SIG2E - Sistema Informático para Gestão de Espaços e Equipamentos

Projeto P20

Relatório Intermédio da Unidade Curricular de Projeto

Licenciatura em Engenharia de Informática, Redes e Telecomunicações

Departamento de Eng. Eletrónica e Telecomunicações e de Computadores

Instituto Superior de Engenharia de Lisboa

Alunos:

André Santos, n.º 48705

João Teixeira, n.º 48710

Orientadores:

Prof. Tiago Miguel Braga da Silva Dias

Prof. Diogo Cardoso

Abril de 2023

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

As atividades de natureza laboratorial e/ou experimental implementam estratégias de ensino que visam facilitar a compreensão e a consolidação de determinados conteúdos programáticos, promovendo uma melhor visão e profundidade do conteúdo lecionado através da interligação dos conteúdos técnicos aos teóricos e motivando a autonomia dos/as alunos/as para o seu estudo. Acresce ainda que estas atividades também potenciam nos/as estudantes o desenvolvimento de competências a nível da comunicação, da argumentação e do diagnóstico de problemas. O trabalho experimental em laboratório é, portanto, um fator crucial para a formação de novos/as engenheiros/as.

Infelizmente, as medidas implementadas nos últimos anos para combater a pandemia por COVID-19 afastaram os/as alunos/as dos espaços físicos de ensino, com significativo prejuízo para a sua formação, não só ao nível da sua componente prática e experimental, como também nos seus hábitos de trabalho. Assim, no atual momento, é muito importante desenvolver estratégias e implementar metodologias que incentivem os/as alunos/as a voltar a usar os laboratórios e os seus equipamentos, não só durante as aulas programadas, mas também no seu tempo livre.

Atualmente, existem diversos fatores que condicionam e desmotivam a utilização em regime livre dos laboratórios do DEETC, desde logo os procedimentos não automatizados e de caráter presencial para a reserva dos espaços e equipamentos, a dificuldade em aceder à informação sobre os horários disponíveis ou o desconhecimento da disponibilidade dos equipamentos. Este projeto visa combater estas dificuldades, enquadrando-se numa estratégia para promoção do trabalho em laboratório junto dos/as alunos/as dos cursos de engenharia informática do Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores (DEETC) do ISEL, em que se inclui o curso de Licenciatura em Engenharia Informática, Redes e Telecomunicações (LEIRT).

2. Descrição do Projeto

Este projeto tem como objetivo o desenvolvimento de um sistema informático para gestão e reserva dos espaços e equipamentos de laboratório afetos ao DEETC do ISEL, designado por Sistema Informático para Gestão de Espaços e Equipamentos (SIG2E).

O sistema a desenvolver será composto por dois elementos: uma aplicação de gestão, designada por *Backend*, e uma aplicação web, designada por *Website*. A aplicação de gestão será a componente que irá fornecer lógica ao sistema, sendo responsável pelo armazenamento e processamento dos dados sobre os espaços e os equipamentos geridos no sistema e os seus utilizadores. A aplicação *web* será a interface de utilização que permitirá aos utilizadores interagirem com o sistema, através de um explorador de *Internet*, possibilitando assim a sua gestão e utilização.

A Figura 1 ilustra a ligação entre os dois elementos do SIG2E, onde o *Frontend* é implementado pela aplicação *web* e o *Backend* por uma base de dados (BD), uma Interface de Programação de Aplicação (em inglês, *Application Programming Interface* - API) e um servidor. A BD é o componente que irá organizar e armazenar a informação de modo a esta ser consistente e de acesso rápido e eficiente, permitindo ainda através de *queries* um processamento personalizado. Através da API será possível estabelecer a comunicação entre o *Frontend* e o *Backend* usando rotas no servidor e recorrendo ao Protocolo de Transferência de Hipertexto (em inglês, *Hypertext Transfer Protocol* - HTTP), um dos protocolos mais utilizados devido à sua fiabilidade, simplicidade e suporte. O servidor executará todas as funcionalidades do sistema, incluindo o cumprimento das regras lógicas do algoritmo, o acesso à informação da BD e o processamento dos pedidos recebidos pela API.

