



Universidade do Minho

Mestrado em Engenharia Informática Requisitos e Arquiteturas de Software (2023/24)

PROBUM

Solução arquitetural e Implementação

Grupo 2-A

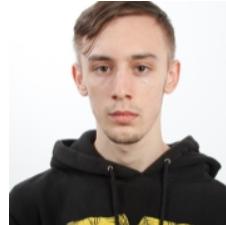
PG53825	Gabriel Alexandre Monteiro da Silva
PG53840	Gonçalo Lobo Freitas
PG52682	Gonçalo Manuel Maia de Sousa
A93318	Jéssica Macedo Fernandes
PG53929	João Paulo Peixoto Castro
PG53932	João Pedro Antunes Gonçalves
PG54004	Luís Alberto Barreiro Araújo
PG54123	Orlando José da Cunha Palmeira
PG54232	Sérgio Miguel Cabral Passinhas dos Santos Costa
PG52705	Tiago Passos Rodrigues
A98728	Vasco Rafael Barroso Gonçalves Rito



PG53825



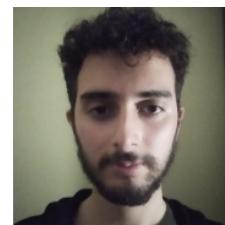
PG53840



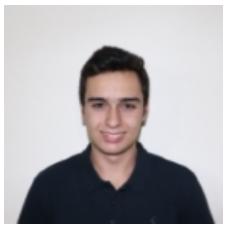
PG52682



A93318



PG53929



PG53932



PG54004



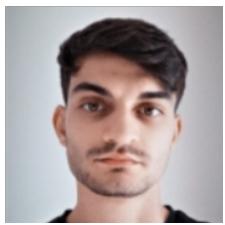
PG54123



PG54232



PG52705



A98728

Prefácio

De seguida, segue-se uma pequena tabela com as notas individuais de cada elemento (valor entre -2.0 e 2.0), acordado por todos os constituintes do grupo:

	Membro	Contributo
PG53825	Gabriel Alexandre Monteiro da Silva	-0.2
PG53840	Gonçalo Lobo Freitas	-0.3
PG52682	Gonçalo Manuel Maia de Sousa	-0.1
A93318	Jéssica Macedo Fernandes	-0.2
PG53929	João Paulo Peixoto Castro	+0.2
PG53932	João Pedro Antunes Gonçalves	+0.2
PG54004	Luís Alberto Barreiro Araújo	-0.3
PG54123	Orlando José da Cunha Palmeira	+1.3
PG54232	Sérgio Miguel Cabral Passinhas dos Santos Costa	0
PG52705	Tiago Passos Rodrigues	-0.5
A98728	Vasco Rafael Barroso Gonçalves Rito	-0.1

Conteúdo

1	Introdução e Objectivos	7
1.1	Resumo dos requisitos funcionais	7
2	Restrições	10
3	Requisitos de qualidade/não funcionais	12
4	Contexto e âmbito do sistema	14
5	Estratégia da solução	15
5.1	Decomposição funcional	15
5.2	Padrão arquitetural	16
5.2.1	Identificação de microsserviços	18
5.3	Modelos de dados das bases de dados dos microsserviços	19
5.3.1	Gestão de Utilizadores	19
5.3.2	Gestão de Salas	19
5.3.3	Gestão de Provas	19
5.3.4	Gestão de Notificações	22
5.4	Tecnologias para a implementação	22
5.4.1	Microsserviços e <i>API Gateway</i>	22
5.4.2	Bases de dados	22
5.4.3	Frontend	23
5.5	Decisões organizacionais	23
6	<i>Building Block View</i>	24
6.1	Gestão de utilizadores	25
6.2	Gestão de salas	26
6.3	Gestão de provas	27
6.4	Gestão de notificações	28
7	<i>Runtime View</i>	30
7.1	Diagramas de sequência	30
7.1.1	Registar docentes	30
7.1.2	Consultar prova corrigida	31
7.1.3	Gerir salas	32
7.1.4	Publicar classificações da prova	33
7.1.5	Aceder às notificações	34
7.1.6	Partilhar prova	35
7.1.7	Registar alunos	35
7.1.8	Classificar respostas	36
7.1.9	Editar perfil	37
7.1.10	Autenticar	38

7.1.11	Consultar detalhes da prova	39
7.1.12	Criar questões	39
7.1.13	Editar prova	40
7.1.14	Responder a prova	41
7.1.15	Editar questões	42
7.1.16	Criar prova	43
8	<i>Deployment View</i>	45
8.1	Contexto de desenvolvimento e teste	45
8.2	Contexto de produção	46
9	Alterações arquitecturais efectuadas	48
9.1	Use cases	48
9.1.1	Use case - Criar prova	48
9.1.2	Use case - Criar questões	48
9.1.3	Use case - Responder a prova	49
9.1.4	Use case - Classificar respostas	49
9.2	Bases de dados	49
9.2.1	Gestão de salas	49
9.2.2	Gestão de provas	50
9.2.3	Gestão de Utilizadores	50
9.3	<i>Building Block View</i>	50
9.3.1	Gestão de provas	50
9.3.2	Gestão de Notificações	50
10	Microsserviços	52
10.1	Gestão de provas	52
10.1.1	Rotas da API	52
10.2	Gestão de salas	53
10.2.1	Rotas da API	53
10.3	Gestão de Utilizadores	54
10.4	Gestão de Notificações	55
10.4.1	Rotas do microserviço de notificações	55
11	Grau de completude da solução	57
11.1	Requisitos funcionais	57
11.2	Requisitos não funcionais	58
12	Contributos individuais na implementação	59

Listas de Figuras

1.1	<i>Use cases</i> do sistema	8
4.1	Diagrama de contexto (<i>business</i>)	14
5.1	Diagrama de blocos - decomposição funcional	15
6.1	Diagrama de componentes	24
6.2	Diagrama de componentes relativo à gestão de utilizadores	25
6.3	Diagrama de componentes relativo à gestão de salas	26
6.4	Diagrama de componentes relativo à gestão de provas	27
6.5	Diagrama de componentes relativo à gestão de notificações	28
7.1	Diagrama de sequência - Registar docentes	31
7.2	Diagrama de sequência - Consultar prova corrigida	32
7.3	Diagrama de sequência - Gerir salas	33
7.4	Diagrama de sequência - Publicar classificações da prova	34
7.5	Diagrama de sequência - Aceder às notificações	34
7.6	Diagrama de sequência - Partilhar prova	35
7.7	Diagrama de sequência - Registar alunos	36
7.8	Diagrama de sequência - Classificar respostas	37
7.9	Diagrama de sequência - Editar perfil	38
7.10	Diagrama de sequência - Autenticar	39
7.11	Diagrama de sequência - Consultar detalhes da prova	39
7.12	Diagrama de sequência - Criar Questões (actualizado)	40
7.13	Diagrama de sequência - Editar prova	41
7.14	Diagrama de sequência - Responder Prova (actualizado)	42
7.15	Diagrama de sequência - Editar Questões	43
7.16	Diagrama de sequência - Criar Prova (actualizado)	44
8.1	Diagrama de <i>deployment</i> (contexto de desenvolvimento e teste)	45
8.2	Diagrama de <i>deployment</i> (contexto de produção)	46
9.1	Arquitetura do microsserviço, abordagem EDUF.	51
9.2	Arquitetura do microsserviço, implementação à data de 4 de janeiro, de 2024.	51

Listas de Tabelas

1.1	Requisitos funcionais basilares	9
2.1	Restrição quanto à infraestrutura informática	10
2.2	Restrição quanto ao isolamento da aplicação de resposta às provas	11
3.1	Requisitos não funcionais que impactam a escolha da arquitetura	12
5.1	Requisitos não funcionais que mais impactam a escolha da arquitetura	17
5.2	Modelo de dados - Utilizadores	19
5.3	Modelo de dados - Salas (actualizado)	19
5.4	<i>Schema</i> de um intervalo de tempo em que uma sala está ocupada	19
5.5	Modelo de dados - Prova	20
5.6	Modelo de dados - Versão	20
5.7	Modelo de dados - Questão	21
5.8	Modelo de dados - Opção	21
5.9	Modelo de dados - Resolução	21
5.10	Modelo de dados - Resposta	22
5.11	Modelo de dados - Notificações	22
5.12	Distribuição de elementos da equipa pelos microsserviços	23

1. Introdução e Objectivos

Este documento foi concebido no âmbito de uma proposta para desenvolver o sistema "PRO-BUM", que visa possibilitar a realização e gestão de exames universitários por meio de uma plataforma informática.

Nele, descrevem-se diversas etapas relacionadas com o planeamento da implementação do sistema. Inicialmente, abordamos o conjunto de restrições impostas à equipa de desenvolvimento, englobando limitações orçamentais, temporais e relacionadas com a implementação do sistema.

Seguidamente, procedemos à análise dos requisitos não funcionais presentes no documento de requisitos, selecionando aqueles com maior impacto na escolha da arquitetura do sistema a desenvolver.

Posteriormente, delimitamos o contexto do sistema, construindo uma decomposição funcional para compreender as fronteiras funcionais do sistema.

De seguida, estabelecemos a arquitetura que sustentará o sistema, decidindo-a com base na análise dos requisitos não funcionais e na sua devida priorização.

Por último, descrevemos detalhes relativos à implementação do sistema. Começamos por descrever os componentes do sistema e a forma como se comunicam entre si, bem como a implementação de todas as funcionalidades do sistema. Também abordamos outros aspetos relevantes, como o modo como o sistema será instalado para permitir a sua utilização.

Nota: Este documento é a continuação do documento da solução arquitetural entregue na fase anterior. Desta forma, tudo o que se manteve igual da fase anterior para a actual encontra-se a uma cor azulada e tudo o que foi actualizado/acrescentado encontra-se escrito a preto.

1.1 Resumo dos requisitos funcionais

Na fase inicial do desenvolvimento deste projeto, foi elaborado um documento de requisitos centrado no domínio do problema. Neste documento, foram definidas as funcionalidades do sistema em relação aos atores correspondentes, servindo como base para extrair os requisitos funcionais do sistema.

As funcionalidades descritas no documento de requisitos foram as seguintes:

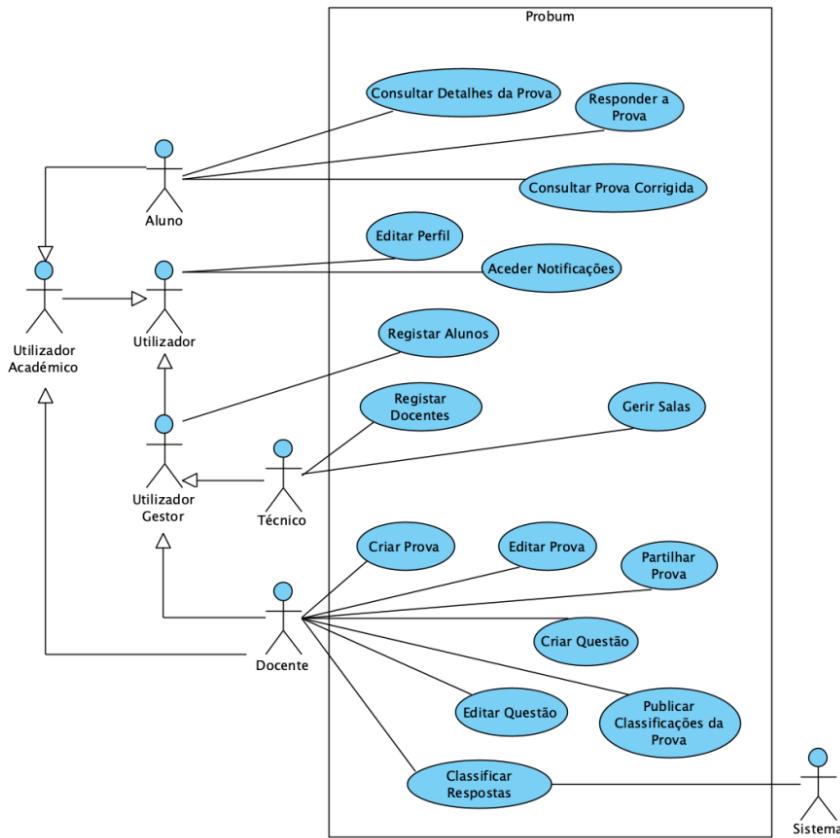


Figura 1.1: *Use cases* do sistema

Tendo em conta o sistema que se pretende desenvolver e o documento de requisitos fornecido, a equipa de desenvolvimento decidiu estabelecer um conjunto de requisitos funcionais que considera que são basilares para poder implementar, no mínimo, uma versão inicial do sistema.

Assim, estabeleceram-se os seguintes requisitos funcionais como os mais prioritários:

Requisito	Use cases	Descrição	Prioridade
Req14	Criar prova	O Docente cria uma prova de avaliação com os seguintes parâmetros: nome da prova, ficheiro com as informações dos alunos a participar na prova, data, hora preferencial, tempo de admissão e duração	Must
Req17	Criar prova e Editar prova	O Sistema faz uma calendarização inteligente, indicando salas, horários e alunos aloca-dos a cada sala	Must
Req18	Criar prova e Editar prova	O Docente aceita (ou rejeita) as propostas de calendarização, podendo alterar detalhes da proposta caso rejeite	Must
Req19	Criar prova e Editar Prova	O Docente inscreve Alunos numa prova de avaliação	Must

Requisito	Use cases	Descrição	Prioridade
Req22	Criar prova, Criar questões, Editar prova e Editar questões	O Docente adiciona questões de escolha múltipla a uma prova de avaliação	Must
Req26	Criar prova, Criar questões, Editar prova e Editar questões	O Docente adiciona um enunciado para cada questão	Must
Req50	Responder a prova	O Aluno seleciona a questão que pretende responder	Must
Req51	Responder a prova	O Aluno submete uma resposta	Must
Req52	Responder a prova	O Aluno termina a Prova a qualquer momento	Must
Req58	Classificar Respostas	O Sistema corrige automaticamente perguntas com resposta pré-definida por um Docente.	Must
Req59	Classificar Respostas	O Docente corrige manualmente uma Prova, atribuindo pontuação a cada resposta	Must
Req60	Publicar Classificações da Prova	O Docente publica as classificações de uma prova	Should
Req63	Consultar Prova Corrigida	O Aluno consulta a sua classificação de uma Prova	Must

Tabela 1.1: Requisitos funcionais basilares

2. Restrições

Nesta secção enunciamos as restrições impostas no documento de requisitos e analisamos o modo como estas irão impactar a nossa solução.

Restrições à Solução

Requisito #:	Rest1	Tipo:	Restrição
Descrição			A aplicação deve executar na infraestrutura atual da respetiva IES
Rationale			Para que não seja necessário investir em novo equipamento
Origem			Cliente
Fit criterion			Todos os componentes de software devem estar instalados em máquinas da IES e todas as funcionalidades da plataforma para os Alunos devem executar em pleno nas máquinas que forem disponibilizadas para as provas de avaliação
Prioridade			Must

Tabela 2.1: Restrição quanto à infraestrutura informática

Esta restrição requer que a equipa de desenvolvimento esteja ciente dos recursos computacionais disponibilizados pelos equipamentos da IES. Assim, o desenvolvimento do sistema precisa de ser concebido de modo a garantir a execução adequada do sistema nos atuais equipamentos da IES.

Requisito #: Rest2	Tipo: Restrição
Descrição	O computador disponibilizado a cada Aluno, no momento em que realiza uma prova de avaliação, apenas deve permitir acesso ao Probum
Rationale	Para evitar que o Aluno recorra a outras aplicações (email, navegadores web, WhatsApp, Skype, etc.) durante a realização da sua prova de avaliação; O produto tem que estar preparado para funcionar em computadores instalados em salas da IES
Origem	Cliente
Fit criterion	Enquanto uma prova de avaliação estiver a decorrer não deve ser possível aceder a nenhuma outra aplicação que não o Probum
Prioridade	Must

Tabela 2.2: Restrição quanto ao isolamento da aplicação de resposta às provas

Esta restrição implica que a implementação do sistema deverá englobar uma limitação ao acesso a todas as funcionalidades do computador atribuído ao aluno, exceto aquelas que são disponibilizadas pelo PROBUM.

Restrições Temporais

- **Descrição:** O documento presente terá de ser entregue até ao dia 1 de dezembro de 2023.
Justificação: De forma a poder ser avaliado o estado do projeto nesta fase intermédia, é necessário que seja feita uma entrega que contenha a segunda fase deste projeto, que consiste essencialmente na solução arquitetural do sistema a desenvolver.
- **Descrição:** A primeira versão do sistema deverá ser entregue até ao dia 22 de dezembro de 2023.
Justificação: De forma a fornecer à IES uma versão inicial do sistema, ainda com algumas funcionalidades não implementadas, para permitir o uso do PROBUM.

Restrições Orçamentais

- **Descrição:** O orçamento total para o desenvolvimento do projeto é de 20 000 € (vinte mil euros), durante um período de 4 meses.
Justificação: A equipa responsável pelo desenvolvimento do projeto é constituída por onze engenheiros de *software*. Para além de ter em conta os salários dos elementos, é preciso também a compra de um domínio, bem como de um serviço de hospedagem da aplicação.

3. Requisitos de qualidade/não funcionais

A equipa de desenvolvimento optou por escolher os requisitos não funcionais que considera ter mais impacto no sistema, e consequentemente, na escolha de uma solução arquitetural. A escolha arquitetural visa maximizar as características arquiteturais de maior prioridade.

Prioridade	IdReq	Descrição
1	RNF14	O sistema tem grande disponibilidade durante horário letivo
2	RNF13	O sistema deve ser escalável
3	RNF9	Qualquer interação entre o utilizador e o Sistema tem um tempo de resposta inferior a 2 segundos
4	RNF10	O sistema opera em modo local caso seja perdida a conexão com o servidor
5	RNF12	O sistema deve correr em computadores de gama baixa
6	RNF16	As alterações a uma prova deverão ser automaticamente guardadas a cada 20 segundos

Tabela 3.1: Requisitos não funcionais que impactam a escolha da arquitetura

Na priorização dos requisitos não funcionais levantados pela equipa de levantamento de requisitos, a equipa de desenvolvimento opta por dar prioridade àqueles que conferem a maior aproximação ao sistema de avaliação atual nas Instituições de Ensino Superior. Queremos com isto dizer, que são reunidos esforços e concentradas prioridades na eliminação de qualquer limitação tecnológica ao sistema de avaliação.

Assim sendo, a equipa destaca alguns dos requisitos, bem como as características arquiteturais mais impactantes na escolha da solução arquitetural.

Existem dois tipos de impacto direto no sistema relacionados com os requisitos não funcionais. A equipa de desenvolvimento classifica-os como sendo de uso individual e/ou coletivo. Ambos são refletidos aquando do uso do sistema, sendo que o individual não influencia o coletivo, ao passo que o coletivo influencia o uso individual de cada utilizador com o sistema.

