar de obter dados da base de dados, não entra em contacto diretamente, mas antes sim faz um *request* à API para que esta vá então encontrar os dados pretendidos por este. Desta forma conseguimos garantir segurança dos nossos dados, uma vez que um utilizador não tem acesso direto aos mesmos e quando acede aos dados é por um meio totalmente controlado (a nossa camada de negócios)

## Tecnologias utilizadas no Front-End

O nosso *front-end* foi desenhado através de três principais ferramentas: React, MUI e Javascript.

React, de acordo com a própria entidade que desenvolveu, é uma biblioteca de Javascript para criar *interfaces* para os utilizadores, onde esta considera-se ser:

* Declarativa: Capacidade de ser fácil de trabalhar e intuitiva;
* Baseada em componentes: O que significa que em react, quase tudo pode ser considerado um componente e que estes podem se chamar uns aos outros criando assim um paradigma de modularidade;
* Updates e renders: Garantem que atualizam os elementos que queremos sempre que determinados dados são alterados;
* Utiliza JSX (Baseado em xml) para indicar o que será renderizado.

Apesar de todas estas características e vantagens do React, ainda existe um problema. Todos os componentes, até agora, serão desenhados em JSX e CSS o que poderá ser bastante trabalhoso e demorado, como tal, algo que se costuma fazer é utilizar bibliotecas de componente altamente dedicadas para facilitar a implementação de elementos visuais ou a customização dos mesmos. Posto isto, para o nosso projeto decidimos utilizar o MUI (Material UI) que é uma biblioteca de *open-source* que implementa material de design da Google. Podemos constatar que a utilização desta biblioteca, não é a das mais fáceis de utilizar para novos utilizadores, mas após algum de tempo de prática a curva de aprendizagem tende a atenuar.

## Modularidade na Aplicação

Toda a nossa aplicação foi pensada e desenvolvida com a ideia de “Modularidade”. Pode-se constar na nossa arquitetura, mas também na forma como o nosso código foi desenvolvido e estruturado. No caso do *front-end* e também devido à forma como o React funciona, todas as nossas páginas são construídas com a reutilização de componentes já existentes, o que permite poupar linhas de código tornando mais fácil de leitura, mais fácil de compreensão e também mais fácil para fazermos *debug* quando necessário. Os principais exemplos que temos de reutilização desses componentes, temos por exemplo o nosso *footer,* os cabeçalhos de cada página, os *layouts* ou até mesmo as nossas tabelas (ver imagem) onde esta representação:

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generatedÉ possível chamando simplesmente o seguinte código:  
  
Desta forma, temos também uma enorme vantagem na facilidade e rapidez de desenvolvimento.

Portanto, sempre que nos foi possível, para as situações em que o código teria de ser utilizado mais do que uma vez, o mesmo seria colocado num só componente sendo depois somente necessário esse mesmo componente. Por vezes aconteceu certas situações onde tornou-se desafiante que um único componente conseguisse ser adaptado em todas as situações, uma vez que por vezes a forma como o componente estava desenhado para uma página poderia não fazer tanto sentido para uma outra página e nesses casos, sempre que possível, adicionávamos parâmetros no componente para definir um comportamento adaptável a diversas situações e assim acabávamos por ter um componente muito mais genérico e facilmente adaptável a todas as situações. O melhor exemplo que temos para este caso foi um componente chamado “PopUpWithMessage”. Este componente, tal como o nome sugere, é um componente que despoleta um *pop-up* com uma determinada mensagem e um ou mais botões:

A close-up of a message

Description automatically generated  
  
  
A white background with black text

Description automatically generated

Observando as imagens acima, reparamos que o mesmo componente tem em semelhança um título e uma mensagem, no entanto, os botões diferem, onde na primeira situação temos dois botões para uma situação que seja necessária uma ação de confirmar ou cancelar e na outra imagem uma situação onde apenas existe a ação de confirmação.  
Para nós, esta diferença de comportamentos é possível de ser obtida de acordo com os parâmetros escolhidos na chamada dos componentes. Vamos mostrar a diferença do código em contraste com o resultado da chamada do mesmo:  
  
A computer screen shot of a message

Description automatically generated

A white background with black text

Description automatically generated

A screenshot of a computer

Description automatically generated

A close-up of a message

Description automatically generated

Tal como é possível de se observar, apenas com a alteração do parâmetro “justOneButton = {true}” conseguimos modificar por completo este comportamento adaptando o mesmo componente a diversas situações. O código acima que não tem o parâmetro declarado, não o tem porque este mesmo tem um *defaul*t a *false.*

E desta forma é nos então possível de generalizar e reutilizar ainda mais o código existente.

Ainda no caso do “React”, temos também a modularidade presente na medida que qualquer pedaço de código existente no nosso projeto é um componente e como tal é igualmente um módulo o que nos facilita bastante não só na separação/organização do projeto de acordo com as ações de cada componente, mas também em situações de *debug* como já foi mencionado.

A nossa modularidade existe também no nosso *back-end*, mais concretamente no “Node.js”, onde neste caso temos também toda a nossa lógica separada por diversos ficheiros de *javascript*, onde cada um é responsável por determinadas ações. Podemos também dizer, tal como já foi mencionado no contexto do *front-end*, com esta estratégia de desenvolvimento é igualmente possível reutilizar código tal como fizemos com o React. Conseguimos observar isto em diversas situações, tal como por exemplo, nas situações que um endpoint da API é chamado, muita das vezes temos de chamar o auth.ensureAuthenticated (ver seguinte imagem), para verificar se o utilizador que está a realizar o *request* está efetivamente ou não autenticado. No fundo o que estamos a fazer é chamar uma função existente sem ter que a construir novamente no *scope* deste *endpoint* para que este tenha acesso à mesma:



## UI e UX

Um dos principais desafios inicialmente sentidos com este projeto, foi a questão do front-end, mais concretamente, o UX (*user experience*) e o UI (*User interface*).

Acontece que o nosso grupo é composto por elementos onde nos sentimos todos mais à vontade em trabalhar no campo da lógica do código, isto é, no *backend*, do que propriamente no *front-end*, então para colmatar esta nossa “fraqueza”, achámos que seria uma mais-valia e também uma forma de “ganhar” algum tempo de desenvolvimento encontrar um projeto em react que nos fornecesse uma base coerente e sustentável e que ajudasse a garantir a responsividade, UX e o UI do nosso projeto.

Após alguma investigação, conseguimos encontrar um *template* que nos proporcionou o que precisávamos.

Foi nos possível encontrar uma empresa chamada “CREATIVE TIM” que disponibiliza para todos uma solução *open-source* já com componentes construídos e próprios do *template* que garantem muito do trabalho técnico de design e de responsividade.

Originalmente este *template* servia para gerir uma página onde seria suposto apresentar e utilizar *dashboards* e outros elementos estatísticos que não se enquadravam no âmbito do nosso projeto, como tal acabou por haver um período do nosso tempo onde foi imperativo a adaptação deste template à nossa realidade onde, e mais concretamente, fizemos:

* Organização dos componentes em diretorias com uma estrutura lógica ao projeto;
* Apagar componentes que não seriam necessário;
* Todos os componentes que queríamos manter, mas que precisavam de algum ajuste, duplicámos o código para uma diretoria nossa e customizámos esse novo componente. Decidimos aplicar esta estratégia de forma a querermos garantir não só que mantínhamos o código original caso necessário como também por vezes tornou-se necessário adaptarmos o código original para certas ocasiões e o customizado para outras.

As seguintes imagens mostram a comparação da página inicial entre o tema na sua forma original e como ficou adaptado:

A screenshot of a computer

Description automatically generated

A group of people sitting in a room

Description automatically generated

É de notar que existem semelhanças no que toca ao aspeto e ao *design* da página e dos elementos, da mesma forma é igualmente possível encontrar diferenças.