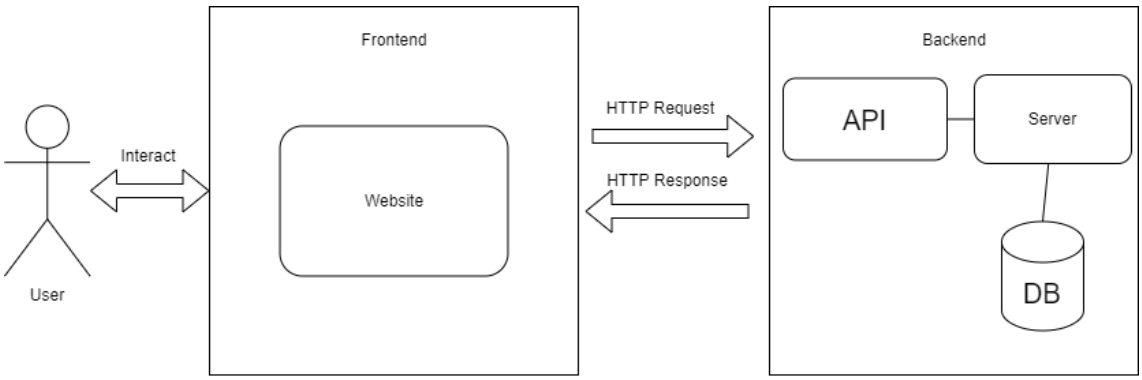

Figura 1 - Diagrama da arquitetura do SIG2E

3. Tarefas a Realizar

A Figura 2 apresenta as tarefas a realizar para o desenvolvimento do SIG2E num diagrama UML do tipo caso de uso [1].

Dentro do bloco designado por SIG2E estão representadas todas as funcionalidades a implementar, bem como a sua interligação. Em suma, o sistema terá de possuir funcionalidades de autenticação, gestão e notificação de falhas. O acesso ao sistema será realizado através do e-mail institucional do/a utilizador/a. Para a gestão, será necessário implementar as funcionalidades de adição, remoção e alteração de propriedades de utilizadores, espaços e equipamentos, bem como a reserva destes dois últimos. Será implementado também um mecanismo de notificação de falhas, responsável por gerar *tickets* para os gestores de laboratório, com a consequente alteração da condição do equipamento ou espaço reportado e beneficiando o informante da avaria.

Os atores à esquerda deste bloco representam os tipos de utilizadores possíveis, i.e., Aluno, Professor, Responsável de Laboratório ou Administrador, em que cada um possui um conjunto de permissões que são herdadas hierarquicamente, conforme o sistema de Controlo de Acesso Baseado em Funções (em inglês, *Role-Based Acess Control* - RBAC). Por exemplo, um Responsável de Laboratório terá todas as permissões de um Professor, que por sua vez terá também todas as permissões de um Aluno. No entanto, um Responsável de Laboratório não conseguirá exercer as funções de um Administrador. Finalmente, os atores à direita do bloco SIG2E representam as entidades responsáveis por executar as funcionalidades.