Em termos de impactos individuais do sistema, destaca-se a priorização da resistência da avaliação em caso de uma falha ao nível da rede. A mesma criaria transtorno para os avaliadores, para os avaliados, e consequentemente para a Instituição de Ensino em questão.

No que respeita às limitações físicas do computador, quer de cada aluno, quer da Instituição de Ensino em causa, onde será realizada a prova, a equipa de desenvolvimento assume o compro-

missos de que a utilização do sistema não será comprometida em computadores de gama baixa (ao ano de 2018).

Ao nível da utilização coletiva do sistema, destacamos a priorização dos requisitos não funcionais que tenham a ver com o uso massivo da aplicação (quer no mesmo instante de tempo, quer ao longo do tempo). Assim, a equipa de desenvolvimento destaca a escalabilidade do sistema, necessária para dar resposta a elevadas cargas de trabalho geradas pela realização das provas (exames a serem feitos ao mesmo tempo, e por uma grande quantidade de alunos), bem como, a elevada quantidade de dados gerada ao longo do uso do sistema pelas Instituições de Ensino, nomeadamente pela criação, realização e arquivo de provas.

4. Contexto e âmbito do sistema

O sistema proposto, denominado Probum, surge como uma solução inovadora para facilitar a realização de provas académicas por alunos de instituições de ensino superior (IES).

O contexto que motiva o desenvolvimento do Probum está fundamentado na necessidade de superar as limitações das infraestruturas informáticas existentes nalgumas IES, que podem ser restritas em termos de dimensão, disponibilidade e capacidade.

O Probum tem como objetivo principal permitir que os alunos de uma determinada unidade curricular possam realizar as suas provas académicas utilizando os recursos informáticos da sua própria instituição de ensino superior.

Este conceito abre caminho para a flexibilidade e acessibilidade, independentemente das condições específicas de cada IES. A proposta do Probum é adaptar-se a diferentes ambientes, promovendo uma utilização eficiente mesmo em instituições com recursos computacionais mais limitados.

De acordo com os *use cases* e os requisitos funcionais estabelecidos no documento de requisitos, foi possível elaborar o seguinte diagrama de contexto do sistema:

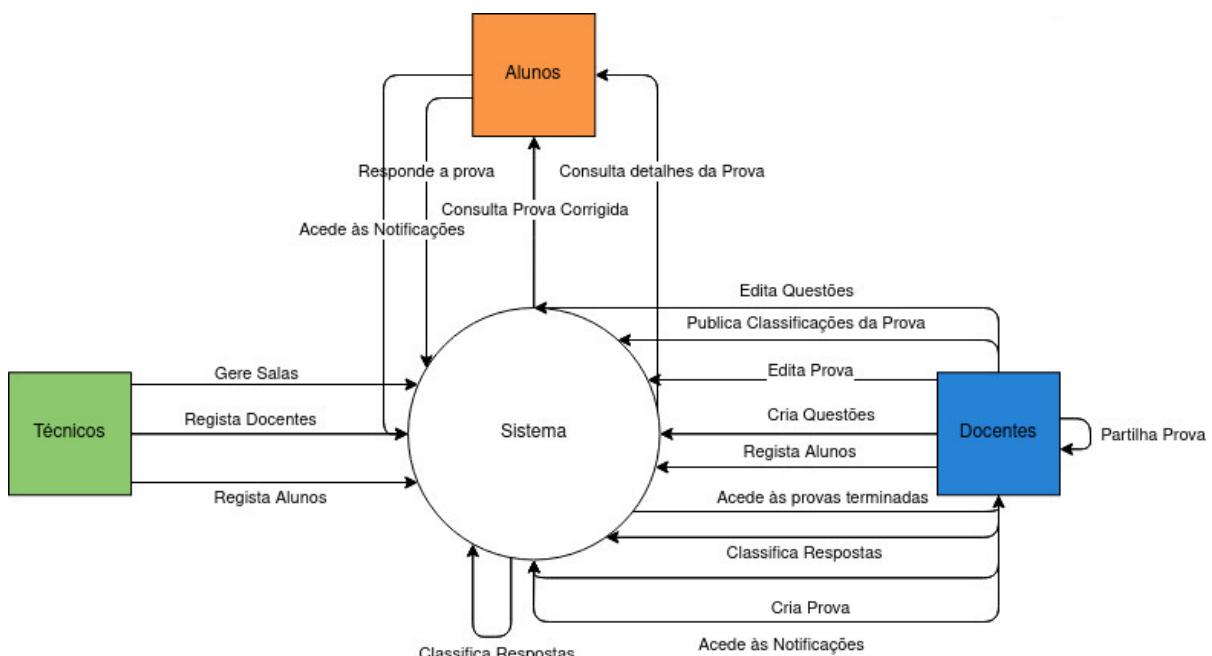


Figura 4.1: Diagrama de contexto (*business*)

No diagrama apresentado, temos como objetivo ilustrar as interações entre o sistema e os atores envolvidos. Este diagrama procura contribuir para a definição inequívoca dos limites do sistema, delimitando o que se insere no âmbito do projeto e estabelecendo as responsabilidades e funcionalidades do sistema de forma clara. Para além disso, é possível inferir que cada utilizador possui funcionalidades distintas. A título de exemplo, um técnico não deve ter a capacidade de responder a uma prova, da mesma forma que um docente não pode registar outro docente.

5. Estratégia da solução

Nesta secção, iremos abordar a decomposição funcional do sistema, tratar da escolha do padrão arquitetural com base na seleção dos requisitos não funcionais efetuada na secção 3 deste documento, definir o modelo de dados do sistema, bem como as tecnologias que serão utilizadas na implementação. Para além disso, também iremos fornecer uma descrição da organização da equipa de desenvolvimento para os diversos componentes do sistema a desenvolver.

5.1 Decomposição funcional

O PROBUM é um sistema de software com uma elevada complexidade. Por este motivo, a equipa de desenvolvimento elaborou uma decomposição funcional descrita num diagrama de blocos em que fazemos uma divisão das responsabilidades do sistema a vários níveis de profundidade, simplificando a sua complexidade, o que permite ter uma melhor compreensão do sistema de forma a facilitar o seu desenvolvimento.

A elaboração da decomposição funcional consistiu numa análise dos requisitos funcionais e quais os elementos que eles manipulam ao nível dos dados, bem como os *use cases* a que eles estão associados.

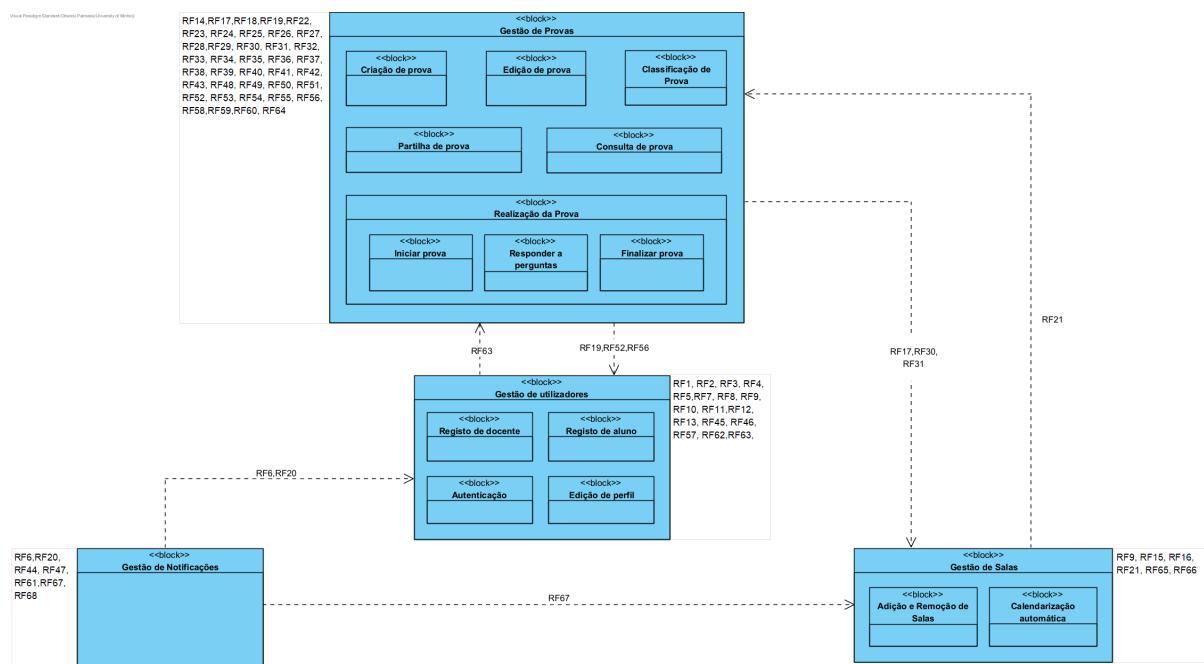


Figura 5.1: Diagrama de blocos - decomposição funcional

Como podemos observar no diagrama acima, a decomposição funcional iniciou-se com o agrupamento dos requisitos funcionais em quatro grandes blocos referentes à gestão das entidades

principais do sistema: as provas, salas, notificações e utilizadores.

Ao descer mais um nível na decomposição, podemos observar a existência de blocos funcionais um pouco mais próximos aos *use cases* (p.e. Criação de prova, Edição de prova, Edição de perfil, etc.)

O bloco funcional "Realização de prova" ainda foi subdividido pelos blocos "Iniciar Prova", "Responder a perguntas" e "Finalizar prova". Nestes três blocos, a inicialização e finalização da prova será feita automaticamente pelo sistema de acordo com os horários das provas e a resposta a perguntas será feita pelos alunos.

A partir do diagrama da figura 5.1, podemos constatar que uma grande parte da complexidade do sistema concentra-se nas provas, pelo que haverá uma intensificação dos esforços da equipa de desenvolvimento na implementação das funcionalidades relativas às provas. Para além disso, são evidenciadas relações que os requisitos funcionais estabelecem entre os blocos funcionais, o que pode indicar as possíveis dependências que existirão entre as responsabilidades de cada bloco funcional na implementação concreta do sistema.

5.2 Padrão arquitetural

A equipa de desenvolvimento considerou que, para este sistema, os melhores padrões arquiteturais a considerar são os Microsserviços e arquitetura em camadas (*layered*). Por um lado, a arquitetura em camadas promove uma organização estruturada e organizada da aplicação. Por outro lado, a arquitetura de microsserviços promove a escalabilidade, resiliência e utilização de tecnologias diversificadas que melhor se ajustam ao contexto de cada microsserviço.

Os requisitos não funcionais que a equipa de desenvolvimento considerou que mais influenciam a escolha da arquitetura do sistema (ver secção 3) são os seguintes (os requisitos estão ordenados por ordem de prioridade, sendo o primeiro o mais prioritário):

Solução arquitetural				
IdReq	Descrição	Característica arquitetural	Microserviços	Layered
RNF14	O sistema tem grande disponibilidade durante horário letivo	Scalability	★★★★★	★
RNF13	O sistema deve ser escalável	Scalability	★★★★★	★
RNF9	Qualquer interação entre o utilizador e o Sistema tem um tempo de resposta inferior a 2 segundos	Performance	★★	★★
RNF10	O sistema opera em modo local caso seja perdida a conexão com o servidor	Reliability	★★★★	★★★
RNF12	O sistema deve correr em computadores de gama baixa	Performance	★★	★★
RNF16	As alterações a uma prova devem ser automaticamente guardadas a cada 20 segundos	Reliability	★★★★	★★★

Tabela 5.1: Requisitos não funcionais que mais impactam a escolha da arquitetura

A elaboração da tabela acima foi feita com base nos seguintes *ratings* expostos no livro "The Fundamentals of Software Architecture: An Engineering Approach" de Mark Richards e Neal Ford:

Architecture characteristic	Star rating
Partitioning type	Domain
Number of quanta	1 to many
Deployability	★★★★★
Elasticity	★★★★★
Evolutionary	★★★★★
Fault tolerance	★★★★★
Modularity	★★★★★
Overall cost	★
Performance	★★
Reliability	★★★★
Scalability	★★★★★
Simplicity	★
Testability	★★★★★

(a) Ratings - Microserviços

Architecture characteristic	Star rating
Partitioning type	Technical
Number of quanta	1
Deployability	★
Elasticity	★
Evolutionary	★
Fault tolerance	★
Modularity	★
Overall cost	★★★★★
Performance	★★
Reliability	★★★
Scalability	★
Simplicity	★★★★★
Testability	★★

(b) Ratings - Layered

De acordo com a análise de requisitos não funcionais na tabela 5.1 e com a análise da escolha desses requisitos apresentada na secção 3 deste documento, a equipa de desenvolvimento estabeleceu que o padrão arquitetural adotado será **microsserviços**.

5.2.1 Identificação de microsserviços

As funcionalidades principais do PROBUM assentam essencialmente nos seguintes pontos:

- Realização de provas
- Classificação de provas
- Calendarização das provas baseada na disponibilidade das salas.
- Notificação dos utilizadores acerca da ocorrência de eventos que lhes são relevantes.
- Registo dos principais utilizadores (Docentes e Alunos) por parte dos técnicos do sistema.

Com os pontos mencionados acima, podemos identificar quatro entidades principais do sistema: Provas (elemento nuclear do sistema no qual todas as funcionalidades se baseiam), Salas (influenciam diretamente a calendarização das provas), Notificações (essencial para notificar os utilizadores de eventos relevantes, nomeadamente a publicação de classificações de provas) e Utilizadores (tal como na maioria dos sistemas de software, os utilizadores estão sempre presentes).

Desta forma, a equipa de desenvolvimento decidiu estabelecer os seguintes microsserviços: Gestão de utilizadores, Gestão de salas, Gestão de notificações e Gestão de provas.

Os quatro microsserviços estabelecidos terão as seguintes responsabilidades:

- Gestão de provas
 - Criação de provas
 - Edição de provas
 - Realização de provas
 - Classificação automática de provas
 - Partilha de provas entre docentes
- Gestão de salas
 - Criação/remoção de salas
 - Calendarização inteligente das provas
- Gestão de notificações
 - Notificação de eventos aos utilizadores
- Gestão de utilizadores
 - Autenticação
 - Registo de utilizadores
 - Alteração de dados de utilizadores.

Os microsserviços estabelecidos consistem nos quatro maiores blocos funcionais. No entanto, ainda se considerou utilizar alguns blocos funcionais mais granulares como microsserviços (por exemplo, a Calendarização Automática). Contudo, verificámos que muito provavelmente esses microsserviços mais pequenos teriam que aceder à mesma base de dados, o que quebra as regras do padrão arquitetural de microsserviços.

5.3 Modelos de dados das bases de dados dos microsserviços

Nesta secção iremos apresentar, para cada um dos microsserviços estabelecidos, os dicionários de dados de cada uma das bases de dados dos microsserviços.

5.3.1 Gestão de Utilizadores

Para a gestão de utilizadores, iremos utilizar uma base de dados **MongoDB** em que teremos uma coleção de utilizadores (alunos, docentes e técnicos).

Nome do campo	Tipo	Utilização
_id	ObjectID	Identificação do utilizador na base de dados.
numMecanografico	String	Identificador do utilizador no sistema
email	String	Email do utilizador
name	String	Nome completo do utilizador
password	String	Palavra-passe do utilizador
type	String	Tipo de utilizador (aluno, docente ou técnico)

Tabela 5.2: Modelo de dados - Utilizadores

5.3.2 Gestão de Salas

Para a gestão de salas, iremos utilizar uma base de dados **MongoDB** em que teremos uma coleção para as salas.

Nome do campo	Tipo	Utilização
_id	ObjectID	Identificação da sala na base de dados.
edificio	String	Identificação do edifício a que pertence.
numSala	String	Identificação da sala no sistema
piso	String	Piso da sala no edifício
capacidade	Integer	Capacidade da sala.
ocupacao	[ocupacaoSchema]	Intervalos de tempo de ocupação das salas.

Tabela 5.3: Modelo de dados - Salas (actualizado)

Cada sala terá uma lista (campo "ocupacao") que armazena os horários em que a sala estará ocupada para a realização de uma prova:

Nome do campo	Tipo	Utilização
dataHoraInicio	DateTime	Hora de início da ocupação da sala.
dataHoraFim	DateTime	Hora de fim da ocupação da sala.

Tabela 5.4: Schema de um intervalo de tempo em que uma sala está ocupada

5.3.3 Gestão de Provas

Para a gestão de provas, iremos utilizar uma base de dados **MongoDB**, na qual a entidade principal é a Prova. Esta, por sua vez, contém informação sobre as versões da mesma. O campo Questões está incluído na entidade Versão e contém informação sobre uma Questão específica. Finalmente, a entidade Opção refere-se a uma opção para colocar numa questão de escolha múltipla ou Verdadeiro/Falso.

No que diz respeito às resoluções dos alunos, teremos duas entidades: a Resposta, que guarda a resposta dada por um aluno a uma certa questão e a Resolução que é o conjunto de respostas que o aluno deu numa certa prova.

Provas

Nome do campo	Tipo	Utilização
.id	ObjectID	Identificação da prova na base de dados.
nome	String	Identificação da prova no sistema.
docentes	[String]	Lista de identificadores dos docentes que elaboraram a prova
unidadeCurricular	String	Nome da unidade curricular associada à prova
retrocesso	Boolean	Indica se o retrocesso nas questões é permitido
aleatorização	Boolean	Indica se a ordem das questões será aleatorizada.
versões	[Versao]	Lista de todas as versões da prova.

Tabela 5.5: Modelo de dados - Prova

Nome do campo	Tipo	Utilização
id	Integer	Identificador da versão na base de dados.
numVersao	Integer	Número identificativo da versão da prova.
alunos	[String]	Lista dos identificadores dos alunos que estarão associados à versão da prova.
sala	String	Identificador da sala onde a versão da prova será realizada.
data	Datetime	Data e hora da realização da versão da prova.
questões	[Questão]	Lista de questões da versão da prova.

Tabela 5.6: Modelo de dados - Versão

Nome do campo	Tipo	Utilização
id	Integer	Identificador da questão na base de dados.
descrição	String	Descrição da questão (enunciado).
imagem (opcional)	Bytes	Imagem da questão.
tipo	Integer	Indica o tipo de questão (V/F, escolha múltipla, etc.).
cotação	Integer	Cotação máxima da pergunta (valor entre 0 e 200)
desconto	Integer	Cotação descontada em caso de resposta errada (valor entre 0 e 200)
opções (opcional)	[Opção]	Opções da questão (apenas para escolha múltipla e V/F)

Tabela 5.7: Modelo de dados - Questão

Nome do campo	Tipo	Utilização
id	Integer	Identificador da opção na base de dados.
texto	String	Texto da opção (escolha múltipla ou V/F)
correta	Boolean	Indica se a opção é correta ou não.