### Compreensão dos elementos visuais

Mesmo com a responsividade garantida pelo nosso template, todos os novos componentes que tivemos de criar bem como todas as nossas páginas; foi importante garantir que continuávamos a manter uma boa relação entre o nosso UI e UX. Isto foi possível ser obtido através das sucessivas questões que nos íamos colocando durante o desenvolvimento:

* Esta página está a ficar com informação em demasia?
* A cor dos botões está coerente/transmite a ideia correta para as respetivas ações?
* Se redimensionar a página, os elementos de UI continuam a serem funcionais e apelativos?
* O Layout no geral é apelativo e funcional?
* Faz sentido esta tabela ser paginada ou devo deixar mostrar todas as linhas?
* O tamanho dos elementos faz sentido? Deveria estar mais pequeno? Maior?

Mesmo com esta nossa “receita caseira” nem sempre foi fácil garantir a resposta certas a estas questões. A nossa equipa não é especialista em elementos visuais, como já foi mencionado anteriormente, o que significa que reconhecemos que certas avaliações dos componentes visuais por vezes tornavam-se pouco técnicas e mais subjetivas, no entanto sempre que possível foi se sempre tentando manter uma coesão visual que garantisse ao máximo estas questões levantadas.

A nossa “receita” passava então pelos seguintes elementos visuais:

### Utilização de *Cards*

A screenshot of a computer

Description automatically generatedOs Cards são componentes existentes no próprio MUI, e funcionam como containers flexíveis e expansíveis com capacidade de possuir diversos conteúdos tal como se pode observar na seguinte imagem:

No nosso projeto, mais concretamente, a maioria das nossas páginas é constituída por diversos cards onde cada um apresenta uma parte da lógica da própria página onde por norma o primeiro card serve para identificar a própria página, através da apresentação da mesma e ainda apresenta uma breve explicação caso se se trate de uma página mais complexa a nível de funcionalidades acabando por permitir criar a ideia de “organização visual”.

Todos a restante lógica da página e/ou funcionalidades são colocadas noutros cards onde o número destes é diretamente proporcional à complexidade e lógica existente.

#### As cores

Todo o nosso website, centra-se nas cores: branco, cinzento-claro, cinzento-escuro e azul-escuro e é possível encontrar esta conjunção em diversos elementos. Por normal o azul-escuro encontra-se mais nos nossos textos, o branco nos *Cards*, o cinzento-escuro na nossa barra de menus principal (situada à esquerda) e o cinzento-claro no fundo das nossas páginas.

Algumas destas cores já vinham com o próprio tema, e face a termos gostado da conjunção, fomos mantendo ao longo do desenvolvimento principalmente porque estas em conjunto apresentam cores neutras e não “berrantes” que acabam por cansar e desgastar a vista do utilizador, sendo possível atingir um cero nível de conforto e tranquilidade e ao mesmo tempo o certo nível de profissionalismo. Afinal de contas o nosso website não é algo que será utilizado por um largo público-alvo, mas antes sim por um nicho.

Ao longo do desenvolvimento fomos também começando a incluir novas cores, principalmente nos botões. Acreditamos que o uso de cores específicas pode significativamente melhorar a experiência do utilizador e tornar a navegação mais intuitiva. No nosso caso, utilizámos as seguintes cores para diferentes ações:

* + Vermelho: Usado para ações de apagar, pois essa cor chama a atenção e remete a perigo e cautela.
  + Verde: Associado a ações positivas. Utilizamos verde para elementos que não envolvem perigo, como edições e submissões, indicando segurança e permissão.
  + Amarelo: Utilizado para destacar elementos aos olhos do utilizador, atraindo a atenção rapidamente.
  + Azul: Utilizado para indicar ações mais neutras. Esta escolha é coerente com as cores principais do nosso site, que são cinzentos, azul e branco.

Esta paleta de cores foi cuidadosamente selecionada para garantir uma interface de utilizador clara e eficiente, melhorando a usabilidade e a experiência geral do utilizador.

A screenshot of a social media post

Description automatically generated

#### Barra de navegação principal

A screenshot of a phone

Description automatically generated A nossa barra de navegação principal é o único elemento com a cor cinzento-escuro, e consideramos que esta escolha é uma mais-valia. A cor mais o contraste, naturalmente atraem a atenção dos utilizadores, que é precisamente o nosso objetivo, porquê?  
  
É neste onde todas as principais opções do nosso site se irão encontrar e esta tem de ser visível ao utilizador porque sem esta o mesmo não tem como navegar estre as nossas diversas opções e funcionalidades.  
  
Tal como é mencionado pela Interaction Design Foundation  
  
…” It’s the ‘steering wheel’ of any site and all the pages therein; it’s also the grand plan, showing what’s all around—with a built-in ‘compass’ to keep users informed as to where they are at any time. A site’s global navigation acts as its central nervous system—at least, to the user’s eye” …  
  
O que, traduzindo fica: “É o “volante” de qualquer site e de todas as páginas nele contidas; também é o grande plano, mostrando tudo ao redor - com uma 'bússola' embutida para manter os utilizadores informados sobre onde estão a qualquer momento. A navegação global de um site atua como seu sistema nervoso central - pelo menos, aos olhos do utilizador.”

#### Indicadores de estado

O nosso website apresenta no total três indicadores de estado:

* Do ponto de vista do negócio: Sempre que um utilizador entra numa conferência é apresentado dois indicadores, um para indicar o quão avançado a conferência se encontra e um outro é em que estado é que esta se encontra.

Achamos que estes indicadores podem ser extremamente úteis uma vez que dão uma visão genérica de como se encontra cada uma das conferências e o facto de estes indicadores serem a primeira coisa que um utilizador vê assim que entra numa conferência pode ser extremamente vantajoso uma vez que é um dado importante e como tal deve ser visto assim que possível para efeitos de monitorização.



* Do ponto de vista técnico:

Sempre que as nossas páginas têm de fazer uma chamada à nossa API, para garantir que o utilizador tem algum feedback sobre se uma determinada operação de escrita/leitura está ou não concluída, temos um indicador visual. Assim que o pedido à API é enviado, nós instanciamos este indicado, após termos a indicação que temos resposta da API o indicador termina.

A white square with a blue stripe

Description automatically generated

#### Barra de navegação da conferência (secundária)

Para além da nossa principal barra de navegação, temos também uma segunda barra de navegação que pode ser encontrada assim que um utilizador entra numa conferência quer pela opção das minhas conferências quer pela opção do *“Call For Papers”*.

A nossa aplicação, consideramos que está composta em dois blocos principais. O primeiro bloco onde está contida toda a aplicação é esta que se vê assim que um utilizador faz login e depois o segundo bloco que surge assim que entramos numa conferência uma vez que esta ação despoleta novas páginas e comportamentos que não eram acessíveis até então.

Como se pode observar na seguinte imagem, nesta imagem a nossa barra de navegação tem as letras de um tom diferente para chamar à atenção e um corpo bastante semelhante a um *“Card”*. Achamos que este design permite, não só chamar à atenção do utilizador como também não cansar muito a vista do utilizador uma vez que já temos a nossa barra principal no mesmo plano, tal como se pode observar na segunda imagem

A screenshot of a web page

Description automatically generated

A screenshot of a computer

Description automatically generated

#### Tabelas

As tabelas, são um dos nossos componentes mais reutilizados no nosso website. Foram inicialmente importados do MUI e foram sendo adaptados às nossas necessidades, onde acabámos por ter personalizar alturas, larguras, efeitos nos cliques entre outros.

A nossa tabela tem alguns comportamentos bastantes próprios e importantes para o bom funcionamento do nosso website tais como: Ordenação dos dados, filtrar dados e ainda esconder/mostrar colunas.