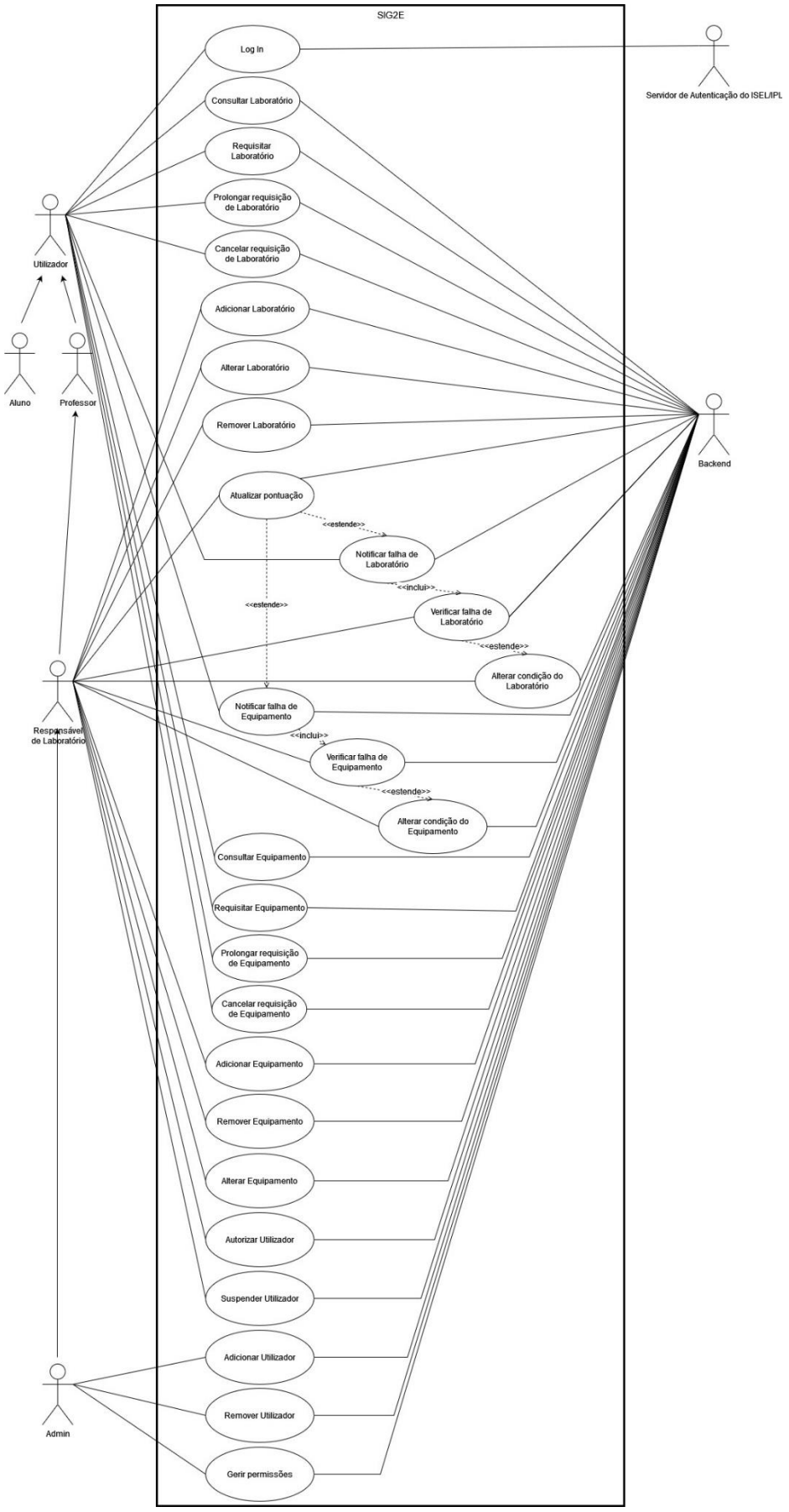

Figura 2 - Diagrama de caso de uso do sistema SIG2E

4. Seleção das Tecnologias

No capítulo 2 apresenta-se um levantamento das componentes necessárias para a construção do SIG2E: uma base de dados, um servidor, uma API e uma aplicação web. Neste capítulo, analisa-se os requisitos de cada um desses componentes detalhadamente com o objetivo de selecionar as tecnologias mais adequadas para sua implementação

4.1 Base de dados

Um sistema de gestão de base de dados (SGBD) é responsável pela gestão de uma ou mais bases de dados, sendo o seu principal objetivo gerir o acesso e processamento dos dados. Estas BD podem tanto ser baseadas do tipo relacional, como não relacional.

As bases de dados relacionais, ou mais frequentemente bases de dados de consulta estruturada (em inglês, *Structured Query language*, SQL), são sistemas de armazenamento de informação estruturada baseadas no modelo relacional [2], sendo este tipo de sistemas constituídos por entidades e relações. Uma entidade (tabela) é um elemento caracterizado por atributos (colunas) e que apresenta os seus valores em énuplo (linhas, ou ainda, em inglês, tuple). A relação é o modo como as entidades (tabelas) se associam. As principais vantagens deste tipo de sistema são a sua ferramenta de *queries* SQL para consulta e processamento de dados e as suas características de atomicidade, consistência, isolamento e durabilidade (ACID), garantindo assim uma maior fiabilidade e integridade da informação [3]. O princípio de atomicidade garante que uma transação (uma ou várias mudanças na BD) é tratada por inteira, sendo esta apenas bem-sucedida ou malsucedida. A consistência assegura a integridade dos dados, ou seja, todos os dados de uma transação devem seguir as regras de integridade da tabela. O isolamento certifica que diferentes transações a correr em modo paralelo não têm efeito umas sobre as outras. Por fim, a durabilidade garante que a informação deve persistir permanentemente, mesmo em casos de erros ou quebras de energia.

No entanto, estes sistemas relacionais apresentam dificuldades na escalabilidade pois possuem um modelo de dados restrito e apresentam complicações na distribuição do processamento por diferentes máquinas (também conhecido como escalonamento horizontal, em inglês, *Horizontal Scaling*), possuindo assim tipicamente um escalonamento vertical (em inglês, *Vertical Scaling*) [4] [5]. Devido a esta dificuldade foram criadas bases de dados não relacionais, também conhecidas por NoSQL.

As bases de dados NoSQL são sistemas de armazenamento de informação que utilizam diferentes tipos de documentos, sendo estes geralmente em formato JSON (do inglês, *JavaScript Object Notation*) e em relações de chave-valor. Ao contrário dos modelos relacionais, geralmente, as bases de dados NoSQL seguem os princípios BASE (basicamente acessível, estado suave e eventualmente consistente) que não garantem a consistência nem fiabilidade dos dados. O princípio de basicamente acessível diz que a informação está sempre disponível ao espalhá-la pelos diversos nós de um *cluster* [2]. Estado suave significa que o modelo de dados

não garante consistência da informação e que esta pode ser alterada ao longo do tempo. Por fim, a base de dados é eventualmente consistente, ou seja, é possível realizar operações antes da informação se tornar consistente. Devido a este tipo de sistemas não utilizarem um modelo de dados restrito possuem uma escalabilidade horizontal superior aos dos sistemas relacionais [6].