Tabela 5.8: Modelo de dados - Opção

Resoluções dos alunos

Nome do campo	Tipo	Utilização
_id	ObjectID	Identificação da resolução do aluno na base de dados.
idAdluno	String	Identificador do aluno que elaborou a resolução da prova.
idProva	String	Identificador da prova a que corresponde a resolução.
idVersao	String	Identificador da versão da prova a que corresponde a resolução.
respostas	[Resposta]	Lista de respostas dadas pelo aluno.

Tabela 5.9: Modelo de dados - Resolução

Nome do campo	Tipo	Utilização
idQuestao	Integer	Identificador da questão respondida.
cotação	Integer	Cotação atribuída pelo docente ou pelo sistema (valor entre 0 e 200)
respostaAberta (opcional)	String	Resposta aberta dada pelo aluno
opçõesEscolhidas (opcional)	[Integer]	Identificadores das selecionadas ou marcadas como verdadeiras pelo aluno.

Tabela 5.10: Modelo de dados - Resposta

5.3.4 Gestão de Notificações

Para a gestão de notificações, iremos utilizar uma base de dados **MongoDB** em que teremos uma coleção para as notificações.

Nome do campo	Tipo	Utilização
_id	ObjectID	Identificação da notificação na base de dados.
notificacao	String	Tipo de notificação.
numero	String	Id de uma sala ou mecanografia de um utilizador.
email	String	Email do utilizador.
nome	String	Nome do utilizador.
lida	Boolean	Indicador de leitura: sim ou não.
prova	String	(Opcional) Nome da prova.
sala	String	(Opcional) Nome da sala.
data	String	(Opcional) Data.
hora	String	(Opcional) Hora.

Tabela 5.11: Modelo de dados - Notificações

5.4 Tecnologias para a implementação

Nesta secção iremos descrever as tecnologias selecionadas pela equipa de desenvolvimento para implementar o sistema.

5.4.1 Microsserviços e *API Gateway*

Nas aplicações dos microsserviços será utilizada a linguagem de programação **Javascript** juntamente com o ambiente de execução **Node.js** e a *framework* **Express** que facilita o processo de criação de API's REST. Para a comunicação com o servidor da base de dados, iremos utilizar a biblioteca **mongoose** para a conexão com as bases de dados MongoDB.

No que diz respeito à *API Gateway*, esta irá utilizar também **Javascript** com **Node.js** e **Express**. No entanto, não terá a necessidade de nenhuma biblioteca de base de dados.

5.4.2 Bases de dados

As equipas de cada microsserviço, após a conceção relativa a cada um dos seus microsserviços, conclui que a tecnologia a utilizar, relacionada com a persistência de dados, e fazendo uso de um Sistema de Gestão de Bases de Dados (SGBD), por todos, será o **MongoDB**, um sistema

de base de dados não relacional orientado a documentos.

Os motivos que mais impactaram a adoção deste SGBD, e consequente modelo de dados não relacional tem a ver com (1) a facilidade de relacionamentos implícitos face a todas as restrições de integridade, e relacionais, explícitas, impostas por um modelo de dados relacional.

Não obstante de uma abordagem concecional mais aprofundada relativamente às entidades e relações inerentes em cada um dos microsserviços (envergando por um modelo de dados relacional), e posteriormente materializadas em mais do que uma tabela, (2) existe toda uma flexibilidade envolvente aquando da adoção de uma base de dados orientada a documentos: esta flexibilidade é materializada no facto de se poderem armazenar diferentes documentos (de respeito à mesma entidade) numa única coleção.

5.4.3 Frontend

Para o desenvolvimento do *frontend*, iremos utilizar a linguagem **Javascript** com o auxílio da biblioteca **React** para a componente interativa da interface do utilizador. Para a componente visual, iremos utilizar o **Tailwind CSS**.

5.5 Decisões organizacionais

A equipa de desenvolvimento é composta por 11 elementos e decidiu-se que todos os elementos devem ser distribuídos pelos quatro microsserviços. No entanto, esta distribuição não será uniforme, uma vez que existem microsserviços com maior complexidade, nomeadamente a Gestão de provas.

Assim sendo, estabeleceu-se a seguinte distribuição de elementos pelos microsserviços:

Microsserviço	Número de elementos
Gestão de provas	4
Gestão de utilizadores	3
Gestão de salas	2
Gestão de notificações	2

Tabela 5.12: Distribuição de elementos da equipa pelos microsserviços

No que diz respeito ao *frontend* e à *API Gateway*, cada equipa de cada microsserviço irá desenvolver a sua parte destes dois componentes do sistema (por exemplo, a equipa do microsserviço de gestão de provas ficará responsabilizada por desenvolver toda a parte do *frontend* que corresponde às funcionalidades relativas a esse microsserviço), uma vez que têm uma noção mais precisa do contexto do seu microsserviço e o modo como este está implementado.

6. Building Block View

Nesta secção iremos abordar os diversos componentes que compõem o sistema. Neste primeiro diagrama apresentamos uma visão global do sistema, que engloba todos os microsserviços, a *API Gateway* e o *frontend* do cliente. O objetivo principal deste diagrama é apresentar os componentes principais do sistema, bem como as interfaces fornecidas e requeridas de cada um que demonstram como estes comunicam entre si. Nos diagramas seguintes iremos detalhar cada componente deste diagrama, nomeadamente os componentes dos microsserviços.

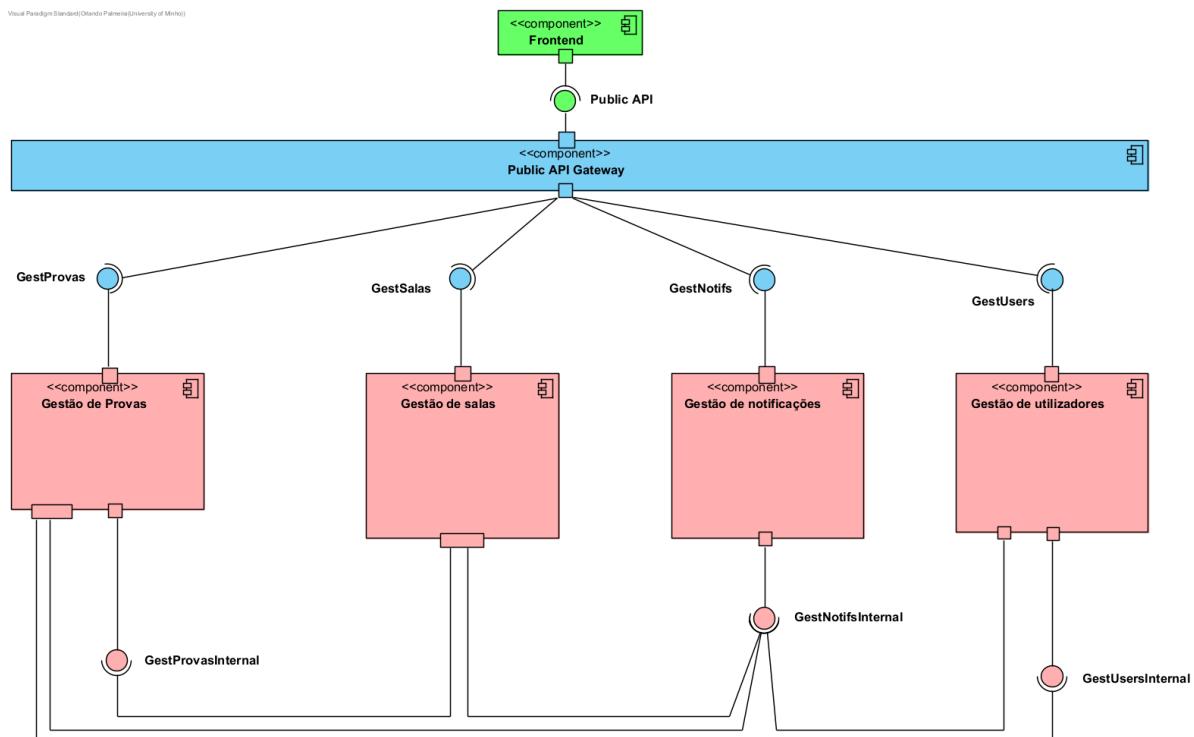


Figura 6.1: Diagrama de componentes

6.1 Gestão de utilizadores

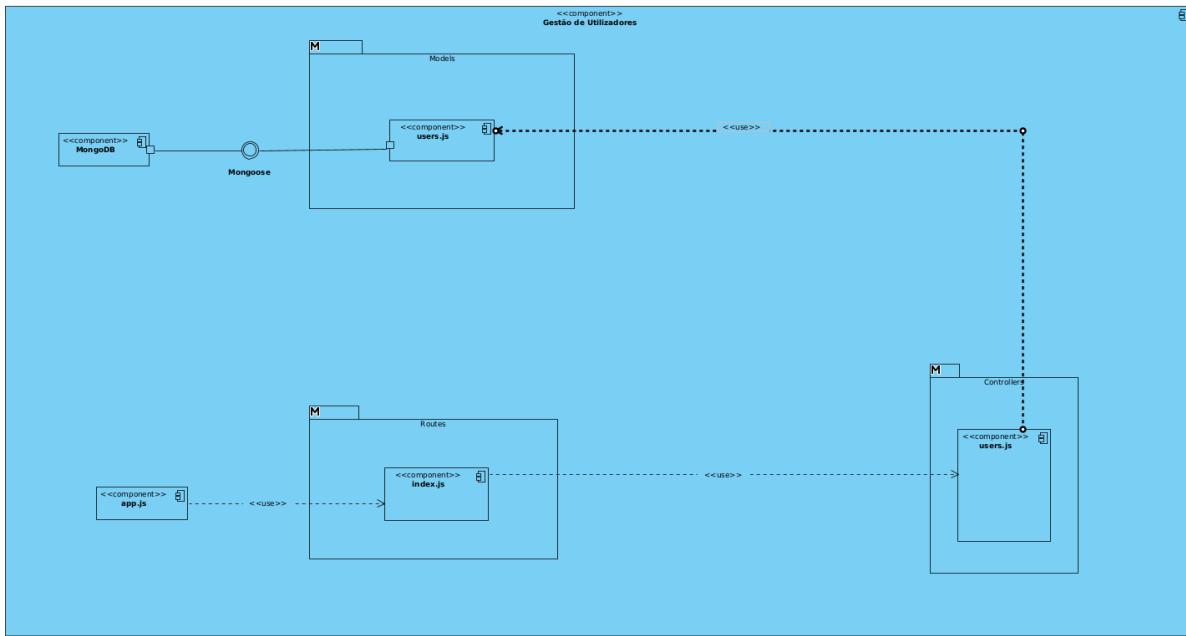


Figura 6.2: Diagrama de componentes relativo à gestão de utilizadores

Neste diagrama, podemos observar a presença de quatro *packages*.

Os *packages* são: (1) *Routes*, (2) *Controllers*, (3) *Models* e Autenticação.

Os componentes existentes, no total, são seis, sendo eles: (1) a componente da base de dados (`MongoDB`), (2) a componente `"users.js"`, inserida no *package* dos *Models*, (3) a componente `"users.js"`, inserida no *package* dos *Controllers*, (4) a componente `"index.js"`, inserida no *package* das *Routes*, (5) a componente `"app.js"`, e (6) a componente `"Authentication Module"`, referente à autenticação do Utilizador.

No *package Models* estará inserido o *schema* relativo a alunos, docentes e técnicos, e assim sendo, será possível realizar todas as respetivas *queries* que regem o microsserviço. As *queries* realizar-se-ão através da biblioteca `mongoose`. A biblioteca é responsável pela ligação entre o `MongoDB` e os *Models*.

No *package* dos *Controllers*, mais especificamente no componente `"users.js"`, irão estar presentes os métodos necessários para a implementação das rotas.

As rotas estarão presentes no ficheiro `"index.js"`, pertencente ao *package* das *Routes*.

No *package* das *Routes* irá constar a implementação das rotas que o microsserviço suporta. As mesmas irão ser utilizadas pelo ficheiro `"app.js"`. Este ficheiro destina-se ao arranque do microsserviço.

No módulo de autenticação, presente no *package* `"Autenticação"`, encontrar-se-á ainda a implementação de todas as rotas necessárias à autenticação do cliente.

6.2 Gestão de salas

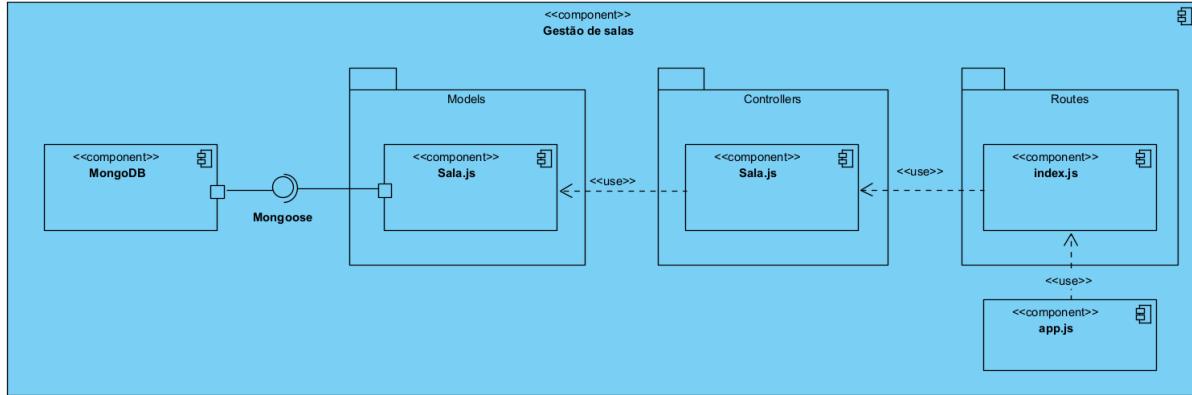


Figura 6.3: Diagrama de componentes relativo à gestão de salas

Neste diagrama, podemos observar a presença de três *packages*.

Os *packages* são: (1) *Routes*, (2) *Controllers*, e (3) *Models*.

Cada *package* engloba um componente, sendo que são cinco os componentes existentes no total, são eles: (1) a componente da base de dados (*MongoDB*), (2) a componente "Sala.js", inserida no *package* dos *Models*, (3) a componente "Sala.js", inserida no *package* dos *Controllers*, (4) a componente "index.js", inserida no *package* das *Routes*, (5) e a componente "app.js".

No *package Models* vai estar inserido o *schema* relativo a uma sala, e assim, realizar-se-ão as *queries* respetivas, através da biblioteca *mongoose*. A biblioteca é responsável pela ligação entre o *MongoDB* e os *Models*.

No *package* dos *Controllers*, mais especificamente no componente "Sala.js", irão estar presentes os métodos necessários para a implementação das rotas.

As rotas destinar-se-ão a estar presentes no ficheiro "index.js", pertencente ao *package* das *Routes*.

Por fim, no *package* das *Routes* irão estar implementadas as rotas que o microsserviço suporta. As mesmas irão ser usadas pelo ficheiro "app.js", destinando-se ao arranque do microsserviço.

6.3 Gestão de provas

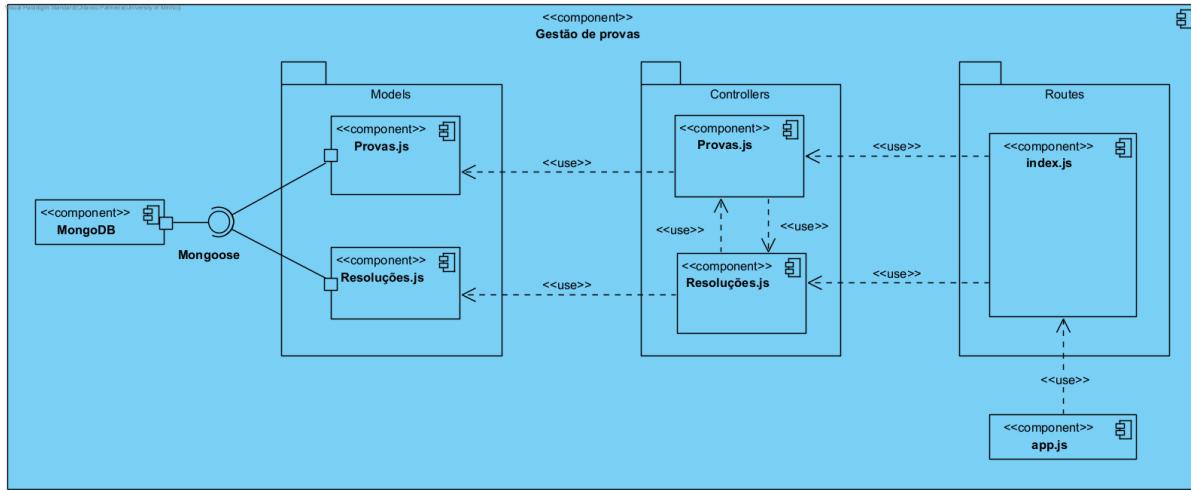


Figura 6.4: Diagrama de componentes relativo à gestão de provas

Neste diagrama, podemos observar a presença de três *packages*.

Os *packages* são: (1) *Routes*, (2) *Controllers*, e (3) *Models*.

Cada um deles possui certos componentes, sendo que são sete os componentes existentes no total: (1) a componente da base de dados (*MongoDB*), (2) as componentes "Provas.js" e (3) "Resoluções.js", inseridas no *package* dos *Models*, (4) as componentes "Provas.js" e (5)"Resoluções.js", inseridas no *package* dos *Controllers*, (6) a componente "index.js", inserida no *package* das *Routes*, (7) e a componente "app.js".

No *package Models* vão estar inseridos os *schemas* relativos às provas e resoluções de provas, e assim, realizar-se-ão as *queries* respetivas, através da biblioteca *mongoose*. A biblioteca é responsável pela ligação entre o *MongoDB* e os *Models*.

No *package* dos *Controllers*, mais especificamente nos componentes "Prova.js" e "Resoluções.js", irão estar presentes os métodos necessários para a implementação das rotas.

As rotas destinar-se-ão a estar presentes no ficheiro "index.js", pertencente ao *package* das *Routes*.

Por fim, no *package* das *Routes* irão estar implementadas as rotas que o microsserviço suporta. As mesmas serão usadas pelo ficheiro "app.js", que se destina ao arranque do microsserviço.

No planeamento na fase anterior, apenas o *controller* das Resoluções dos alunos era dependente do controller das provas. No entanto, devido à funcionalidade de consultar as provas ainda não realizadas, foi necessário criar uma dependência do *controller* de provas relativamente ao de resoluções de modo a saber quais as provas realizadas por um aluno.