Na seguinte imagem temos um exemplo de uma tabela nossa. É possível constatar que não só conseguimos adicionar texto como também diversos botões com comportamentos diferentes, botões estes que permitem corresponder às necessidades da lógica do negócio:

A screenshot of a computer

Description automatically generated

A seguinte imagem mostra um exemplo do menu que as tabelas fornecem assim que se clica nos três pontos da coluna:

A screenshot of a computer

Description automatically generated

Assim que clicamos em filtrar, temos três parâmetros, podemos aplicar um filtro sobre a coluna em questão ou mudar para uma outra. Temos também a opção de definir que tipo de operador queremos aplicar no nosso filtro (ver segunda imagem) e por último o valor a ser filtrado. Assim que o utilizador vai dando os seus inputs, os dados vão sendo automaticamente filtrados:

A screenshot of a computer

Description automatically generated

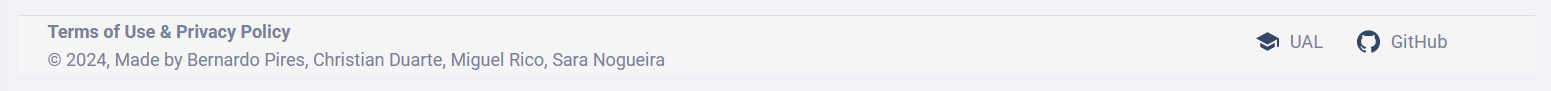
A screenshot of a computer

Description automatically generated

#### Footer

O nosso footer é composto por três 4 elementos principais, dois botões com hiperligações onde temos o primeiro “UAL” que após ser clicado é aberta a hiperligação para o site da nossa universidade e um segundo “GitHub” onde é aberto o nosso repositório.

De seguida temos os nossos termos de uso



## Navegação entre as páginas

Em desenvolvimento web, podemos navegar entre páginas de duas maneiras. Podemos navegar por rotas definidas, ou seja, a alteração de um URL permite ir para uma determinada página, ou podemos deixar essa navegação sob a responsabilidade do JavaScript, onde este pode, por exemplo, abrir um componente que representa uma nova página. Isso dá ao utilizador a impressão de que uma nova página foi aberta, enquanto o URL permanece exatamente o mesmo.

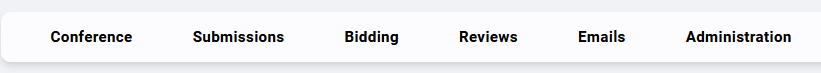
No nosso website, utilizámos ambas as estratégias, conforme necessário, sendo que a nossa principal estratégia de navegação adotada foi através da navegação por rotas, ou seja, pela alteração do URL, onde o utilizador clica em um botão específico e é direcionado para o URL correspondente, e como é que estamos a fazer isto? Assim que nossa aplicação é construída pelo JavaScript, dois componentes são chamados: 'routes.js' e 'conferenceRoutes.js'. Estes dois, são no fundo arrays de objetos, onde cada objeto representa uma opção dos menus de navegação, mas mais do que isso, cada objeto tem também especificado a sua própria rota.

A seguinte imagem permite mostrar o exemplo:

A screen shot of a computer program

Description automatically generated

O exemplo acima mostra as opções que definem o menu que existe dentro das conferências:



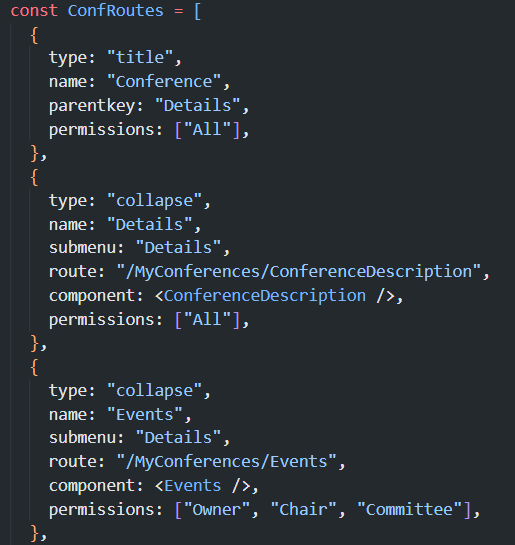
Mais concretamente:

A screen shot of a conference

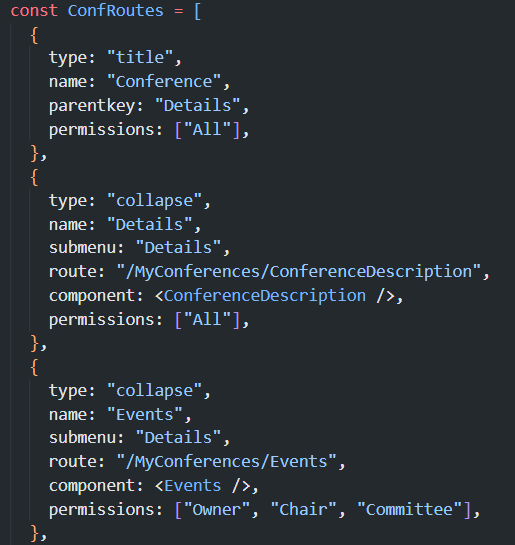
Description automatically generated

Onde podemos interpretar o seguinte:

Opção do menu primária



Opção do menu secundária (sub-menu)



Define se é um título ou uma opção clicável

Nome a aparecer

Referência da entidade pai para uma relação de pai-filho (menu – submenu)

Que tipos de papéis de utilizadores têm acesso a esta opção

Qual o pai desta opção (menu)

Qual a rota a ser carregada

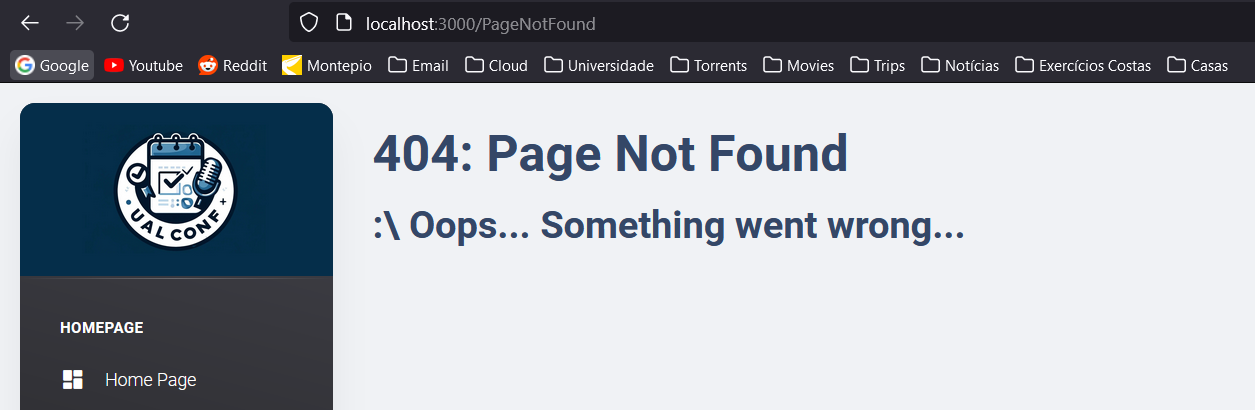
Qual o componente a ser carregado para memória associado a esta rota

Esses arrays são processados e interpretados nos componentes que constituem as nossas barras de navegação.