A Tabela 1 apresenta de um modo sucinto as diferenças levantadas nos parágrafos anteriores

Base de dados SQL	Base de dados NoSQL
Sistema de gestão de base de dados relacional	Sistema de gestão de base de dados não relacional
Escalabilidade Vertical	Escalabilidade Horizontal
Modelo de dados Restrito	Modelo de dados Flexível
Transações ACID	Transações BASE

Tabela 1 - Características das bases de dados SQL e NoSQL

Como o sistema SIG2E é constituído por atributos e entidades intrinsecamente relacionados, por exemplo, um equipamento (entidade) tem sempre de ter um indentificador único e um estado de utilização (atributos) e pode estar a ser ocupado por um utilizador (entidade), que também terá atributos como indentificador único e nome. Deste modo, devido à natureza e baixo volume de dados que se espera armazenar, a opção mais indicada são bases de dados relacional, não sendo assim necessário considerar soluções que permitam escalabilidade horizontal.

Das diferentes opções existentes no mercado de sistemas de gestão de base de dados relacionais, neste trabalho será utilizado o PostgreSQL, devido à sua natureza gratuita e de código aberto e à sua grande comunidade ativa, possuindo ainda funcionalidades bastante competitivas com os restantes tipos de base de dados disponíveis.

4.2 Servidor

O servidor é a componente do sistema que implementa a aplicação responsável pelo processamento dos pedidos da aplicação web, implementando a lógica da aplicação, interagindo com a base de dados e disponibilizando as rotas para a API.

Para a criação do servidor web, foram analisadas diversas linguagens de programação e tecnologias que atendessem às necessidades do projeto, deste modo descartando o uso de linguagens com *frameworks* que não disponibilizam nativamente ou via bibliotecas os recursos necessários para a realização do projeto. Além disso, embora Java e PHP ainda sejam extensamente utilizados, existem linguagens mais modernas e dinâmicas que oferecem uma gama mais ampla de recursos e ferramentas de desenvolvimento . Outro critério utilizado na seleção das linguagens foi a curva de aprendizagem, tendo sido descartadas linguagens que apresentam um alto nível de dificuldade na sua aprendizagem, como Ruby. Dentre as linguagens restantes, JavaScript e Python foram as mais relevantes pois possuem *frameworks* que agilizam o processo de implementação e são mais compatíveis com as necessidades do projeto. Após considerar as *frameworks* Django [7] e Flask [8] para Python, e o Express [9] e o Deno [10] para JavaScript, optou-se por utilizar JavaScript devido a experiência prévia com esta tecnologia e por ser uma linguagem que pode ser utilizada tanto no desenvolvimento do cliente como no do servidor.

No entanto, devido à sua natureza dinâmica, o JavaScript é mais propício a problemas uma vez que não tem um analisador estático [11], desta forma, optou-se por utilizar TypeScript [12], uma linguagem que estende JavaScript adicionando tipificação. TypeScript é uma linguagem de *scripting*, ou seja, os programas são interpretados em tempo de execução, nativamente pelo navegador. Consequentemente, para a sua execução no servidor surge a necessidade de dispor de um ambiente de execução, que é um sistema que permite a execução de código fora de um navegador web e é responsável pela gestão dos recursos do sistema, como o acesso à memória e rede. No caso de JavaScript existem três principais ambientes de execução: NodeJS [13], Deno e Rhino [14]. Entre estas opções o NodeJS destaca-se pela sua ampla adoção e suporte ativo. O Deno, por sua vez, é conhecido pela sua segurança e suporte nativo ao TypeScript, embora seja uma ferramenta mais recente e possua uma comunidade menos ativa que o NodeJS. Por fim, o Rhino oferece integração com Java, sendo este executado na máquina virtual Java (em inglês, Java Virtual Machine, JVM). Uma vez que existem mais bibliotecas e *frameworks* e também devido a experiência prévia com a tecnologia, o ambiente de execução adotado para o servidor é NodeJS [15].

Para disponibilizar os recursos do servidor via uma API existem diferentes *frameworks* que podem ser utilizadas, tal como Express, Hapi [16] e Koa [17]. Express foi desenvolvido para auxiliar na criação de aplicações web, fornecendo um conjunto de recursos que agilizam o processo de desenvolvimento, desde declaração de rotas a manipulação de pedidos e respostas [18]. O Hapi foi criado para resolver problemas de velocidade do Express, armazenando em cache os pedidos para respostas mais rápidas [19], o Koa é uma opção mais leve computacionalmente pois remove componentes não estritamente necessários [20]. Neste trabalho optou-se por usar Express dado que as outras duas *frameworks* são suas derivadas e porque é a que tem mais suporte e documentação disponível.