6.4 Gestão de notificações

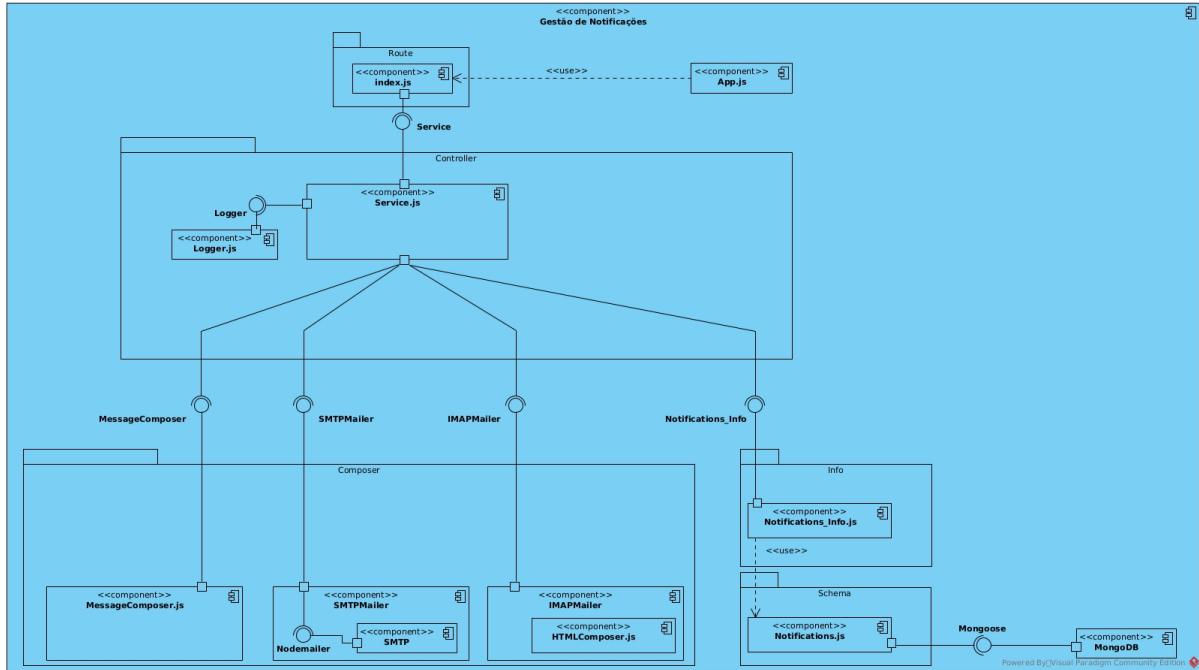


Figura 6.5: Diagrama de componentes relativo à gestão de notificações

Supra explicitamos o diagrama de componentes referente ao microsserviço das Notificações do sistema PROBUM.

O serviço de notificações do sistema assenta a sua conceção, e posterior desenvolvimento, numa arquitetura por camadas.

Para a comunicação com o exterior (entrada e saída de dados), a equipa de desenvolvimento cria um *package* designado por "Route".

É através deste *package* que, *a priori*, a ingestão de dados ocorre no sistema, não obstante, da adoção de posteriores mecanismos que permitam tolerância a falhas, nomeadamente, na ingestão, gravação e posterior envio de notificações para o utilizador (via *e-mail*).

Aquando da chegada de um pedido numa determinada rota, os dados são transacionados para o interior do sistema através de um controlador, pertencente ao *package Composer*, denominado por "Service.js".

Este controlador é responsável por orquestrar operações mediante a necessidade da rota.

As operações dizem respeito ao armazenamento de dados, ao seu tratamento, bem como, ao envio de uma notificação (por *e-mail*) ao utilizador em causa.

O controlador "Service.js" utiliza ainda um componente auxiliar, o "Logger.js", que efetua o registo das operações inerentes ao microsserviço: (1) a operação, (2) a data e a hora, (3) o utilizador em causa, e (4) a indicação do sucesso, ou insucesso, da operação.

O *package Composer* é concebido no âmbito de divisão e conquista.

Aqui, encontram-se divididas as funcionalidades *per se* do microsserviço: (1) compor uma mensagem de notificação, (2) enviar a mensagem para o respetivo *e-mail* do utilizador, e (3) leitura de *e-mails* através da plataforma do próprio utilizador.

O *package Info* fica responsável pela gestão de componentes que digam respeito à estruturação organizacional dos dados através de *queries*: simples e compostas.

Os componentes presentes neste *package*, tratando-se de um único componente até ao momento, para além de se encarregarem da filtragem aos documentos retirados da base de dados, ficam também encarregues do envio de dados para inserção no *schema* "Notifications" (única à data da presente documentação).

O *package Schema* respeita a coleção de dados inerentes à funcionalidade do sistema de notificações como um todo. Os dados são transformados nos respetivos documentos e alocados a cada *schema* através do recurso à biblioteca *mongoose*.

O *package* tem ainda a responsabilidade da extração dos documentos correspondentes a cada *schema*.

7. Runtime View

Nesta secção, expomos o funcionamento da solução proposta por meio de diagramas de sequência dos *use cases* do sistema. Estes diagramas abrangem uma diversidade de cenários, exibindo as interações entre os actores e os inúmeros componentes do sistema, bem como situações de excepção ou erro e a maneira como o sistema as aborda.

7.1 Diagramas de sequência

Os diagramas de sequência a seguir apresentados descrevem detalhadamente o funcionamento do sistema, apresentando cada interacção realizada entre os componentes do sistema, nomeadamente as interacções entre os microsserviços propostos na solução. Para além disso, apresentam o modo de funcionamento do *frontend* para os utilizadores do sistema.

7.1.1 Registar docentes

Este *use case* descreve a funcionalidade de um técnico registar um docente na plataforma do PROBUM. São fornecidas duas alternativas de registo: (i) registo individual em que o técnico insere manualmente os dados do docente num formulário para o efeito ou (ii) registo por ficheiro em que é fornecido ao sistema um ficheiro com os dados de vários docentes. Em ambas as alternativas existe uma verificação dos dados inseridos, em que o sistema impede a submissão de dados corrompidos e/ou repetidos. Se os registos forem feitos com sucesso, os docentes registados receberão um e-mail com os dados das suas contas do PROBUM.

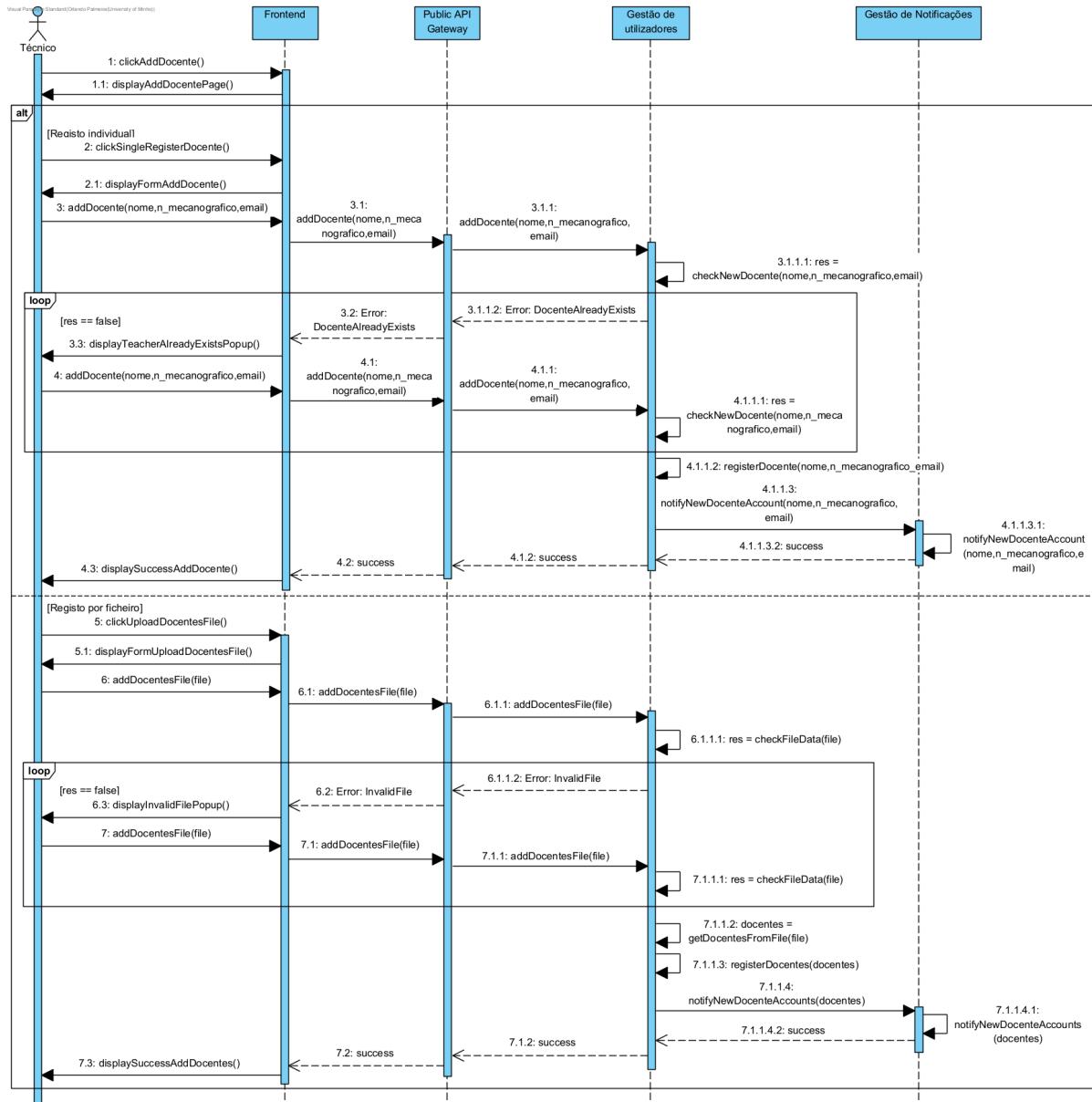


Figura 7.1: Diagrama de sequência - Registar docentes

7.1.2 Consultar prova corrigida

Este *use case* descreve a funcionalidade em que um aluno verifica os detalhes de uma prova cuja classificação já foi divulgada pelo docente responsável. Esta funcionalidade revela-se bastante directa, visto que se limita a uma mera consulta da correção de uma prova, o que implica um reduzido número de interações com a *API Gateway* e os micro-serviços.

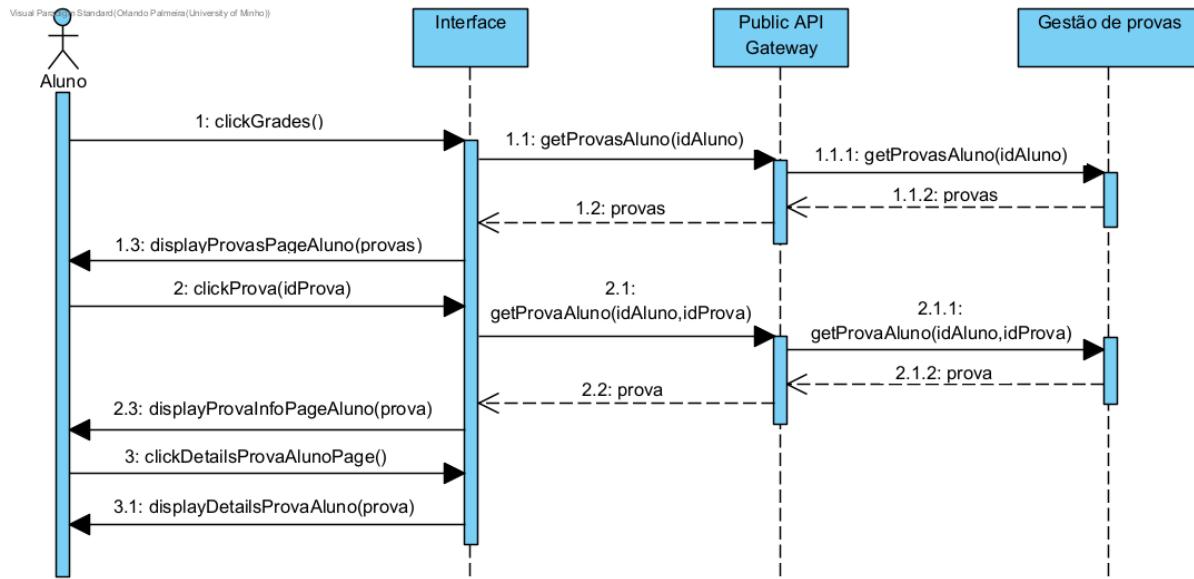


Figura 7.2: Diagrama de sequência - Consultar prova corrigida

7.1.3 Gerir salas

Este *use case* descreve a funcionalidade em que um técnico realiza a inserção ou remoção de salas no sistema. O diagrama de sequência evidencia alguma complexidade nas interacções entre os micro-serviços, especialmente no processo de remoção de salas. Esta complexidade decorre da necessidade do sistema em notificar os docentes responsáveis por uma prova que foi atribuída a uma sala que será eliminada.

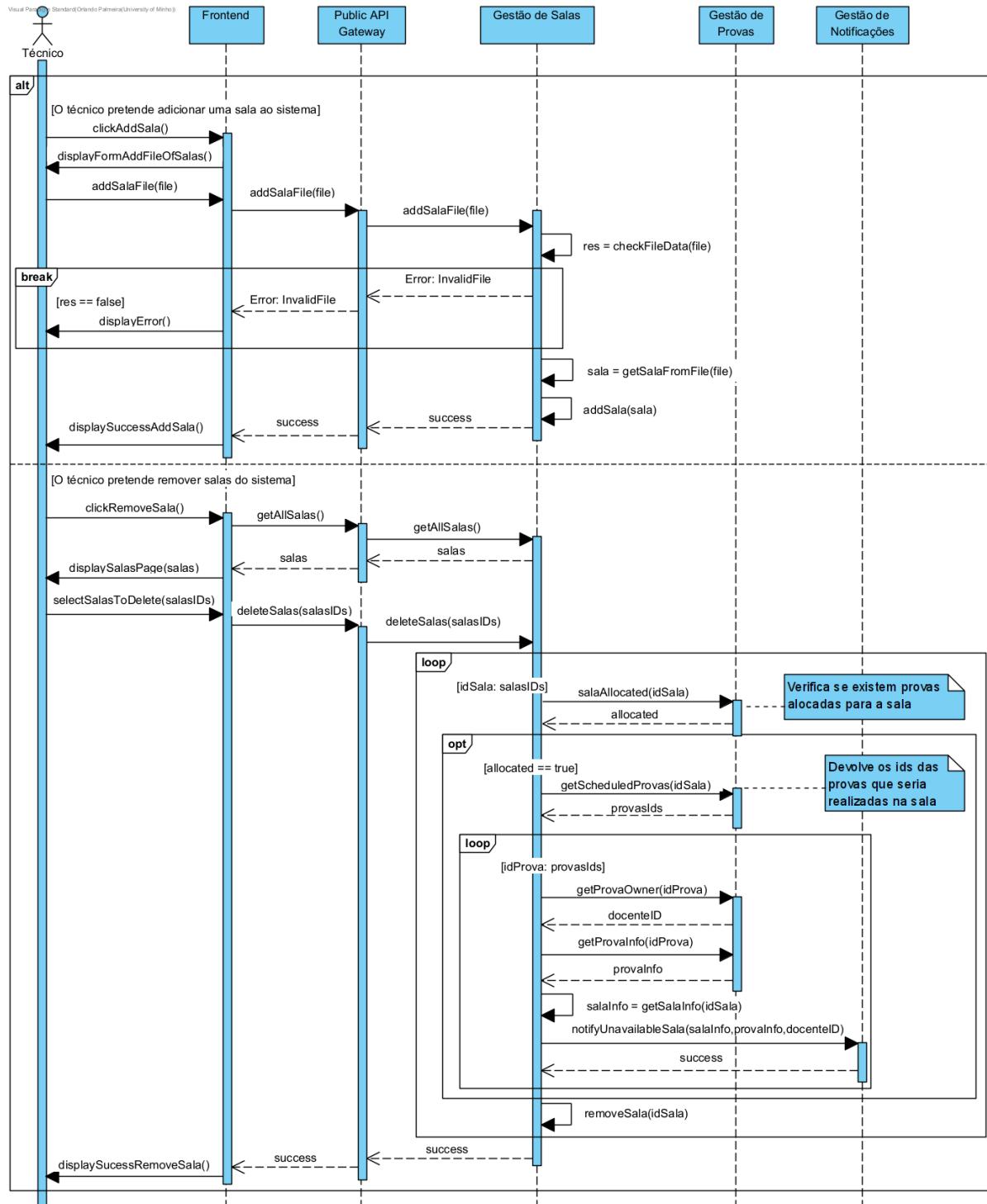


Figura 7.3: Diagrama de sequência - Gerir salas

7.1.4 Publicar classificações da prova

Este *use case* descreve a funcionalidade em que um docente publica as classificações de uma prova. No diagrama de sequência, está contemplada a verificação de que todos os alunos que realizaram a prova já foram classificados. No final, todos os alunos serão notificados sobre a publicação das classificações por meio do micro-serviço de gestão de notificações.

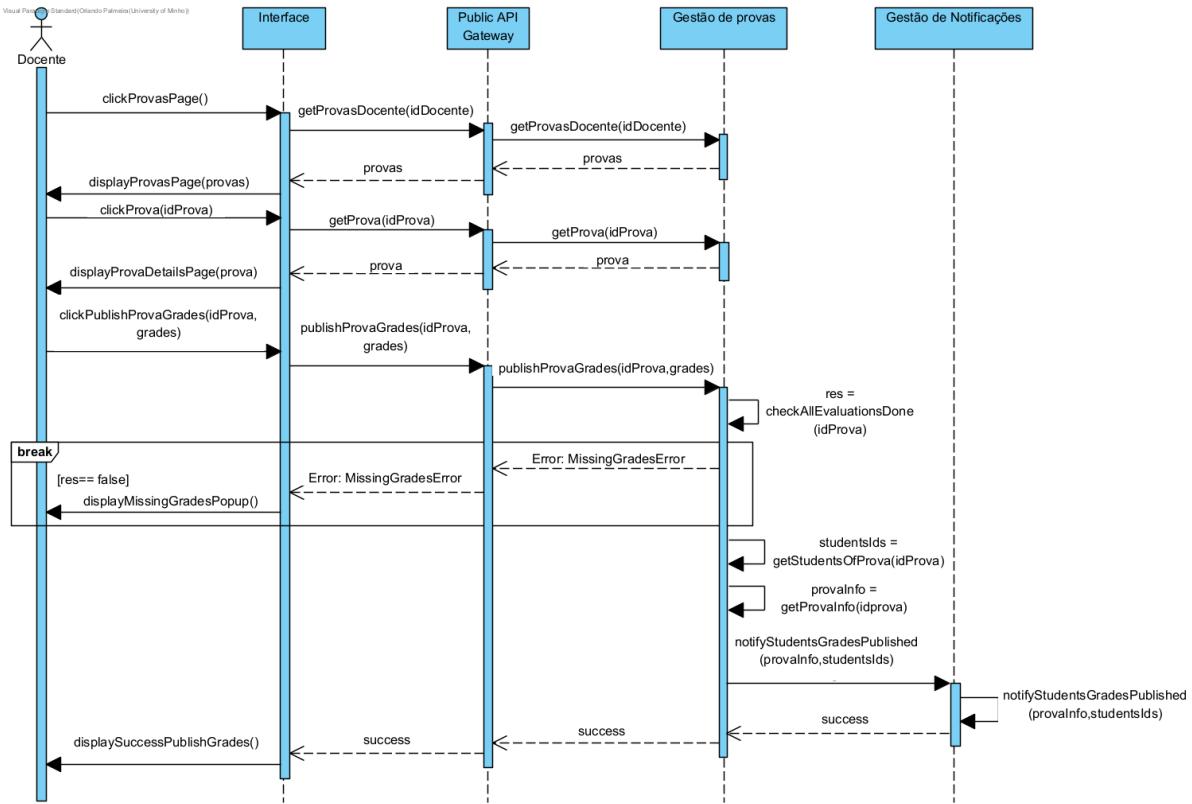


Figura 7.4: Diagrama de sequência - Publicar classificações da prova

7.1.5 Aceder às notificações

Este *use case* descreve a funcionalidade na qual um utilizador consulta as suas notificações. No diagrama de sequência, contempla-se a capacidade do utilizador receber um alerta na eventualidade de existirem notificações não lidas. Não obstante, o utilizador ainda tem a possibilidade de consultar notificações que já foram previamente lidas.