Quando o site está a ser construído, cada um destes arrays é processado e cada rota definida para cada objeto é utilizada e guardada nas seguintes linhas de código:A computer screen shot of a code

Description automatically generated

É possível constatar que existem duas rotas que são estáticas: a Home page, que é a página principal, e o que está a fazer no código é permitir que quando colocamos algo como <https://ualconf.vercel.app>, vá diretamente para o elemento da HomePage. O segundo elemento que temos é a Página não encontrada, que é chamada quando vamos para qualquer URL que não exista. O que permite que, se fizermos <https://ualconv.vercel.app/estaPaginaNaoExiste>, seja indicado o seguinte ao utilizador:



Esta, tal como dissemos, é a principal forma que estamos a utilizar para permitir a navegação entre as páginas do nosso website, mas também temos alguns casos onde aparente estarmos a mudar de página e o URL permanece o mesmo. Analisando o seguinte exemplo:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

De acordo com a nossa arquitetura e implementação, tudo o que são “opções principais” têm uma rota atribuída. Tudo o que for “ações secundárias”, ou seja, que existem dentro das primárias, então fazemos a gestão através do JavaScript, onde, por exemplo, com o clique num botão, um componente é renderizado, acabando por esconder o anterior. Este tipo de estratégia dá-se o nome de Single Page Application (SPA).

A forma mais comum que utilizamos para fazer esta gestão interna é através de eventos de cliques em botões que alteram um estado, e, com base na alteração desse estado, o JSX do React volta a ser renderizado e permuta entre o que o utilizador está a visualizar:

A screen shot of a computer program

Description automatically generatedNeste exemplo, quando o utilizador clica no botão de review da tabela, é disparado um evento de clique (onClick), onde uma das coisas que faz é dizer que o valor do openReview é true.

Esta alteração faz com que o JSX do React volte a renderizar os seus elementos e, assim que verifica se o openReview (que está a ser utilizado no próprio JSX) é true ou false, através de uma verificação ternária, é escolhido o elemento a ser apresentado, tal como se pode constatar com a seguinte imagem:

A computer code with text

Description automatically generated with medium confidence

## Estrtutura do projeto

Para separar a lógica que já estava construída e fornecida pelo próprio *template* da lógica adaptada e construída de raiz por nós, criámos a seguinte lógica na hierarquia do diretório do projeto:

Tal como se pode observar na seguinte imagem, tudo o que é indicado por “Our” e “pages” foi construído por nós, sendo que o resto é o que o próprio *template* já fornece.

Nas “pages”, temos todas as páginas do nosso site, sendo que, portanto, uma página é no fundo um componente que chama tantos outros componentes.

Nos “OurComponents” e nos “OurLayouts” são todos os componentes onde reutilizámos código do prório *template* ou componentes novos que criámos de raiz.

A screenshot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

## Contextos e gestão de comportamentos

A framework React permite-nos usar contextos. Estes contextos são uma memória estática que pode ser acedida por todos os componentes, garantindo que certos componentes tenham valores partilhados. Se compararmos com o paradigma de programação orientada a objetos, poderíamos fazer uma comparação algo grotesca dizendo que seria semelhante a uma classe com atributos e métodos estáticos para que fossem sempre acedidos por todos os que os chamassem, no entanto o React vai um pouco mais além de implementar somente esta ideologia, uma vez que, qualquer componente que esteja a utilizar uma variável de um contexto e assim que essa variável é alterada (independentemente do componente que a alterou), estes componentes vão ser novamente renderizados de forma a implementar a nova atualização.

No nosso projeto criámos dois contextos, um contexto de autenticação e um outro contexto para as conferências. O primeiro contexto é o que gere, tal como o nome sugere, tudo o que envolve com autenticações, mostra qual o ID, o email do utilizador; indica se o mesmo está ou não com o *log in* feito e ainda se é um *admin* da aplicação ou não. Este contexto é muitas das vezes utilizado em muitas das nossas páginas, assim que são abertas, têm de fazer requisições à nossa API, para garantir segurança, só permitimos que as mesmas são feitas se e só se o utilizador estiver com *log in* feito, ou por exemplo, para definir que menus são mostrados ao utilizador, é também com base nas variáveis do nosso contexto.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer program

Description automatically generated

O contexto da conferência, foi um contexto criado somente para trabalhar no *scope* das conferências. Isto é, assim que um utilizador faz *login* este contexto ainda não tem nada armazenado, no entanto quando o mesmo entra para uma área de uma conferência através das opções “Call For Papers” ou “My Conferences” o contexto da conferência passa automaticamente a ser utilizado. A ideia implementada neste contexto é muito idêntica ao contexto da autenticação, isto é, disponibilizar opções do menu consoante a “Role” desempenhada pelo utilizador e identificar nos pedidos à API qual é o ID da conferência atual onde o mesmo está atualmente a trabalhar.

Um dos maiores desafios que tivemos para implementar os contextos foi a persistência da memória dos mesmos, isto porque, os contextos em react, assim que a página é fechada e volta a ser aberta ou numa situação de um *refresh*, a mesma perde os valores das variáveis que outrora possuía. Para colmatar este obstáculo, sempre que estas situações acontecem, os contextos, fazem automaticamente uma chamada específica a um *endpoint* da API, que devolvem os valores supostos para as variáveis do contexto, sendo que no caso do contexto da autenticação, é através da cookie de autenticação, enquanto no caso do contexto da conferência, é através dos valores armazenados na base de dados.

Com os contextos é então possível concluir que é através destes que conseguimos implementar mecanismos de segurança, como por exemplo, fazer aparecer ou esconder opções, realizar ou não pedidos à nossa API, entre outros. É possível também concluir que com esta técnica é muito mais fácil de garantir a integridade dos dados em toda a aplicação uma vez que uma variável no contexto é acessível em qualquer parte do código.

## Segurança no lado do cliente

### Descrição dos nossos componentes e layouts

#### OurComponents

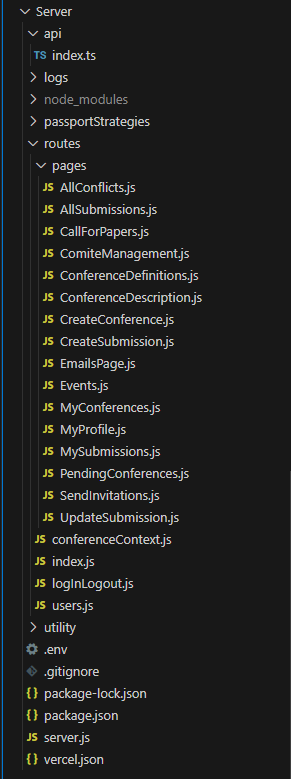
|  |  |
| --- | --- |
| Componente | Descrição |
| **ErrorHandling** |  |
| ErrorSiginSignup.js | Este componente é utilizado para exibir mensagens de erro específicas durante os processos de login ou registo. Quando ocorre um erro, o componente apresenta uma mensagem de erro ao utilizador e oferece um botão para redirecionar de volta à página inicial. |
| **Footer** |  |
| Footer.js | Este componente é responsável por renderizar o rodapé da aplicação. O rodapé inclui links importantes, como o link para o GitHub do projeto e os Termos de Uso e Política de Privacidade. |
| **Info** |  |
| AllSubmissionsDetails.js | Este componente apresenta detalhes sobre todas as submissões feitas para uma conferência específica. Inclui informações como título, autores e data de submissão. |
| ConferenceDetails.js | Este componente mostra informações detalhadas sobre uma conferência, incluindo a descrição, datas importantes (submissão, revisão, conferência), detalhes do organizador, website, contacto, localização, área da conferência, tipo de conferência e o número mínimo e máximo de reviewers. |
| MoreDetails.js | Semelhante ao ConferenceDetails.js, fornece detalhes adicionais sobre uma conferência. Organiza informações em abas, incluindo descrições, datas importantes e outros detalhes relevantes da conferência. |
| PopUpWithMessage.js | Um componente de diálogo (popup) que exibe uma mensagem ao utilizador. |
| SubmissionDetails.js | Este componente apresenta detalhes específicos de uma submissão, como título, autores, status, data de submissão e resumo (abstract). |
| UpdateSubmission.js | Permite aos utilizadores atualizarem uma submissão existente. Inclui campos para editar informações dos autores, título, resumo (abstract) e upload de novos ficheiros. |
| **loading** |  |
| LoadingCircle.js | Este componente exibe um círculo de carregamento para indicar que uma ação está em carregamento. É utilizado para melhorar a experiência do utilizador ao fornecer feedback visual durante operações assíncronas. |
| **navBars** |  |
| ConferenceNavBar.js | Este componente define a barra de navegação específica para conferências, permitindo que os utilizadores naveguem por diferentes seções do sistema de gestão de conferências. |
| ConferencesNavbarMobile.js | Versão móvel da barra de navegação das conferências. Exibe opções de menu em um layout adaptado para dispositivos móveis, permitindo uma navegação eficiente em telas menores. |
| LeftNavbar.js | Este componente renderiza a barra de navegação lateral, permitindo acesso a diferentes páginas da aplicação. Ajusta a visibilidade dos itens de menu com base nas permissões do utilizador. Utiliza contextos para determinar o estado de autenticação e permissões do utilizador, garantindo que apenas opções relevantes sejam exibidas. |
| NavbarMobile.js | Renderiza um menu móvel para navegação, exibindo links como "Return to Home", "Sign in" e "Sign up". É utilizado no sign in ou sign up em dispositivos móveis. |
| NavBarSettings.js | Permite aos utilizadores ajustar configurações de layout, como cores da barra lateral, tipo de barra lateral (transparente, escura, branca) e modo claro/escuro. |
| SignInOutNavBar.js | Barra de navegação específica para páginas de login e registo, oferecendo links para essas páginas e a página inicial. Inclui suporte para visualização em dispositivos móveis e integração com o SignInAndOutLayout. |
| UpperNavBar.js | Barra de navegação superior que inclui breadcrumbs, mensagens de boas-vindas e ícones de notificação. É utilizado na página inicial. |
| **Table** |  |
| CompleteTable.js | Componente de tabela para exibir dados em formato tabular. Suporta seleção de linhas, configuração de altura e largura, e personalização de estilos. Inclui funcionalidades como paginação e seleção de linhas para facilitar a gestão de grandes conjuntos de dados. |