4.3 API

Conforme apresentado na secção 2, o modo de comunicação entre o *Frontend* e o *Backend* do SIG2E é definido por uma API, que estabelece o formato de dados da comunicação e a forma de acesso aos seus métodos.

Alguns dos tipos de API mais conhecidos são baseados no protocolo SOAP [21] (do inglês, *Simple Object Access Protocol*) ou REST [21] (em inglês, Representational State Transfer).

As API baseadas no protocolo SOAP permitem a exposição de recursos do servidor que são invocados pelo cliente por meio de mensagens no formato XML (do inglês, *Extensible Markup Language*). Essas mensagens são compostas por um envelope que encapsula os detalhes da mensagem, incluindo o corpo e o cabeçalho, este último sendo opcional. O corpo da mensagem contém a informação a ser transmitida para o destinatário, que pode ser uma invocação de recursos por parte do cliente ou uma resposta por parte do servidor. O cabeçalho contém informações adicionais relacionadas à mensagem e ao processamento da mesma, como meta dados, informações de autenticação, configurações de segurança, entre outros [22].

As API baseadas no protocolo REST pressupõem que as comunicações são sem estado, o que significa que o servidor não armazena informações sobre os clientes. Em vez disso, são os próprios clientes que, para cada pedido, enviam as informações necessárias para que o servidor possa interpretar o pedido adequadamente. A forma como o cliente invoca os pedidos ao servidor é por meio de métodos HTTP, como GET, PUT, POST e DELETE, utilizando um indicador de recurso único (em inglês, *Uniform Resource Identifier* - URI) que identifica o recurso solicitado. Ao contrário do protocolo SOAP, os dados trocados nas mensagens de uma API REST podem assumir vários formatos, como XML, JSON, texto em claro, entre outros.

No desenvolvimento do SIG2E optou-se por implementar uma API que implementa o protocolo REST, pois estas requerem menos processamento do que as baseadas no protocolo SOAP, consumindo uma menor largura de banda e sendo mais simples de desenvolver [23].

4.4 Aplicação Web

Uma aplicação web é um programa informático disponibilizado aos utilizadores através de um navegador web (em inglês, *browser*) e que é entregue através da rede por um servidor web. Nesta transação, que geralmente utiliza o protocolo HTTP, o navegador envia pedidos a um servidor web, que os processa e envia de volta ao cliente para serem assim apresentados no navegador. Estas aplicações consistem em páginas web e que são comumente construídas através das linguagens HTML, CSS (do inglês, *Cascading Style Sheets*) e Javascript. A linguagem HTML é utilizada para definir a estrutura e conteúdo estático das páginas, sendo este o elemento base da aplicação web. No entanto, para melhorar a aparência e apresentação visual das páginas, é utilizada à linguagem CSS, que é responsável pela definição de estilo e disposição dos elementos. Assim, o HTML e o CSS trabalham em conjunto para tornar a aplicação web

mais atrativa e fácil de usar. Por fim, de modo a criar páginas dinâmicas é possível executar código Javascript para atualizar o conteúdo das páginas HTML em tempo real sem ter de atualizar toda a página, sendo assim aumentada a interatividade.

Devido ao processo de criação de páginas web ser algo moroso e suscetível a erros, ao longo dos anos foram surgindo *frameworks* que auxiliam e agilizam o processo de construção de aplicações web através da utilização de recursos pré-construídos e testados, como componentes expansíveis para o domínio da aplicação (por exemplo, existir um botão genérico que pode ser utilizado para construir botões específicos à aplicação), oferecendo assim abstração. Com isto, será utilizada uma *framework* de CSS para customizar a apresentação dos documentos que foram gerados pela *framework* de construção de elementos da página.

Relativamente à geração dos elementos que compõem as páginas web, as *frameworks* mais relevantes atualmente são Angular, React e Vue. Neste projeto optou-se por utilizar React pois é reativo, reutilizável, atual e possui uma comunidade bastante ativa, sendo esta a *framework* de *Frontend* mais utilizada [24].

Desta decisão resulta que as opções mais viáveis para customizar a apresentação dos documentos gerados com a *framework* React, i.e. o estilo da página, são Bootstrap-React, MUI e TailwindCSS. Neste caso, escolheu-se utilizar MUI devido aos seus elementos pré-construídos apresentarem um aspeto mais moderno.

5. Implementação do Projeto

Neste capítulo, descreve-se a implementação dos componentes do SIG2E apresentados na Figura 1.