Este diagrama de sequência revela uma complexidade reduzida, uma vez que se trata unicamente de uma operação de consulta que envolve apenas um micro-serviço.

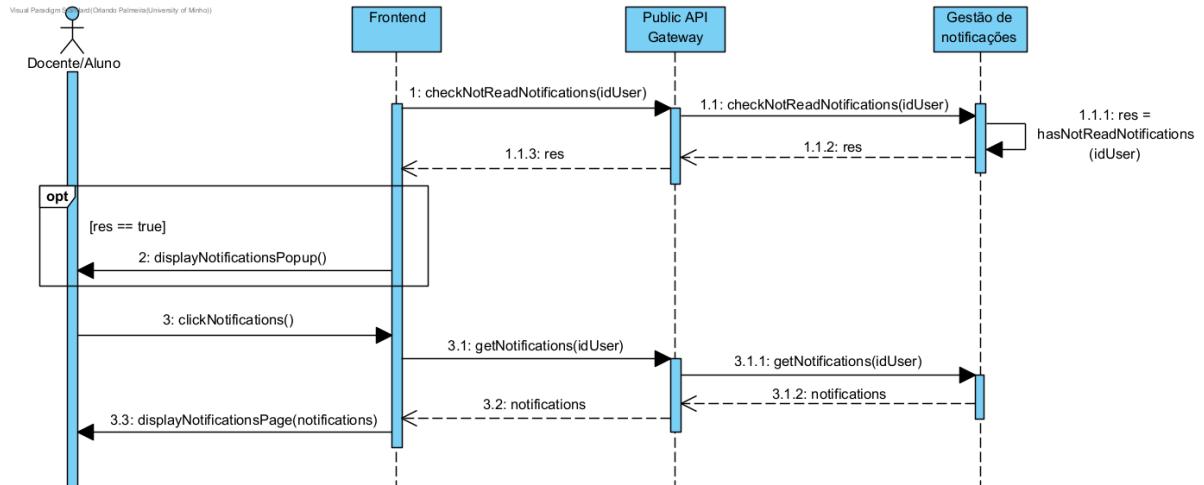


Figura 7.5: Diagrama de sequência - Aceder às notificações

7.1.6 Partilhar prova

Este *use case* descreve a funcionalidade de um docente partilhar uma prova sua com outros docentes. O docente que está a realizar a partilha deve inserir os *e-mails* dos docentes com quem pretende partilhar a prova. No diagrama de sequência está contemplada a verificação que o sistema faz aos *e-mails* fornecidos, verificando se estes existem e se pertencem a utilizadores que também são docentes.

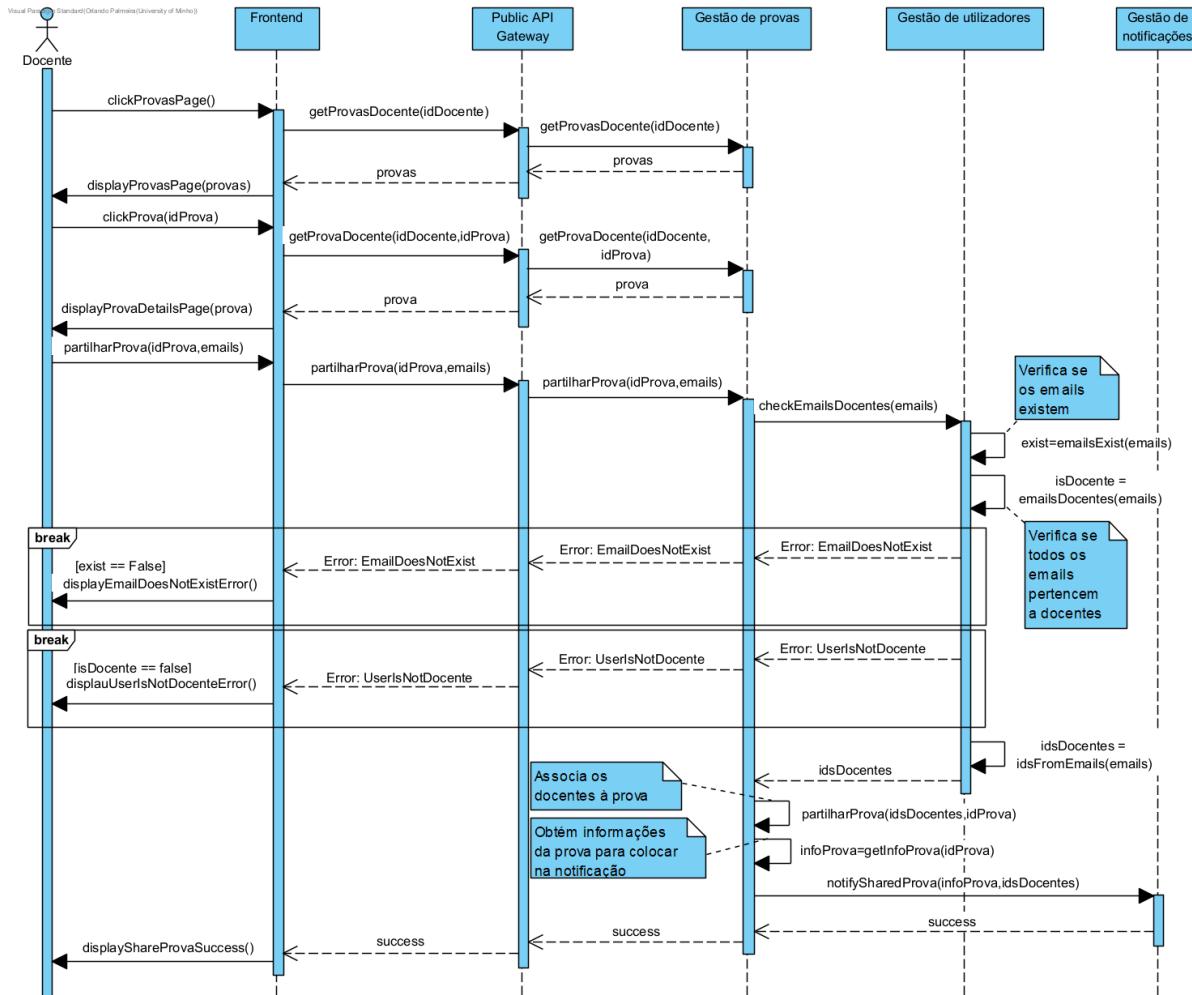


Figura 7.6: Diagrama de sequência - Partilhar prova

7.1.7 Registar alunos

Este *use case* descreve a funcionalidade de um técnico registar um aluno na plataforma do PROBUM. São fornecidas duas alternativas de registo: (i) registo individual em que o técnico insere manualmente os dados do aluno num formulário para o efeito ou (ii) registo por ficheiro em que é fornecido ao sistema um ficheiro com os dados de vários alunos. Em ambas as alternativas existe uma verificação dos dados inseridos, em que o sistema impede a submissão de dados corrompidos e/ou repetidos. Se os registo forem feitos com sucesso, os alunos registados receberão um e-mail com os dados das suas contas do PROBUM.

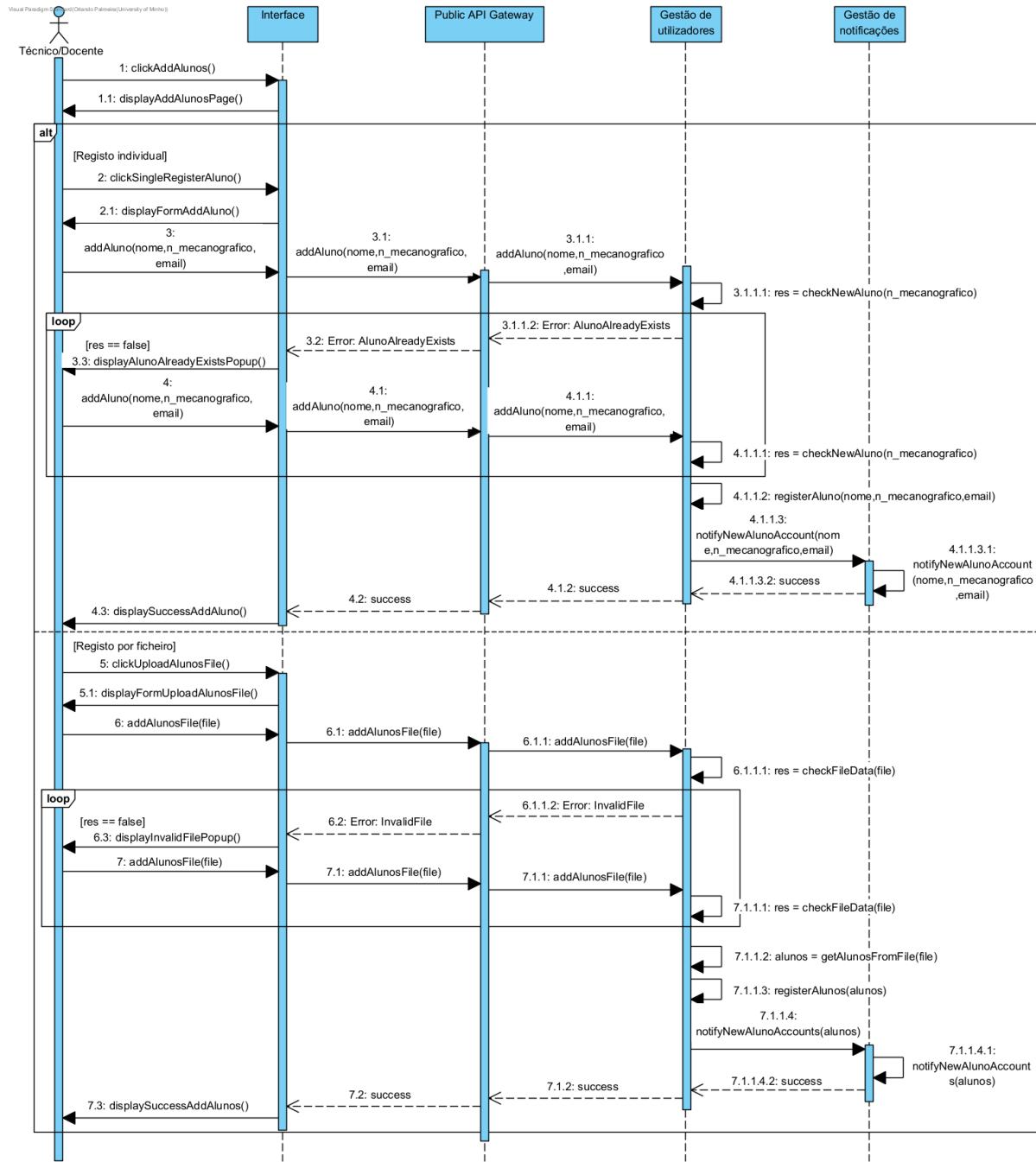


Figura 7.7: Diagrama de sequência - Registar alunos

7.1.8 Classificar respostas

Este *use case* descreve a funcionalidade de um docente efectuar a correcção das respostas dos alunos a uma determinada prova. No diagrama de sequência está contemplada a funcionalidade de correcção automática de questões criadas para esse efeito.

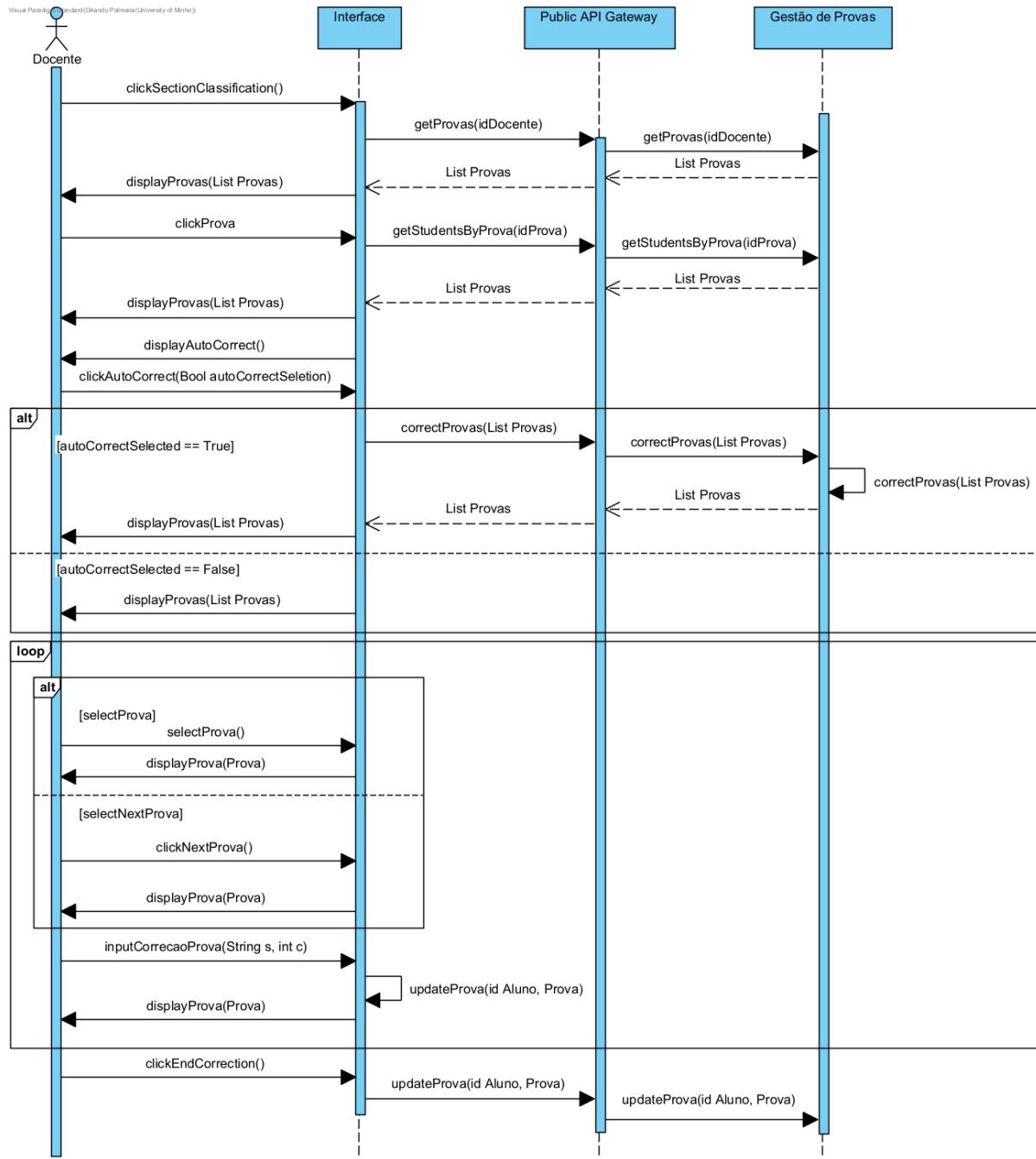


Figura 7.8: Diagrama de sequência - Classificar respostas

7.1.9 Editar perfil

Este *use case* descreve a funcionalidade em que um utilizador modifica alguns dos dados do seu perfil (email e/ou palavra-passe). O diagrama de sequência contempla todas as verificações realizadas pelo sistema ao *input* do utilizador, garantindo que a palavra-passe é segura e que o *e-mail* não está previamente registado no sistema.

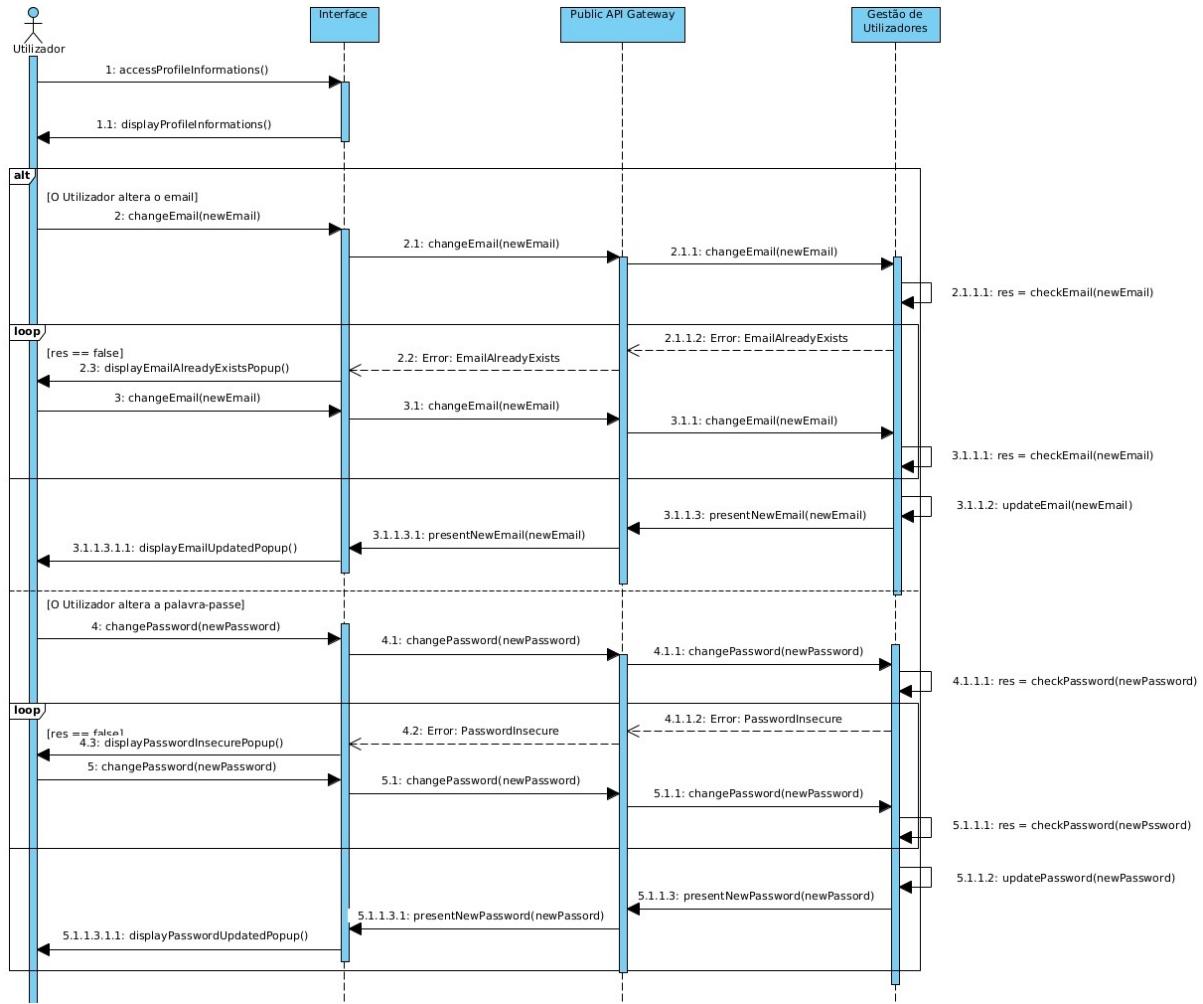


Figura 7.9: Diagrama de sequência - Editar perfil

7.1.10 Autenticar

Este *use case* descreve a funcionalidade em que um utilizador realiza a autenticação na plataforma do PROBUM. O diagrama de sequência engloba todas as verificações efetuadas pelo sistema no *input* do utilizador, que incluem a confirmação da existência do *e-mail* fornecido e a validação da palavra-passe.