#### OurLayouts

|  |  |
| --- | --- |
| Componente | Descrição |
| PageLayout.js | Um componente de layout que define a estrutura básica de grande parte das páginas deste projeto, incluindo configurações de fundo e disposição dos componentes. |
| SignInAndOutLayout.js | Layout específico para páginas de login e registo. Inclui uma imagem de fundo configurável e um container centralizado para exibir formulários de login ou registo. Utiliza o SignInOutNavBar para manter a consistência visual com outras páginas de autenticação. |

## Backend

### Estrutura do backend:



Esta estrutura foi criada com o propósito de ajudar a organização dos ficheiros. Retira o mesmo princípio do frontend, onde cada página terá o seu código associado, excluindo casos como login e logout que estarão no mesmo ficheiro.

#### Descrição dos Diversos Ficheiros:

##### .env:

Contém informação sensível e importante para o funcionamento de diversas ferramentas do projeto, como por exemplo, a base de dados, conta de e-mail e autenticação com a base de dados. Esta informação poderá conter passwords, IP’s, portas ou mesmo usernames.

##### .gitignore:

Neste ficheiro estão incluídos todos os ficheiros/pastas que são para serem ignorados em todas as alterações feitas para o github. Neste projeto temos somente o node\_modules.

O node\_modules foi ignorado devido ao elevado tamanho que o ficheiro tem.

##### Logs:

Uma imagem com texto, Tipo de letra, captura de ecrã, número

Descrição gerada automaticamente

Esta pasta contém todo o tipo de logs do servidor, nomeadamente logs de base de dados e gestão de utilizadores. Segue uma imagem que mostra alguns dos logs referentes à base de dados:

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

##### Node\_modules:

Node\_Modules é uma pasta que armazena todas as bibliotecas e dependências do projeto instaladas através do NPM\*. Esta pasta facilita a importação e utilização dessas depências no código.

Nota: NPM (Node Package Manager) é um gestor de pacotes para JavaScript, que é usado para instalar, compartilhar e gerir dependências em projetos Node.js.

##### Package-Lock.json:

Este ficheiro é gerado automaticamente pelo NPM e serve para documentar exatamente quais versões de cada dependência foram instaladas, garantido assim a consistência dessas mesmas instalações.

##### Package.json:

É um arquivo que contém metadados do projeto e uma lista de dependências necessárias para o projeto, definindo scripts de execução e outras configurações importantes.

##### Server.js

Ficheiro principal do *backend* que serve para organizar e interligar todo o tipo de atividades associadas com o *backend*. O servidor está preparado para tratar pedidos API, autenticação de utilizadores e gestão de sessões.

#### Pré-requisitos

Node.js e npm instalados.

Variáveis de ambiente PORT e SECRET configuradas para o porto do servidor e o segredo da sessão, respetivamente. (Estas variáveis estão incluídas no ficheiro .env)

### Dependências Principais do Projeto

No nosso projeto, utilizamos várias dependências que desempenham papéis cruciais no funcionamento do servidor. Abaixo, apresentamos uma descrição detalhada de cada dependência e as vantagens de utilizá-las.

#### Express

O Express é um framework minimalista e flexível para Node.js, essencial para a gestão de pedidos HTTP e a integração de funcionalidades através de middleware. Este framework é projetado para facilitar a criação de aplicações web e APIs de maneira rápida e fácil, oferecendo uma camada fina de recursos fundamentais para a web e dispositivos móveis.

Vantagens:

* Facilidade de Uso: API simples e intuitiva que facilita o desenvolvimento de aplicações web.
* Flexibilidade: Suporta a utilização de middleware para adicionar funcionalidades e tratar pedidos HTTP de maneira modular.
* Bastante Reutilizável: Grande comunidade de utilizadores e vasta gama de recursos e plugins disponíveis.

#### Body-Parser

Este é um middleware que atua na análise dos corpos dos pedidos recebidos, principalmente em formatos como JSON e URL-encoded. O Body-Parser extrai toda a parte do corpo de um fluxo de pedido de entrada e expô-lo em um formato mais fácil de ser manipulado dentro do código Node.js.

Vantagens:

* Simplicidade: Facilita a extração e manipulação de dados enviados pelos utilizadores através de formulários ou APIs.
* Eficiência: Garante que os dados sejam acessíveis e manipuláveis de forma eficiente e segura.

#### Passport

O Passport é um middleware extremamente flexível e modular para autenticação em aplicações Node.js. Ele é projetado para servir como um ponto único de configuração de diversas estratégias de autenticação, como login local, OAuth e OpenID, entre outros.

Vantagens:

* Modularidade: Suporta múltiplas estratégias de autenticação, permitindo facilmente a integração de novos métodos.
* Segurança: Facilita a implementação de autenticação robusta e segura.
* Flexibilidade: Não impõe um tipo específico de armazenamento dos dados do utilizador, oferecendo liberdade ao desenvolvedor.

#### Express-Session

Este middleware é utilizado para gerenciar sessões de utilizador através de um armazenamento de sessão no servidor que mantém os estados entre as interações HTTP.

Vantagens:

* Persistência de Sessão: Permite armazenar e recuperar dados do utilizador de forma segura durante a sessão ativa.
* Melhoria de Experiência do Utilizador: Identifica e interage com o utilizador sem necessidade de autenticação a cada nova requisição.

#### CORS (Cross-Origin Resource Sharing)

CORS é um pacote de middleware para o Express que permite configurar de forma flexível as políticas de compartilhamento de recursos entre diferentes origens.