5.1 Base de Dados

Um dos componentes fundamentais do SIG2E é a base de dados que armazena toda a informação do sistema. A projeção da BD foi realizada através do modelo entidade relação (ER) [25] que representa graficamente as entidades e relações de um sistema de informação de um modo simples e concreto, facilitando a identificação das entidades e as suas relações [26]. Para a criação do modelo ER foi necessário identificar as entidades do sistema bem como os seus atributos e chaves.

O SIG2E visa gerir equipamentos e espaços, entendendo-se que a gestão significa que os gestores podem realizar a sua alteração e todos utilizadores a sua requisição. Os espaços referenciam-se aos laboratórios do ISEL caracterizados por terem bancadas de trabalho. Uma bancada de trabalho pode ou não ter equipamentos acoplados, pelo qual a sua requisição inferência também a requisição destes equipamentos. Um equipamento é requisitado e pode ou não ser móvel caso esteja fixo numa bancada. A requisição pode ser para múltiplos equipamentos e bancadas, possuindo ainda uma prioridade, i.e., uma reserva de aula é mais importante que uma reserva em tempo livre, pelo qual será necessário indicar o motivo da reserva e a sua duração. Após a obtenção do espaço ou equipamento, este pode apresentar defeitos que condicionam a sua utilização, com isto será necessário apresentar uma notificação de avaria.

A Figura 3 representa o modelo ER do SIG2E, onde as entidades são representadas por retângulos, as relações por losangos, os atributos por elipses e as chaves são sublinhadas. As

entidades e as relações fracas são identificadas pela sua figura com dupla linha.