Em caso de autenticação bem-sucedida, o utilizador será notificado adequadamente e receberá, de forma oculta, um *token* que servirá como método de identificação em todas as operações que o utilizador realizar.

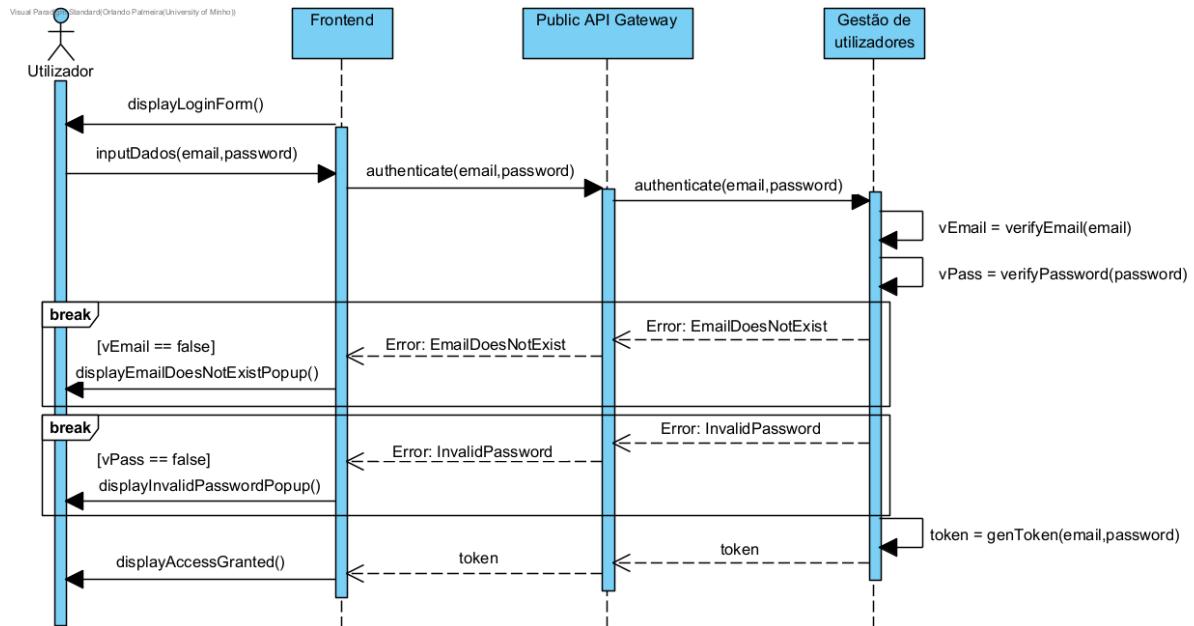


Figura 7.10: Diagrama de sequência - Autenticar

7.1.11 Consultar detalhes da prova

Este *use case* descreve a funcionalidade em que um aluno verifica os detalhes de uma prova cuja classificação já foi divulgada pelo docente responsável. Esta funcionalidade revela-se bastante directa, visto que se limita a uma mera consulta das informações de uma prova, o que implica um reduzido número de interações com a *API Gateway* e os micro-serviços.

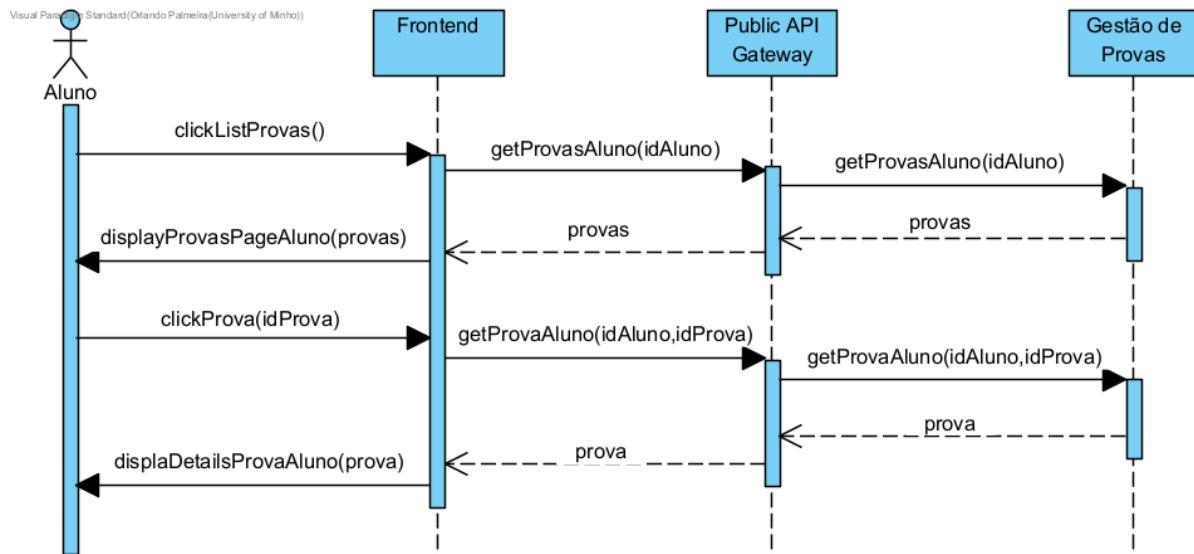


Figura 7.11: Diagrama de sequência - Consultar detalhes da prova

7.1.12 Criar questões

Este diagrama de sequência ilustra a funcionalidade da criação de questões por parte do Docente. O diagrama é bastante simples. Nesta funcionalidade apenas se justifica a utilização do *frontend* aquando do processo de armazenamento em memória das questões criadas pelo Docente.

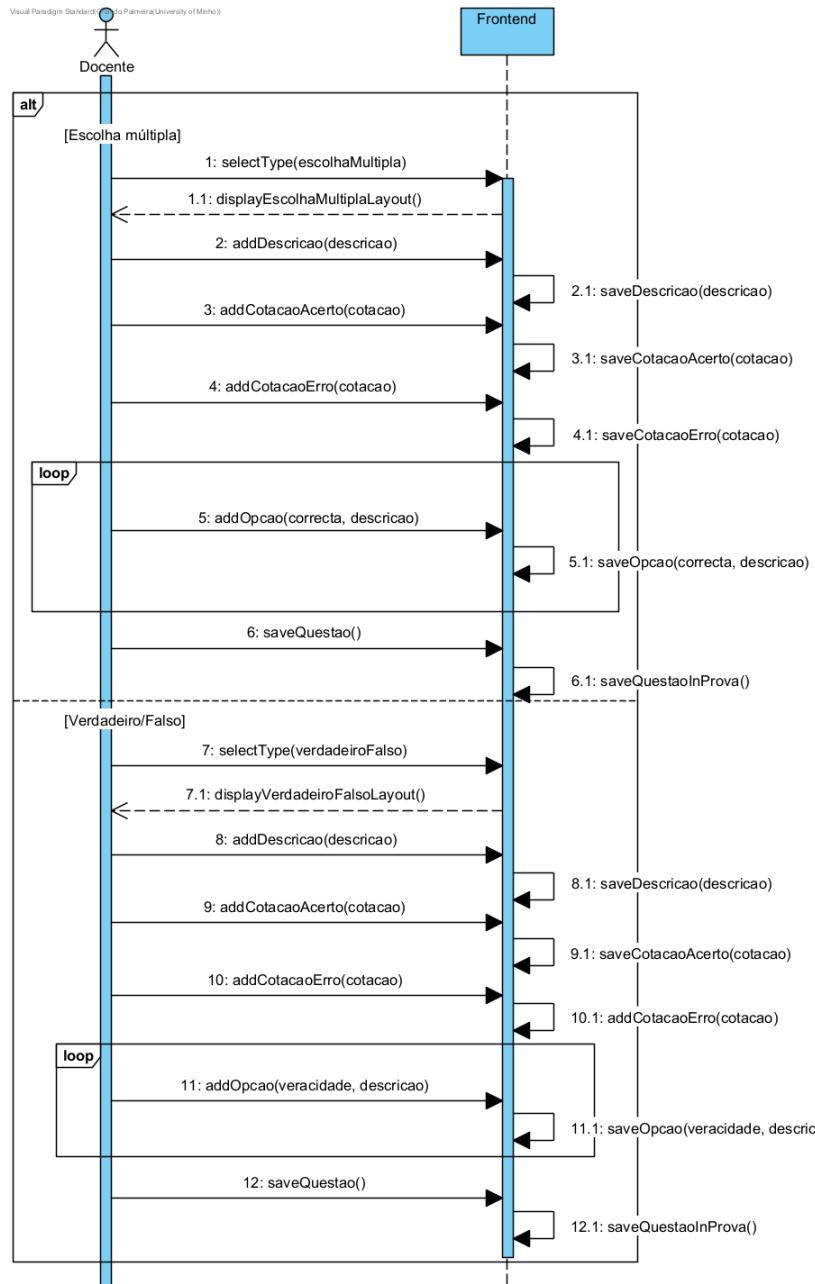


Figura 7.12: Diagrama de sequência - Criar Questões (actualizado)

Como podemos ver no diagrama actualizado, a criação das questões é feita totalmente de forma local. Desta forma reduzimos as transacções entre a API Gateway e os microsserviços. A justificação da alteração deste *use case* encontra-se na secção ”Alterações arquitecturais efectuadas”.

7.1.13 Editar prova

Este diagrama de sequência referente ao use case Editar Prova envolve essencialmente três serviços. Um desses serviços consiste no serviço de Gestão de Provas, um do serviço de Gestão de Notificações e o outro no serviço de Gestão de salas. O serviço de Gestão de Provas justifica-se pelo facto de necessitarmos de obter a listagem de provas que estão associadas ao determinado docente, assim como ter acesso a uma determinada prova selecionada, obter uma dada versão, guardar as alterações, ou visualizar os seus respetivos detalhes e questões. Já o serviço de Gestão de Salas justifica-se pelo facto de necessitarmos de saber a disponibilidade dos recursos antes

da realização da prova, ou seja, se o Docente alterar a data, ou a sala, teremos de aceder a esse serviço de modo a averiguar se essa sala está livre nesse horário. Logo, se essas informações referentes à realização da prova forem alteradas é necessário recorrer ao serviço de notificações para notificar os alunos de que a localização ou a hora foram alterados. No que toca às interações com a interface, estas envolvem apenas o armazenamento dos dados na memória do browser, pelo que não implica uma interação com os serviços (visível na note do diagrama). Por sua vez, a API Gateway irá coordenar quais os serviços que serão acedidos ao longo do processo.

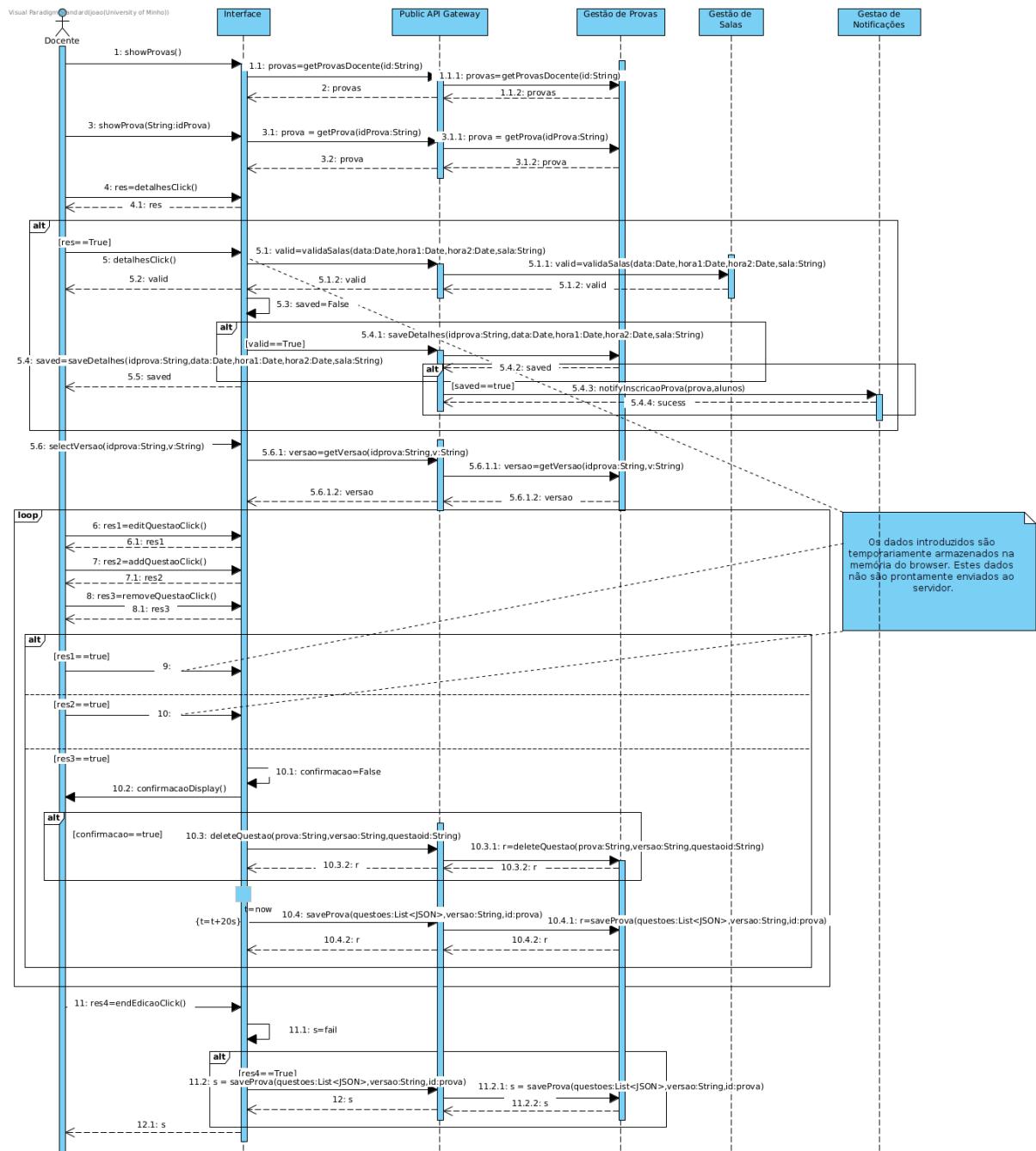


Figura 7.13: Diagrama de sequência - Editar prova

7.1.14 Responder a prova

Este diagrama de sequência representa a funcionalidade do aluno responder a uma prova na qual está inscrito. A interface do aluno começa por obter as provas nas quais o aluno está

inscrito. Após isso o aluno pode seleccionar uma prova e a partir daí responder às perguntas nela presente. No diagrama ainda é abordado o caso em que o sistema impede o retrocesso nas questões caso essa configuração seja seleccionada pelo docente.

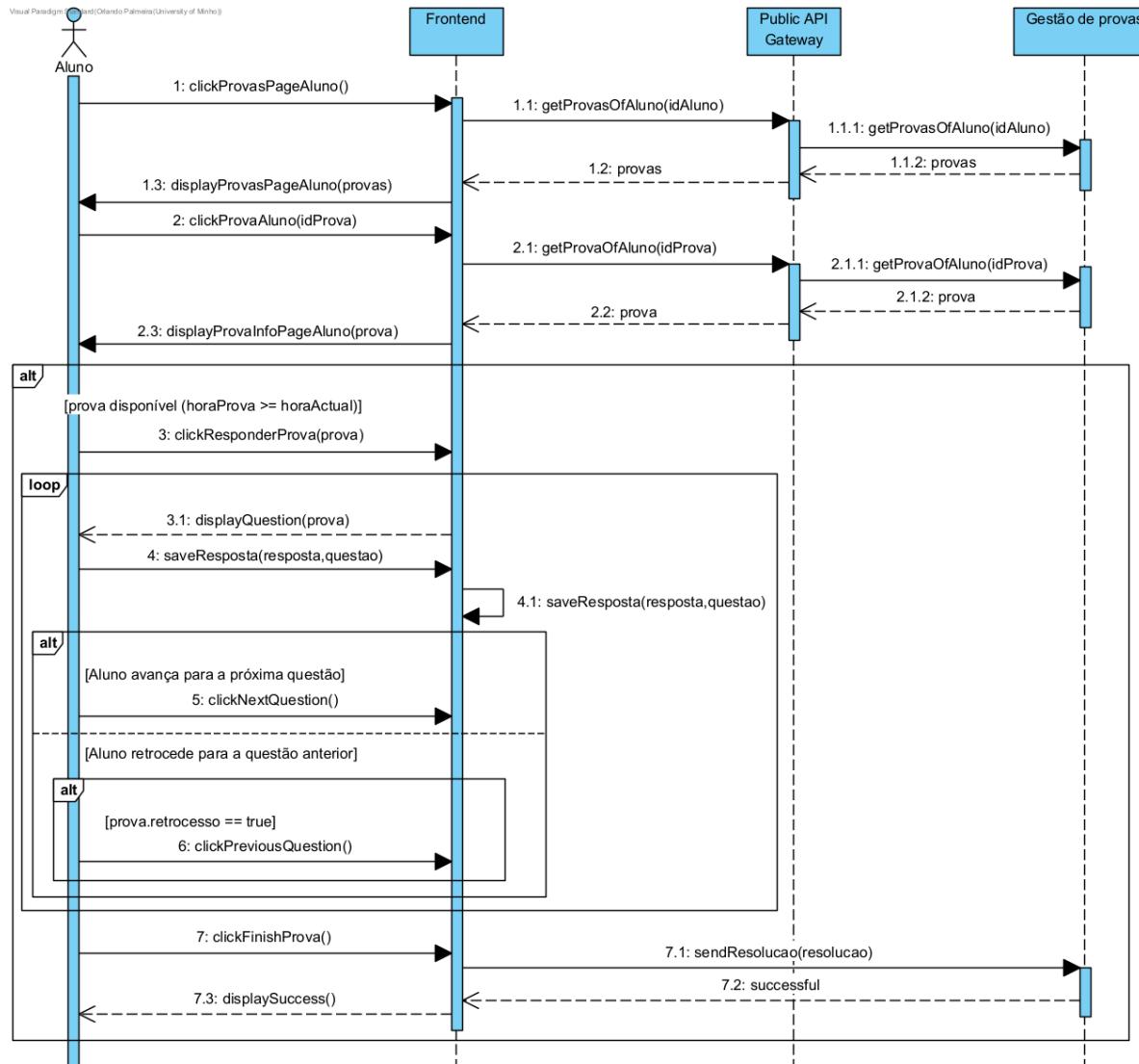


Figura 7.14: Diagrama de sequênciа - Responder Prova (actualizado)

O diagrama acima representa a versão implementada nesta fase. Neste caso, não foi abordado o tempo de admissão e as respostas são guardadas no dispositivo do aluno até que a prova seja terminada pelo aluno e só nesse momento é que as respostas são enviadas.