Vantagens:

* Segurança: Controla quais domínios podem aceder aos recursos da aplicação, protegendo contra acessos não autorizados.
* Flexibilidade: Facilita a comunicação entre diferentes domínios, essencial para aplicações que interagem com várias fontes de dados.

#### Connect-pg-simple

Este módulo fornece armazenamento de sessões para express-session utilizando uma base de dados PostgreSQL. É configurado para usar a base de dados através da pool de conexões do nosso módulo pg.

Vantagens:

* Persistência de Sessão: Armazena sessões de utilizador na base de dados PostgreSQL, permitindo a recuperação de sessões em diferentes instâncias do servidor.
* Escalabilidade: Suporta grandes volumes de sessões com desempenho eficiente.

#### Bcrypt

O bcrypt é uma biblioteca utilizada para a criptografia de senhas. Ele é projetado para ser uma maneira segura e robusta de hash de senhas.

Vantagens:

* Segurança: Garante que as senhas sejam armazenadas de forma segura, protegendo contra ataques de força bruta e outras vulnerabilidades.
* Confiabilidade: Amplamente utilizado e testado, proporcionando um nível elevado de segurança na gestão de autenticação.

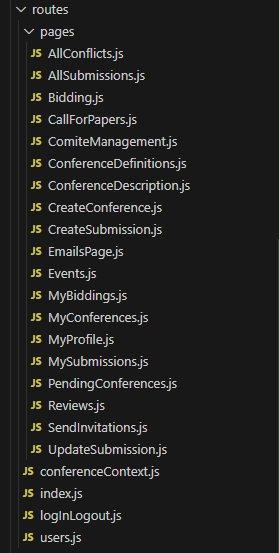
#### PG (node-postgres)

O pg é um cliente PostgreSQL para Node.js, que permite interagir com a base de dados PostgreSQL de maneira eficiente.

Vantagens:

* Desempenho: Otimizado para operações de alta performance com PostgreSQL.
* Facilidade de Uso: API intuitiva que simplifica as operações com a base de dado

### Routes



Dentro desta pasta temos as rotas que irão definir o comportamento de certas páginas no frontend, nomeadamente o registo de conta, login e criação de conferências.

|  |  |  |
| --- | --- | --- |
| Route | Propósito | Funcionalidades |
| AllConflicts.js | Gerir conflitos associados às submissões de conferências. | - Recolhe todos os conflitos relacionados a submissões.    - Utiliza autenticação para assegurar acesso apenas a utilizadores autorizados.  - Consulta a base de dados para obter os conflitos e devolve os resultados. |
| AllSubmissions.js | Gerir submissões de trabalhos para uma conferência específica. | - Obtém submissões de uma conferência específica, incluindo detalhes dos autores.  - Utiliza autenticação para assegurar acesso restrito.  - Filtra submissões com base no ID da conferência fornecido na requisição. |
| Biddings.js |  |  |
| CallForPapers.js | Fornecer informações sobre chamadas para submissão de artigos. | - Obtém detalhes sobre conferências que estão a aceitar submissões.  - Filtra conferências abertas para submissões com base nas datas atuais.  - Retorna informações como nome da conferência, país, datas e tópicos. |
| ComiteManagement.js | Gerir membros do comité de uma conferência. | - Recolhe informações sobre os membros do comité.  - Permite remover e atualizar o papel dos membros.  - Utiliza autenticação para assegurar acesso apenas a utilizadores autorizados. |
| ConferenceDefinitions.js | Fornecer descrição detalhada de uma conferência. | - Recolhe detalhes abrangentes sobre uma conferência específica.  - Utiliza autenticação para assegurar que apenas utilizadores autorizados possam aceder aos dados. |
| ConferenceDescription.js |  |  |
| CreateConference.js | Criar conferências. | - Permite a criação de uma nova conferência com todos os detalhes necessários.  - Fornece tipos e áreas de conferências disponíveis.  - Utiliza autenticação para assegurar que apenas utilizadores autorizados possam criar conferências. |
| CreateSubmission.js | Gerir submissões de trabalhos. | - Permite a criação de novas submissões incluindo detalhes dos autores e ficheiros.  - Utiliza autenticação para assegurar acesso restrito. |
| EmailsPage.js | Gerir envio de emails para membros da conferência. | - Permite o envio de emails para destinatários específicos ou grupos.  - Verifica existência de membros do comité.  - Utiliza autenticação para assegurar que apenas utilizadores autorizados possam enviar emails. |
| Events.js | Gerir eventos de conferências. | - Recolhe todos os eventos relacionados a uma conferência.  - Verifica e cria eventos com base nas datas importantes da conferência.  - Utiliza autenticação para assegurar acesso restrito. |
| MyBiddings.js |  |  |
| MyConferences.js | Fornecer informações sobre conferências do utilizador. | - Obtém conferências associadas ao utilizador, incluindo os papéis desempenhados.  - Utiliza autenticação para assegurar acesso apenas ao utilizador. |
| MyProfile.js | Gerir perfil do utilizador. | - Recolhe e atualiza dados do perfil do utilizador.  - Permite a atualização da senha.  - Utiliza autenticação para assegurar acesso restrito. |
| MySubmissions.js | Gerir submissões do utilizador. | - Obtém submissões feitas pelo utilizador para uma conferência específica.  - Permite eliminar e descarregar ficheiros de submissões.  - Utiliza autenticação para assegurar acesso restrito. |
| PendingConferences.js | Gerir conferências pendentes de aprovação. | - Obtém conferências pendentes.  - Permite aceitar ou rejeitar conferências e notifica os organizadores.  - Utiliza autenticação para assegurar acesso restrito. |
| Reviews.js |  |  |
| SendInvitations.js | Enviar convites para participantes. | - Envia convites por email, gerando códigos de convite e atribuindo papéis.  - Verifica convites enviados e permite eliminá-los.  - Utiliza autenticação para assegurar que apenas utilizadores autorizados possam enviar convites. |
| UpdateSubmissions.js | Atualizar informações de submissões. | - Permite atualização de submissões e obtenção de informações específicas.  - Utiliza autenticação para assegurar acesso restrito. |
| ConferenceContext.js | Gerir o contexto da conferência do utilizador. | - Recolhe e atualiza o contexto atual da conferência do utilizador.  - Utiliza autenticação para assegurar acesso restrito. |
| Index.js | Ponto de entrada para todas as rotas. | - Integra e define todas as rotas no servidor principal.  - Organiza funcionalidades do CMS. |
| Loginlogout.js | Gerir autenticação de utilizadores. | - Permite login e registo de utilizadores utilizando Passport para autenticação.  - Permite logout e fornece informações de autenticação ao frontend.  - Utiliza autenticação para assegurar acesso restrito. |

### PassportStrategies



#### Utilização do Passport para Autenticação

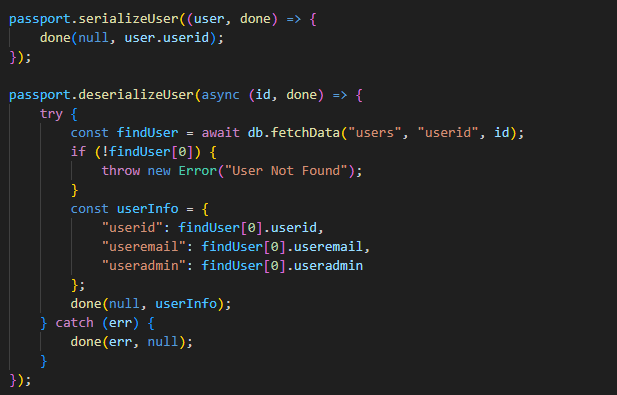
O Passport é uma biblioteca de autenticação para Node.js, que facilita a integração de múltiplas estratégias de autenticação numa aplicação web. No nosso projeto, o Passport desempenha um papel crucial na gestão de autenticação de utilizadores, garantindo que apenas utilizadores autorizados possam aceder a funcionalidades específicas da aplicação.