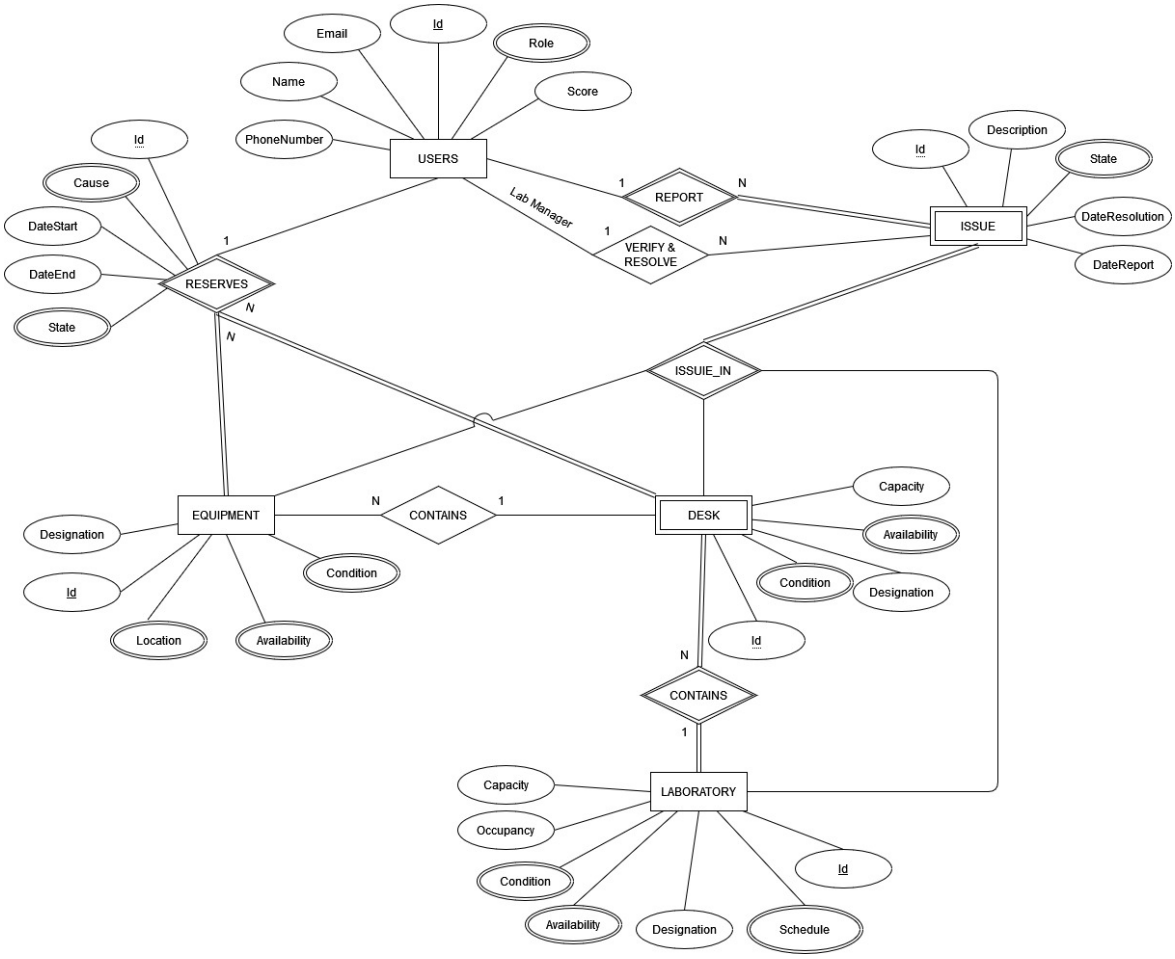

Figura 3 - Diagrama do modelo ER SIG2E

Após a conclusão do modelo ER realizou-se a transformação para o modelo físico. Para esta transformação, foram criadas tabelas correspondentes às entidades do modelo ER com colunas correspondentes aos atributos. Neste modelo de dados, os atributos são tipificados e existe o conceito de chaves primárias e chaves estrangeiras. Uma chave primária segue os mesmos princípios de chave do modelo ER e as chaves estrangeiras são elementos que fazem referência a outra tabela de modo a criar uma relação. O modelo físico do SIG2E está presente no apêndice.

6. Calendarização

A Figura 4 apresenta o cronograma das tarefas do projeto através de um diagrama *Gantt.* No diagrama, o projeto foi dividido pelas suas quatro principais componentes: BD, Servidor, API e aplicação web. Até à data, os quatro componentes foram todos estudados e foi produzida a documentação justificativa do processo de seleção de cada um. Foi também documentado a projeção da BD (Digrama de Entidade-Relacionamento), a sua implementação e os testes automáticos realizados. Para a entrega final, todas as componentes serão implementadas e testadas em simultâneo, dando prioridade à implementação de cada funcionalidade isoladamente.

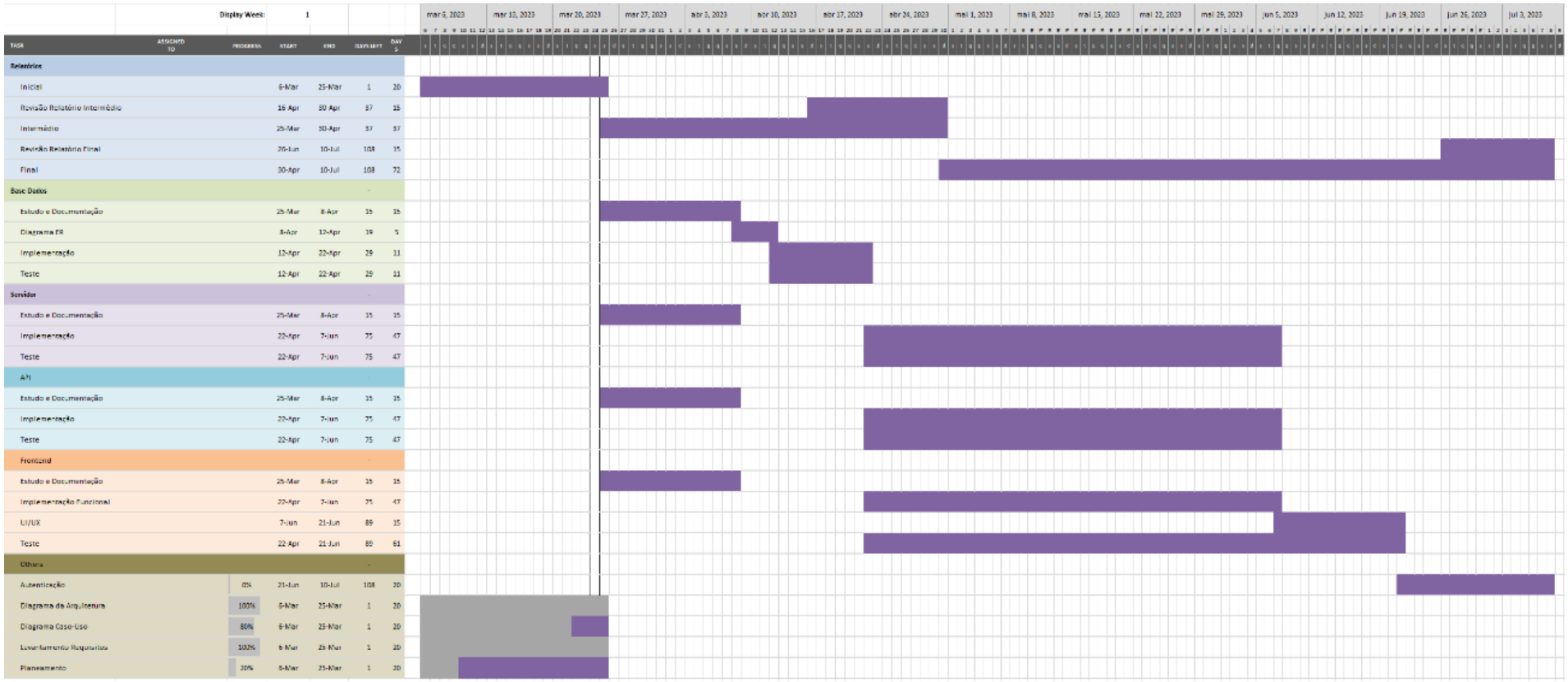

Figura 4 - Diagrama de Gantt do projeto SIG2E

7. Referências

[1] M. Fowler, “ACM Digital Library,” 1 10 2003. [Online]. Available: https://dl.acm.org/doi/10.5555/861282.