7.1.15 Editar questões

Este *use case* descreve a funcionalidade em que um docente edita as questões de uma prova. Esta funcionalidade é bastante linear pois a troca de informação revela-se sobretudo entre o Docente e a Interface. A interação entre o Docente, a *Public API Gateway* e o micro-serviço de Gestão de Provas, apenas se torna necessária quando o Docente pretende guardar as alterações das questões da respetiva prova.

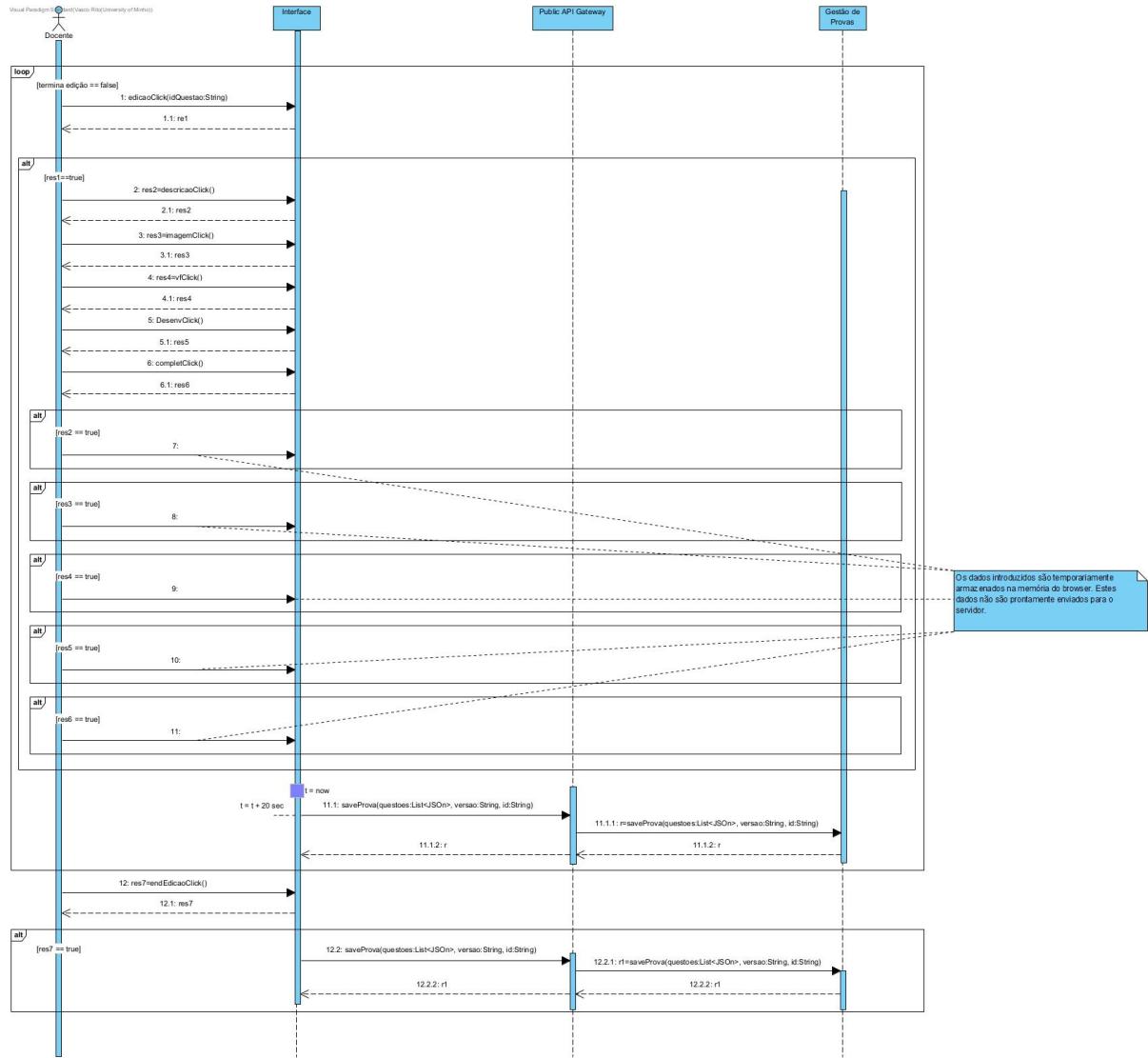


Figura 7.15: Diagrama de sequência - Editar Questões

7.1.16 Criar prova

O diagrama de sequência do UC de '*Criar Prova*' consiste no Docente como ator principal e envolve a criação de uma nova prova de avaliação. Este processo é um pouco mais complexo que os outros diagramas, pois apesar de envolver principalmente a interação entre o Docente e a Interface, inclui também interações com todos os outros micro serviços da nossa aplicação. Neste diagrama é necessária a introdução de detalhes básicos da prova, como data, hora e duração. Além disso, abrange o registo de alunos na prova e a criação de questões. Durante este processo, o Sistema propõe uma calendarização baseada nas informações fornecidas pelo Docente.

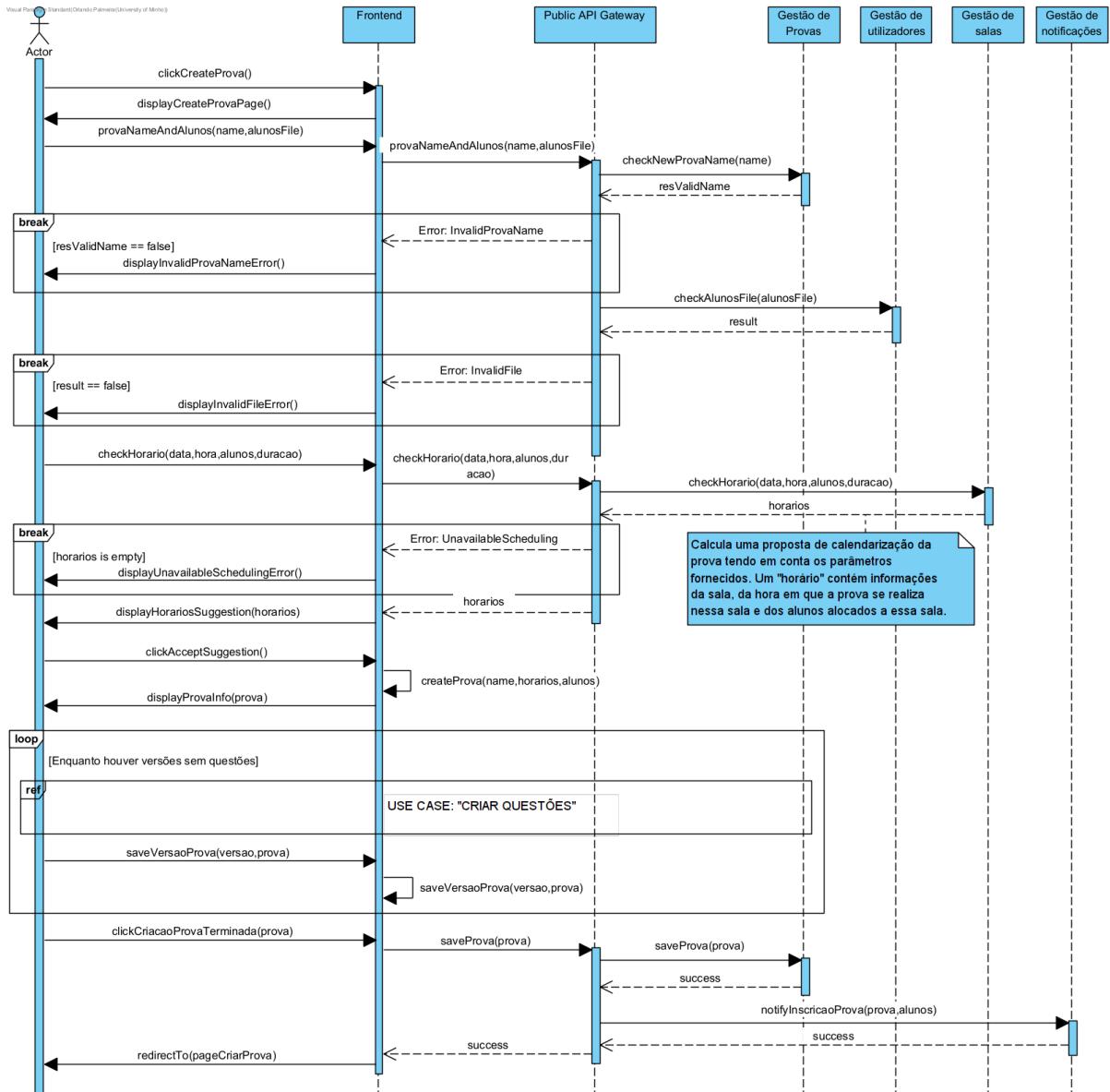


Figura 7.16: Diagrama de sequência - Criar Prova (actualizado)

O diagrama acima representa a versão actualizada do *use case* de criação de provas. Devido à complexidade deste *use case*, a equipa considerou a omissão/alteração de alguns passos. Esta alteração está devidamente justificada na secção ”Alterações arquitecturais efectuadas”.

8. Deployment View

Nesta secção iremos apresentar dois diagramas de *deployment* do sistema PROBUM. Primeiramente, iremos apresentar um diagrama em que toda a aplicação (Microsserviços, *API Gateway* e aplicação cliente) executa num só computador. Esse diagrama descreve o caso em que a aplicação está a ser executada para efeitos de teste e desenvolvimento. Por fim, apresentamos outro diagrama que demonstra o *deployment* ideal do sistema, isto é, como é que a aplicação seria instalada e executada num cenário de produção.

8.1 Contexto de desenvolvimento e teste

O nosso sistema é sustentado por uma arquitetura de microsserviços que prevê a instalação de cada microsserviço num sistema dedicado, ou seja, cada microsserviço(excluindo a API Gateway) é sustentado pela sua base de dados e pelo serviço de gestão do microsserviço. Estes elementos, por sua vez, vão estar inseridos num docker container, sendo que a comunicação de cada container é feita por HTTP. O Serviço API Gateway, que também vai estar inserido num docker container, vai ser responsável por fazer a comunicação entre os diferentes microsserviços e o Browser, ou seja, o Cliente que faz pedidos na aplicação.

Desta forma, foi elaborado o seguinte diagrama de *deployment*:

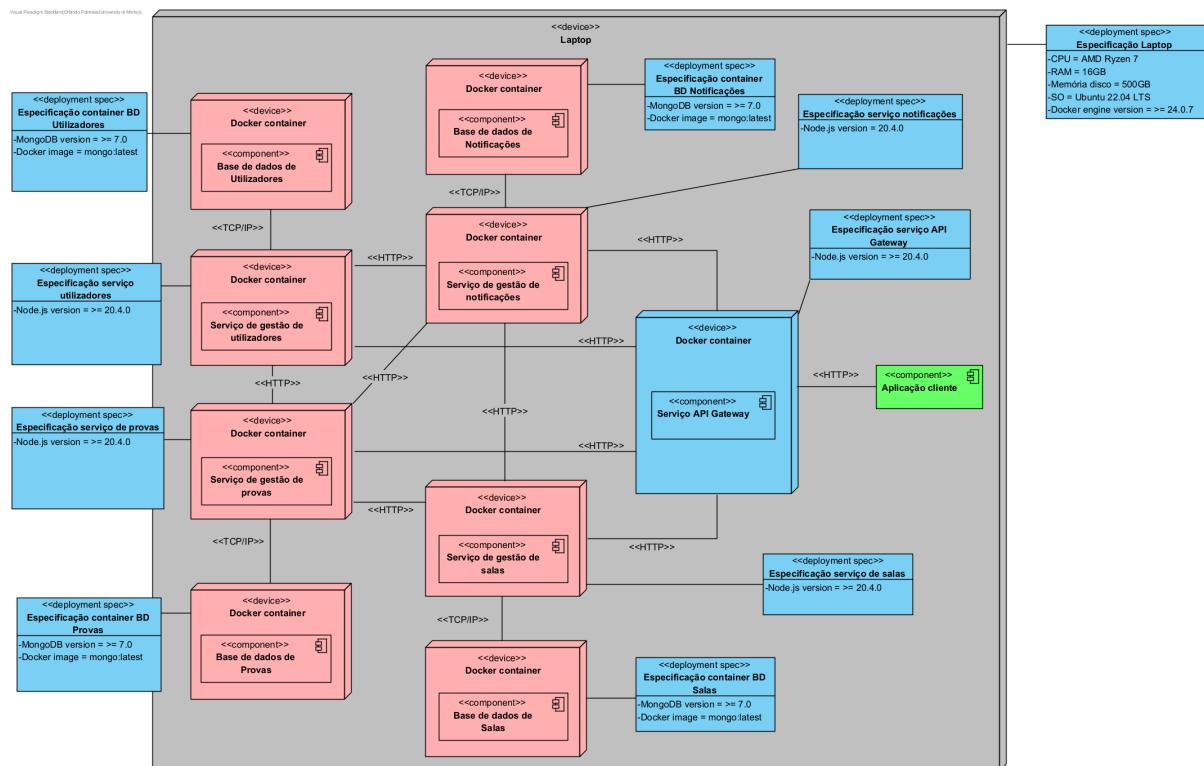


Figura 8.1: Diagrama de *deployment* (contexto de desenvolvimento e teste)

Como podemos ver, cada elemento deste sistema vai estar presente num mesmo dispositivo, e cada "microssistema" comunica com os outros através de sockets TCP/IP, sendo que as bases de dados só comunicam com o servidor do microsserviço correspondente.

8.2 Contexto de produção

Num contexto de produção, o sistema PROBUM necessita de ser instalado numa infraestrutura física que consiga sustentar as exigências impostas pelos utilizadores e as IES.

Para atingir esse objetivo, a equipa de desenvolvimento determinou que cada serviço da aplicação (*API Gateway* e Microsserviços) deve ser executado num dispositivo dedicado (máquina virtual). Para além disso, os recursos alocados serão fornecidos por um serviço de *cloud* (Amazon), uma vez que estes serviços fornecem a capacidade de ajustar automaticamente os recursos da aplicação à medida que a carga imposta varia com uma utilização mais ou menos intensiva, o que contribui para uma melhor elasticidade do sistema.

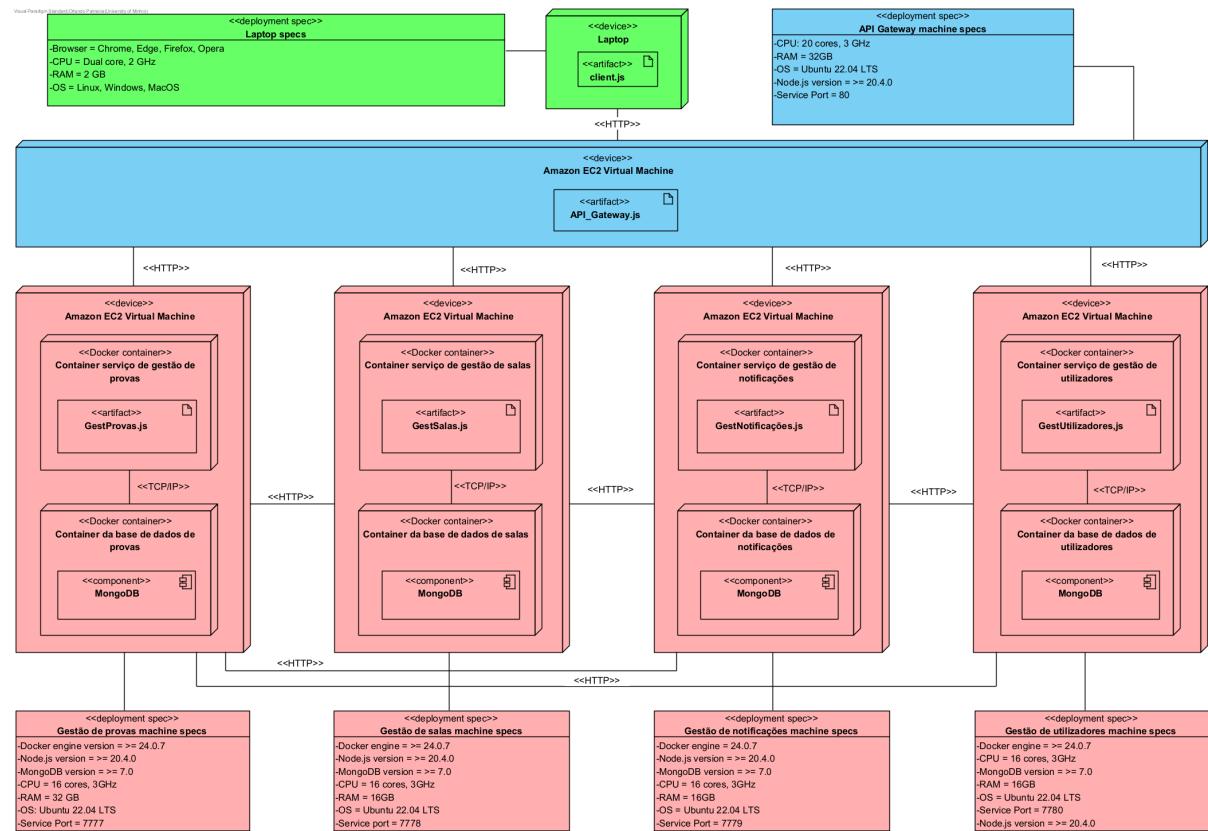


Figura 8.2: Diagrama de *deployment* (contexto de produção)

No diagrama acima, podemos observar que o sistema irá necessitar, no mínimo, de 5 servidores para atender às funcionalidades necessárias. São apresentados também os recursos computacionais mínimos para que cada componente do sistema funcione corretamente, sendo que os serviços da *API Gateway* e Gestão de provas são aqueles que têm mais recursos alocados. Isto deve-se ao facto destes dois serviços serem aqueles que serão mais requisitados na utilização do sistema. Por um lado, a *API Gateway* irá atender todos os pedidos enviados por todos os clientes. Por outro lado, a Gestão de provas estará sujeita a uma grande sobrecarga, uma vez que este serviço será altamente requisitado pelos docentes (criação, edição e classificação) e pelos alunos (realização e consulta de classificações).