#### Funcionalidades do Passport

1. Flexibilidade e Modularidade: O Passport é altamente modular e flexível, suportando uma vasta gama de estratégias de autenticação, desde logins tradicionais com nome de utilizador e palavra-passe até autenticação OAuth com provedores como Google, Facebook e Twitter. Esta flexibilidade permite-nos escolher a estratégia que melhor se adapta às necessidades do nosso projeto.
2. Gestão de Sessões: O Passport facilita a gestão de sessões de utilizadores através da serialização e desserialização dos dados do utilizador. Este processo garante que as sessões sejam mantidas de forma segura e eficiente, permitindo a persistência do estado do utilizador entre diferentes pedidos HTTP.

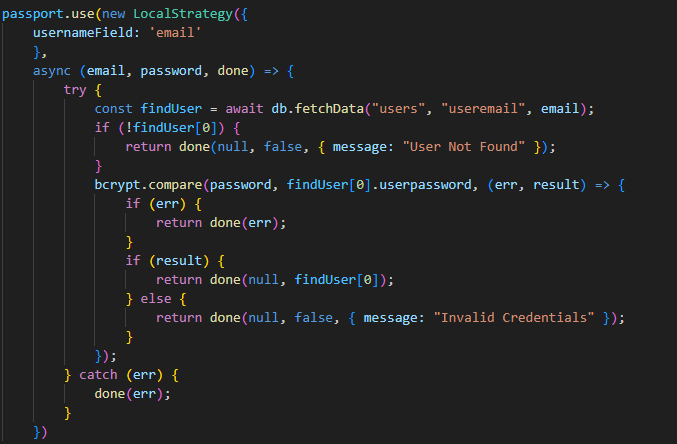
#### Implementação do Passport no Projeto

1. Serialização e Desserialização: A serialização do utilizador envolve guardar o identificador único do utilizador na sessão, enquanto a desserialização utiliza esse identificador para recuperar o utilizador da base de dados. O código a seguir ilustra a implementação desta funcionalidade no nosso projeto:



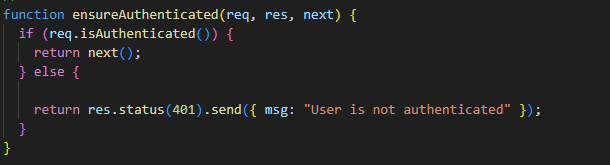
Neste código, a função serializeUser guarda o userid do utilizador na sessão. A função deserializeUser usa o userid para recuperar os detalhes do utilizador da base de dados e reconstituir um objeto de utilizador para ser utilizado na aplicação.

1. Estratégia Local: A estratégia local é configurada para verificar as credenciais do utilizador ao fazer login, utilizando o email e a palavra-passe. A implementação é conforme o seguinte código:

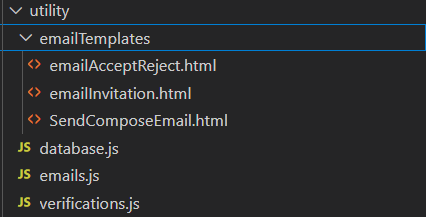


A estratégia local é configurada para utilizar o campo email como nome de utilizador. A função de verificação procura o utilizador na base de dados pelo email fornecido, e compara a palavra-passe fornecida com a palavra-passe armazenada utilizando o bcrypt.

1. Função ensureAuthenticated: A função ensureAuthenticated verifica se o utilizador está autenticado antes de permitir o acesso a uma rota específica. Esta função é utilizada para proteger as rotas que requerem autenticação, como a rota de logout e a rota /authUser:



### Módulo Utility



No nosso projeto, o diretório utility contém vários módulos essenciais que fornecem funcionalidades de suporte à aplicação principal. Entre esses módulos, destacam-se:

* database.js: Gere a comunicação com a base de dados.
* emails.js: Responsável pelo envio de emails utilizando templates HTML.
* verifications.js: Contém funções de verificação e validação de dados.

#### Database.js

Está presente neste ficheiro todas as queries necessárias para que seja possível buscar a informação necessária à base de dados. É possível também atualizar ou eliminar esses dados.

#### Verifications.js

Este ficheiro contém somente uma função que é usada para validar se o usar está ou não autenticado. É usado no passport.

#### Emails.js

O módulo emails.js é responsável por enviar e-mails personalizados utilizando templates HTML. Esta funcionalidade é crucial para comunicações automáticas, como confirmações de registo, convites e outras notificações.

##### Utilização do Nodemailer

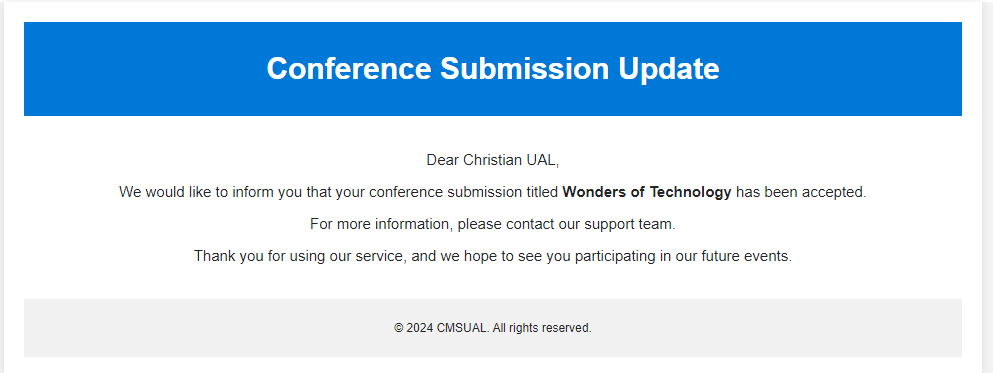
O Nodemailer é uma biblioteca Node.js para enviar emails, que suporta várias formas de transporte, incluindo SMTP, Sendmail, e serviços de email como Gmail, Outlook, entre outros.

Vantagens do Nodemailer:

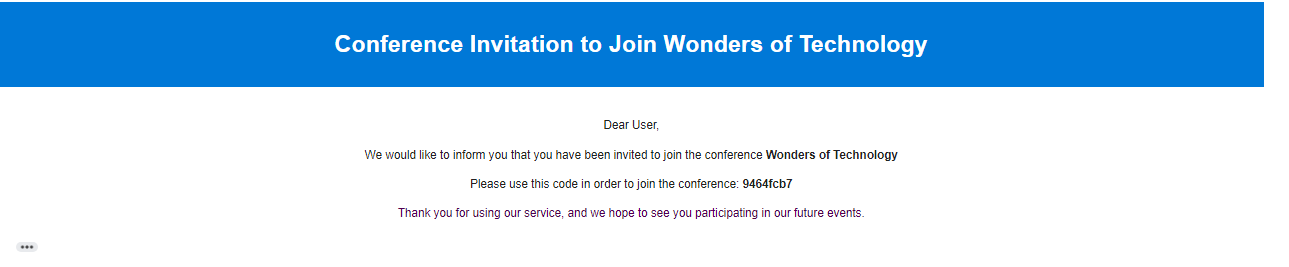
* Facilidade de Configuração: Permite configurar rapidamente um transporte de e-mail utilizando serviços populares.
* Flexibilidade: Suporta várias estratégias de envio de email,
* Personalização: Permite enviar emails HTML e anexos, proporcionando maior flexibilidade na formatação e conteúdo dos emails.

Temos templates de emails para diferentes tipos de ocasiões, por exemplo, se uma conferência foi aceita ou não.