[2] E. F. Codd, The Relational Model for Database Management: Version 2, Addison-Wesley, 1990.

[3] B. Medjahed, M. Ouzzani e A. Elmagarmid, 2009. [Online]. Available: https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1107&context=ccpubs.

[4] A. P. R. R. Ragul R, “IJITEE,” 03 2020. [Online]. Available: https://www.ijitee.org/wp-content/uploads/papers/v9i5/E2418039520.pdf.

[5] A. H. A. Hinai, “DiVA,” 31 08 2016. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:957015/FULLTEXT01.pdf.

[6] J. Pokorny, “ResearchGate,” 12 2011. [Online]. Available: https://www.researchgate.net/publication/221237715_NoSQL_Databases_a_step_to_database_scalability_in_Web_environment.

[7] “Django,” [Online]. Available: https://www.djangoproject.com/start/overview/.

[8] “Flask,” [Online]. Available: https://flask.palletsprojects.com.

[9] “NPM,” [Online]. Available: https://www.npmjs.com/package/express.

[10] “Deno,” [Online]. Available: https://deno.land/manual@v1.32.3/introduction#feature-highlights.

[11] Z. Gao, C. Bird e E. T. Barr, “Microsoft,” 09 2017. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/09/gao2017javascript.pdf.

[12] “TypeScript,” [Online]. Available: https://www.typescriptlang.org/.

[13] "NodeJS," [Online]. Available: https://nodejs.org/en/about.

[14] "GitHub," [Online]. Available: https://github.com/mozilla/rhino.

[15] G. Jadhav e F. Gonsalves, "IRJET," 06 2020. [Online]. Available: https://www.irjet.net/archives/V7/i6/IRJET-V7I61149.pdf.

[16] "NPM," [Online]. Available: https://www.npmjs.com/package/@hapi/hapi.

[17] "NPM," [Online]. Available: https://www.npmjs.com/package/koa.

[18] F. Copes, The Express Handbook, 2018.

[19] "Hapi," [Online]. Available: https://hapi.dev/tutorials/caching/?lang=en_US.

[20] "KoaJS," [Online]. Available: https://koajs.com/.

[21] A. Soni e V. Ranga, "ResearchGate," 07 2019. [Online]. Available: https://www.researchgate.net/publication/335419384_API_Features_Individualizing_of_Web_Services_REST_and_SOAP.

[22] G. M. Waleed e R. B. Ahmad, "ResearchGate," 12 2008. [Online]. Available: https://www.researchgate.net/publication/224386981_Security_protection_using_simple_object_access_protocol_SOAP_messages_techniques.

[23] F. Halili e E. Ramadani, "ResearchGate," 28 02 2018. [Online]. Available: https://www.researchgate.net/publication/323456206_Web_Services_A_Comparison_of_Soap_and_Rest_Services.

[24] M. Bauer, "DiVa," 07 2021. [Online]. Available: https://uu.diva-portal.org/smash/get/diva2:1591850/FULLTEXT01.pdf.

[25] R. E. a. S. Navathe, Fundamentals of Database, USA: Addison-Wesley Publishing Company, 2010.

[26] A. Badia, 2001. [Online]. Available: https://www.researchgate.net/publication/220415410_Entity-Relationship_modeling_revisited.

[27] “Visual Paradigm,” [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/.

[28] “Adatum,” [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1749772821000087.

[29] S. Journals. [Online]. Available: https://journals.sagepub.com/doi/10.1177/0020720916689103?icid=int.sj-full-text.similar-articles.7.

[30] J. Makkonen, “Aalto University,” 12 11 2017. [Online]. Available: https://aaltodoc.aalto.fi/bitstream/handle/123456789/29224/master_Makkonen_Joni_2017.pdf.

[31] K. R. Srinath, “IRJET,” 12 2017. [Online]. Available: https://www.irjet.net/archives/V4/i12/IRJET-V4I1266.pdf.

[32] “Tech Insider,” Sun Microsystems, 1995. [Online]. Available: https://www.tech-insider.org/java/research/acrobat/9503.pdf.

[33] “Deno,” [Online]. Available: https://deno.com/runtime.

[34] D. Marinescu e M. Kaufmann, Cloud Computing – Theory and Practice, 2nd Edition, 2017.