E-mail em caso de aceitação ou rejeição de conferência:



E-mail e convite:



## Segurança

A segurança é uma das nossas prioridades no desenvolvimento deste projeto. Num mundo cada vez mais digital, onde os dados sensíveis são bastante importantes e possivelmente valiosos, garantir a segurança das informações dos utilizadores é essencial. A necessidade de segurança no nosso projeto é especialmente crítica devido à sensibilidade dos dados manuseados, que incluem informações pessoais, detalhes de submissões, e comunicações internas confidenciais.

A proteção de dados não é apenas uma obrigação legal, conforme estipulado por regulamentações como o Regulamento Geral sobre a Proteção de Dados (RGPD) na União Europeia, mas também uma responsabilidade ética para com os utilizadores que confiam os seus dados ao sistema.

Garantir a segurança envolve a implementação de diversos sistemas de segurança, cada um com um papel específico na proteção contra diferentes tipos de ameaças. Estes sistemas incluem a autenticação controlada para garantir que apenas utilizadores autorizados tenham acesso aos dados, a encriptação de palavras-passe para proteger contra acessos não autorizados em caso de fuga de dados, e a configuração segura de servidores e bases de dados para proteger contra-ataques externos.

Além disso, o desenvolvimento seguro do software implica a revisão constante de código para identificar e corrigir vulnerabilidades antes que possam ser exploradas.

### Autenticação e Gestão de Sessões com Passport.js

Serialização e Desserialização\*: Passport.js serializa o utilizador para armazenar apenas o ID do utilizador na sessão. Durante a desserialização, o ID é utilizado para recuperar somente informações necessárias do utilizador, mantendo os dados sensíveis fora da sessão.

\* Serialização é o processo de converter uma informação (neste caso, informação sobre o utilizador) em um formato que pode ser facilmente armazenado.

\* Desserialização é o processo inverso da serialização. É a conversão do formato armazenado de volta para um tipo de informação utilizável.

### Encriptação de Palavras-Passe com Bcrypt

Configuramos a hash com o Bcrypt para que as passwords fiquem protegidas e adicionamos um "salt" às mesmas\*, desta forma, mesmo que haja dois utilizadores com a mesma password a encriptação será diferente para cada um, tornando a palavra-passe segura muito mais difícil de ser quebrada por ataques de força bruta.

O Bcrypt usa uma variante do algoritmo Blowfish, este Blowfish utilizada uma rede de feistel, este permite tamanhos de passwords variáveis (de 32 a 448 bits).

Como variante, Bcrypt adapta o Blowfish com a função “EksBlowfish” para hashing da palavra-passe, a grande diferenção é que “EksBlowfish” foi feito com o propósito de ser mais lento e intensivo em recursos, o que aumenta a dificuldade de ataques de brute force.

\*O salt é um valor aleatório que é adicionado a password antes do hash.

Mesmo que a base de dados seja comprometida, as palavras-passe permanecem seguras devido à robustez do algoritmo Bcrypt.

Vercel

O Vercel fornece HTTPS automaticamente para todos os projetos nele implementados, garantindo que todas as comunicações entre os utilizadores e o servidor sejam encriptadas. Isso protege contra intercepções de dados e ataques de man-in-the-middle.

O Vercel proporciona um ambiente isolado para o frontend, minimizando os riscos de ataques entre aplicações. Suporta também a configuração de CORS para restringir requisições a partir de domínios não autorizados, protegendo a API contra acessos não autorizados.

### Base de Dados em Neon.tech

Oferece backups automáticos, replicação de dados e recuperação de desastres, garantindo que os dados estejam sempre disponíveis e seguros. Além disto, implementa encriptação de dados em repouso e em trânsito, além de fornecer controlos de acesso robustos para gerir permissões de utilizadores.

### CORS (Cross-Origin Resource Sharing)

Garante que apenas aplicações confiáveis possam interagir com a nossa API.

### Envio Seguro de Emails com Nodemailer

O Nodemailer é configurado para usar um serviço de e-mail seguro com suporte TLS (Transport Layer Security), garante que todas as comunicações por e-mail sejam criptografadas e protege os dados do utilizador contra intercepções.

## Diagramas

### Base de dados

# Capítulo 4 – Implementação

## Deploy do nosso backend

Tal como já foi mencionado, o nosso back-end foi construído em nodejs com express. Para podermos publicar o nosso código tivemos bastante dificuldade e tivemos de realizar diversas tentativas para conseguirmos fazer a publicação. A tentativa com mais sucesso foi com o vercel, no entanto assim que o servidor estava *operacional* detetou-se que as cookies de autenticação estavam sempre a ser descartadas e o utilizador não conseguia fazer a autenticação. Após várias semanas a tentar encontrar uma solução em plataformas como o vercel decidimos encontrar um método mais tradicional e como tal pensámos em criar uma máquina virtual numa cloud e configurar o webserver com apache ou nginx e foi neste momento já de desespero que nos foi oferecido um espaço numa máquina pessoa por parte de um colega nosso. Posto isto, o nosso back-end entá num servidor “caseiro” de Linux sem ambiente gráfico e através do nginx a fazer de webserver.

Para nós conseguirmos fazer deploys, temos uma vpn para conseguirmos “chegar” ao nosso servidor e através de sftp ou ssh conseguimos fazer a transferência das nossas atualizações para o servidor, passando assim as alterações para produção.

Deploy do Projeto no Vercel

Com o desenvolvimento concluído, a próxima etapa crucial é o deploy do projeto, garantindo que ele esteja acessível aos utilizadores finais. Escolhemos a plataforma Vercel para hospedar tanto o backend como o frontend, devido à sua robustez, facilidade de uso e capacidade de integração contínua com repositórios Git.

Deploy do Frontend no Vercel

* Configuração Inicial: Começamos por conectar o nosso repositório GitHub ao Vercel. Após a autenticação no Vercel, importamos o projeto do GitHub, selecionando o repositório correspondente ao frontend desenvolvido em React.
* Configurações de Ambiente: Adicionámos variáveis de ambiente necessárias, como URLs da API, chaves de API, entre outras, diretamente na interface do Vercel para garantir que o ambiente de produção tenha as mesmas configurações necessárias que o ambiente de desenvolvimento.
* Deploy Automático: Uma das vantagens do Vercel é a integração contínua. Cada vez que uma nova modificação é feita no repositório GitHub, o Vercel automaticamente inicia um novo deploy, mantendo a aplicação sempre atualizada.

Deploy do Backend no Vercel

Outros:

Objetivos

O objetivo principal deste projeto é desenvolver um sistema de gestão de conferências fácil de usar e que melhore significativamente a eficiência e eficácia na organização de eventos académicos e profissionais.

Seguem alguns objetivos secundários:

* Automatizar o processo de submissão e avaliação de trabalhos, reduzindo potenciais erros manuais e otimizando o tempo dos organizadores e avaliadores.
* Promover a acessibilidade e participação global em conferências através de uma plataforma web intuitiva e acessível.

Metodologia

Este projeto será desenvolvido seguindo uma metodologia ágil, permitindo flexibilidade e adaptação ao longo do processo de desenvolvimento.

Inicialmente, será realizada uma pesquisa de requisitos para entender as necessidades dos utilizadores finais, seguida pela prototipagem e desenvolvimento iterativo do sistema. As tecnologias React e Node.js serão utilizadas para o desenvolvimento do Frontend e Backend.

Importância do Projeto

A importância deste projeto reside na sua capacidade de transformar a maneira como conferências são organizadas, oferecendo uma solução que não só poupa tempo, mas também aumenta a qualidade dos eventos académicos e profissionais. Ao automatizar processos críticos e fornecer uma plataforma acessível, este CMS tem o potencial de ampliar a participação global em conferências, superando barreiras geográficas.

Além disso, o projeto contribui para a literatura académica em sistemas de informação, demonstrando como as tecnologias web podem ser aplicadas para resolver problemas complexos na organização de eventos.