Em cada servidor de cada microsserviço serão executados dois *containers* Docker relativos à base de dados e à própria implementação do microsserviço. No servidor da *API Gateway* não utilizaremos containers, uma vez que este será o único componente a ser executado no servidor da API, reduzindo o *overhead* da execução de um *container*.

9. Alterações arquitecturais efectuadas

Nesta secção iremos descrever as alterações que a equipa de desenvolvimento decidiu aplicar relativamente ao que foi planeado no documento da solução arquitectural entregue na fase anterior deste projeto.

9.1 Use cases

A equipa decidiu realizar algumas alterações aos use cases no que diz respeito a melhorar a simplicidade de implementação, bem como melhorar o desempenho através da tentativa de redução de transações entre microsserviços ou entre a API Gateway e os microsserviços.

9.1.1 Use case - Criar prova

Nos documentos de requisitos e de planeamento da solução arquitectural, este *use case* possui um grande número de passos e, devido a algumas limitações temporais para a entrega do projeto, alguns desses passos tiveram de ser ligeiramente alterados/omitidos.

Em primeiro lugar, não se conseguiu implementar a opção permitir ao docente aceitar/recusar os horários propostos pelo sistema. Neste momento, a aplicação apresenta um conjunto de propostas de calendarização da prova com os alunos e salas alocados a cada proposta e a prova fica marcada nos horários calculados pelo sistema.

A segunda alteração realizada é que a prova apenas é criada no fim, isto é, quando já tem toda a sua informação completa (o seu nome, alunos inscritos, versões, questões, etc.). Na versão do documento da solução arquitectural, a prova ficava registada no sistema ainda sem questões associadas e depois a aplicação cliente ia adicionando as questões à prova à medida que o docente as ia criando (Use case - Criar questões). A equipa considerou que, além da complexidade de implementação desta versão, esta exige uma carga maior no que diz respeito ao envio de pedidos aos microsserviços. Assim, decidiu-se que a prova fica efetivamente registada no sistema após o docente criar todas as questões da prova. Sucintamente, a prova fica guardada no dispositivo do docente até ao momento em que este efetua a submissão da prova no sistema e é por este motivo que, nesta fase, o diagrama de sequência atualizado só aborda transações entre o utilizador e o *frontend*.

9.1.2 Use case - Criar questões

Tal como o *use case* anterior, este também sofreu algumas alterações relativamente ao que foi previamente planeado.

Em primeiro lugar, na solução final só temos a possibilidade de criar questões de escolha múltipla e Verdadeiro/Falso.

Por uma questão de simplicidade de implementação e manipulação de dados, na solução final não é colocada uma cotação em cada opção de escolha múltipla ou Verdadeiro/Falso, mas sim uma cotação na pergunta em geral.

Omitiu-se também a possibilidade de colocar imagens nas questões, uma vez que ainda exigiria alguma complexidade no que diz respeito ao armazenamento de imagens na base de dados.

Nesta fase, o docente tem a possibilidade de criar questões de escolha múltipla e verdadeiro/falso. Quando cria uma questão, coloca a descrição, opções e cotação em caso de acerto ou erro.

9.1.3 Use case - Responder a prova

Relativamente ao *use case* de responder a uma prova, a equipa tentou aproximar-se ao máximo à implementação previamente planeada.

A implementação deste *use case* iniciou-se pela implementação dos passos principais que consistem em impedir o acesso a uma prova que ainda não se iniciou, bem como impedir o retrocesso nas questões caso o docente configure a prova desse modo.

Relativamente a alterações, a equipa não abordou o tempo de admissão e implementou a submissão de questões apenas no momento em que o aluno termina a prova. Desta maneira, reduzimos o número de interações que o *frontend* efetua com os microsserviços, uma vez que só no final da prova é que essa comunicação é efetuada.

9.1.4 Use case - Classificar respostas

Relativamente ao *use case* de Classificar uma prova apenas foi implementado um método de correção automática de Provas.

A implementação deste *use case* envolveu comparar todas as respostas dadas pelo aluno com as respostas dadas como corretas, foi decidido que o aluno só teria a pontuação se escolheu todas as respostas corretas e mais nenhuma, caso contrário é-lhe removida a penalidade.

Como nesta implementação apenas foram consideradas perguntas de escolha múltipla e de verdadeiro e falso, não foi implementada uma maneira de realizar correção manual.

9.2 Bases de dados

Nesta secção serão justificadas todas as alterações que foram necessárias nos modelos de dados do sistema.

9.2.1 Gestão de salas

Relativamente ao modelo de dados do microsserviço de gestão de salas previamente planeado, tivemos de acrescentar um campo "ocupacao" que possui uma lista de todos os horários em que a sala está ocupada. A presença deste campo facilita a implementação da calendarização além de a tornar mais eficiente. Na versão anterior (no documento da solução arquitetural) seria necessário percorrer as provas para saber quando uma sala está ocupada.

9.2.2 Gestão de provas

Relativamente ao modelo de dados do microsserviço de gestão de provas, a única alteração realizada foi a remoção do campo ”imagem”, uma vez que não tivemos a possibilidade de descobrir atempadamente soluções para armazenar imagens na base de dados. Nesta fase, o docente ainda não consegue criar questões com imagens. Além disso, o modelo de dados da Versão foi também alterado de modo a incluir mais detalhes sobre o local da prova e a sua duração, foram, por isso, adicionados os campos, *piso*, *edificio* e *duração*.

No que toca ao modelo da resolução nada foi alterado.

9.2.3 Gestão de Utilizadores

Relativamente ao modelo de dados do microsserviço de gestão de utilizadores, foram feitas algumas alterações. Inicialmente foi pensado em ter três *schemas*: um para os alunos, outro para os docentes e outro para os técnicos, pelo que chegamos à conclusão de que se tornava complicado de manipular os três tipos de utilizadores em conjunto. Devido a esse facto, os três *schemas* converteram-se num só que engloba todos os utilizadores, e que possui um campo *type* que distingue os utilizadores. Outra alteração feita foi na variável *name* que passou a ser denominada *username* por questões de autenticação.

9.3 *Building Block View*

9.3.1 Gestão de provas

Nesta fase, a equipa sentiu a necessidade de criar uma dependência no *controller* de provas relativamente ao *controller* de resoluções. Esta necessidade advém do facto da funcionalidade de verificar as provas **ainda não realizadas** nas quais um aluno está inscrito obter a informação das provas que o já aluno realizou. Para isso, o *controller* de provas precisa de aceder às informações das resoluções do aluno através do respectivo *controller*.

9.3.2 Gestão de Notificações

No período de conceção, seguindo uma abordagem EDUF, concebeu-se um componente de IMAP com o objetivo de munir o serviço com um correio eletrónico interno.

Tal não veio a ser implementado nesta iteração, optando exclusivamente pelo uso de uma componente de SMTP (recurso a correio eletrónico externo).

Infra seguem-se as ilustrações arquitectónicas.

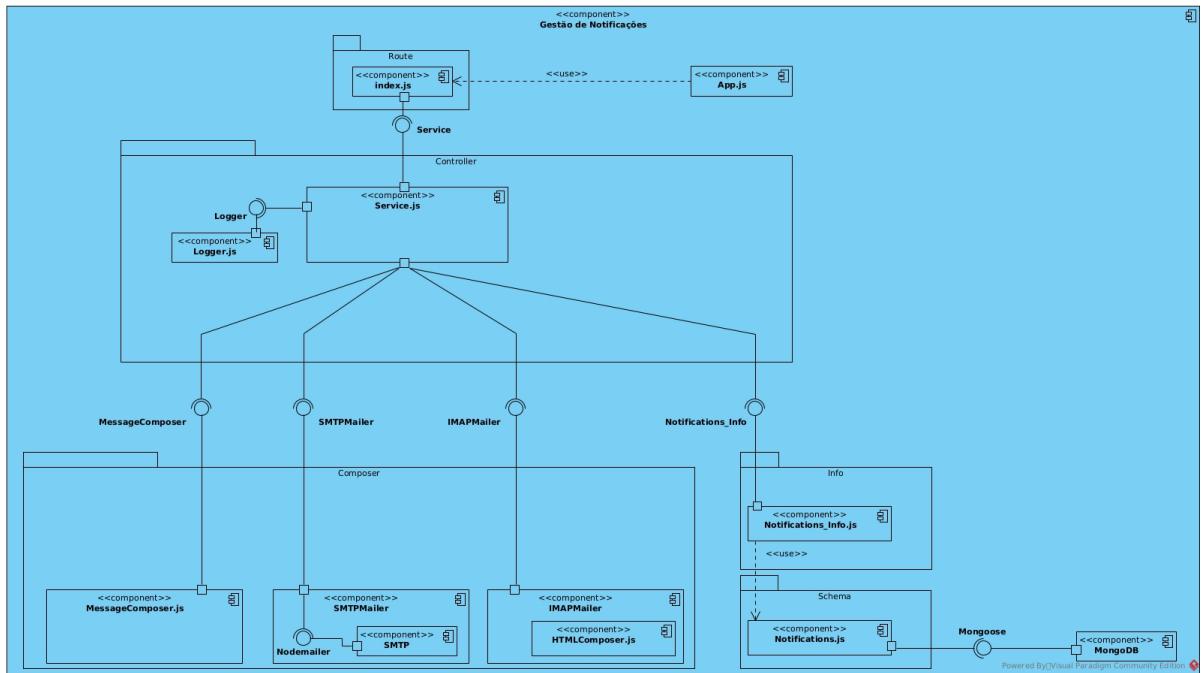


Figura 9.1: Arquitetura do microsserviço, abordagem EDUF.

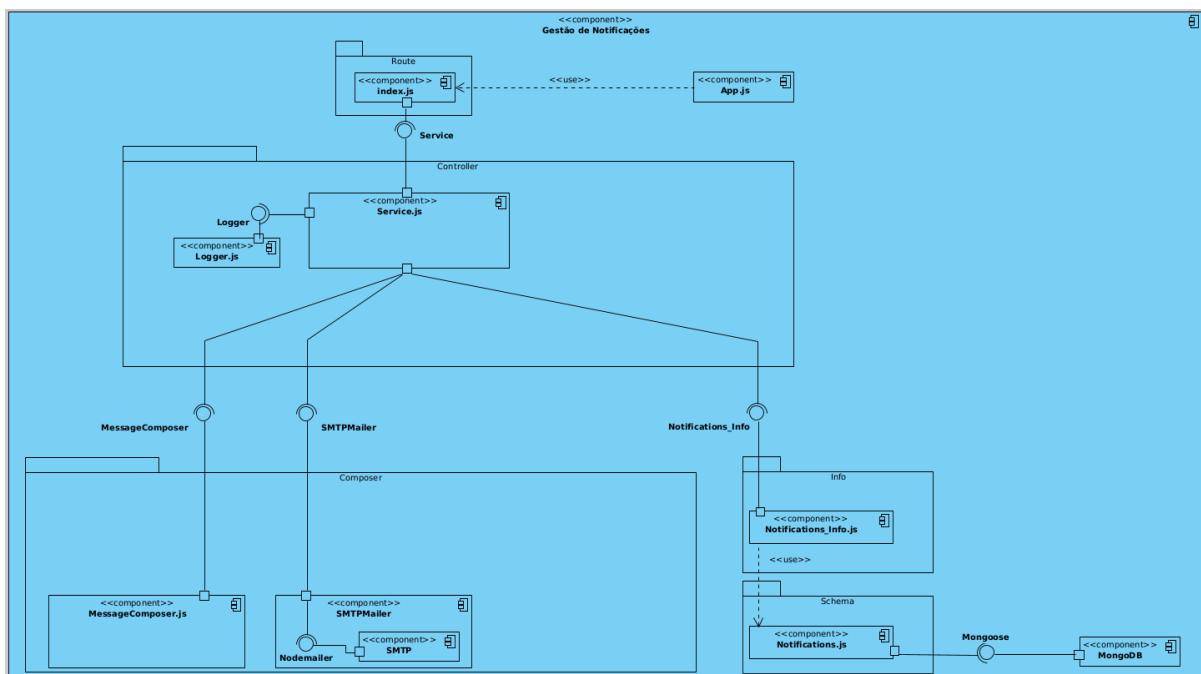


Figura 9.2: Arquitetura do microsserviço, implementação à data de 4 de janeiro, de 2024.

10. Microsserviços

10.1 Gestão de provas

Este microsserviço é responsável pela gestão das provas, bem como das resoluções dos alunos a essas provas. Nesta secção iremos explicitar a API fornecida por este serviço através da explicação de cada uma das rotas fornecidas.

10.1.1 Rotas da API

GET /provas/checkname?name="nome da prova"

Esta rota trata-se de um pedido GET no qual o nome da prova é enviado através de uma *querystring*. Posteriormente, o servidor do microsserviço de gestão de provas avaliará se esse nome já existe numa prova do sistema. Em caso afirmativo, indica que o nome é inválido. Caso contrário, indica que o nome é válido.

POST /provas

Esta rota trata-se de um pedido POST no qual todos os dados da prova (nome, salas, alunos, versões, questões, etc.) estão colocados no *body* do pedido. Após a sua receção, o servidor do microsserviço regista a prova na base de dados.

GET /provas/:id

Esta rota trata-se de um pedido GET no qual o identificador da prova é fornecido como um parâmetro. Neste pedido, o servidor do microsserviço devolve todos os dados de uma prova na resposta.

POST /provas/resolucoes

Esta rota trata-se de um pedido POST no qual todos os dados de uma resolução (id do aluno, id da prova, respostas do aluno, etc.) estão colocados no *body* do pedido. Após a sua receção, o servidor do microsserviço regista a resolução na sua base de dados.

GET /provas/alunos/:numMecAluno/naoRealizadas

Esta rota trata-se de um pedido GET no qual o número mecanográfico do aluno é fornecido como um parâmetro. No momento de receção deste pedido, o servidor deste microsserviço responde com todos os dados de todas as provas (não realizadas) nas quais o aluno está inscrito.

GET /provas/alunos/:numMecAluno/realizadas

Esta rota trata-se de um pedido GET no qual o número mecanográfico do aluno é fornecido como um parâmetro. No momento de receção deste pedido, o servidor deste microsserviço responde com todos os dados de todas as provas que o aluno já realizou.

GET /provas/:idProva/resolucoes/correcaoAuto

Esta rota trata-se de um pedido GET no qual o identificador da prova é fornecido como um parâmetro. No momento da receção do pedido, o servidor percorre todas as resoluções da prova solicitada e classifica todas as questões de escolha múltipla e Verdadeiro/Falso, atribuindo as respetivas cotações.

Esta rota é crucial para a funcionalidade de despoletar a correção automática das provas.

GET /provas/docente/:idDocente

Esta rota trata-se de um pedido GET no qual é fornecido o número mecanográfico de um docente. No momento do pedido o servidor percorre todas as provas e devolve as provas às quais o Docente especificado tem acesso.

GET /provas/questoes/:idProva/:versao

Esta rota devolve uma prova dado o id e a versão correspondente. Esta rota é usada para devolver as questões relativas de uma dada prova para assim ser possível fazer o render dessas questões para o aluno visualizá-las.

GET /provas/resolucoes/aluno/:numMecAluno/:idProva

Esta rota é usada para obter as resoluções de cada uma das questões de uma dada prova do aluno.

10.2 Gestão de salas

Este microsserviço é responsável pela gestão das salas onde se realizam as provas, bem como a alocação dessas salas para a realização das provas. Nesta secção iremos explicitar a API fornecida por este serviço através da explicação de cada uma das rotas fornecidas.

10.2.1 Rotas da API

POST /salas/calendarizacao

Esta rota trata-se de um pedido POST no qual se fornece (no *body*) a data e hora da prova e a lista de números mecanográficos dos alunos inscritos na prova. Na resposta ao pedido, o servidor devolve um conjunto de propostas de calendarização em que cada proposta possui uma data e hora, a sala da realização e os alunos alocados a essa sala.

Se não for possível realizar uma calendarização de acordo com os parâmetros enviados no pedido, o servidor responde ao pedido sem nenhuma proposta de calendarização, isto é, com uma lista vazia.

GET /salas/

Esta rota trata-se de um pedido GET, que não recebe argumentos extra, e ao qual o servidor responde retornando a lista de todas as salas registadas no sistema.

POST /salas/

Esta rota trata-se de um pedido POST, onde é fornecido um ficheiro com as informações referentes às salas a adicionar. O sistema responde adicionando as salas à base de dados. Em caso de insucesso é criada uma mensagem pop up de erro.

POST /sala/:idSala/alocar

Esta rota trata-se de um pedido POST no qual é fornecido o id de uma sala, a hora e a duração da prova. O sistema responde alocando a sala relativa ao id fornecido para as horas e duração correspondentes.

DELETE /salas/:idSala

Esta rota trata-se de um pedido DELETE, no qual se fornece o id da sala que pretendemos remover. Em resposta ao pedido, o servidor devolve a lista de salas disponíveis, já atualizada sem a sala removida.

POST /salas/alocar

Esta rota trata-se de um pedido POST, no qual são fornecidas várias salas, uma hora e duração. Como resposta o sistema aloca as salas respetivas para uma hora e duração, de modo a serem alocadas para uma prova.

10.3 Gestão de Utilizadores

Este microsserviço é responsável pela gestão de utilizadores. Nesta secção iremos explicitar a API fornecida por este serviço através da explicação de cada uma das rotas fornecidas.

POST /login

Esta rota faz um POST, no qual se fornece o email e a password de um utilizador. Na resposta a este pedido o servidor reencaminha o utilizador para a *homepage*, em caso de sucesso(valida o número mecanográfico) e em caso de insucesso apresentará a seguinte mensagem "O email ou a password que inseriu estão incorretos. Tente novamente.".

POST /register

Esta rota faz um POST, no qual se fornece um *body* com o seguinte conteúdo: nome, email, password, numero mecanográfico e tipo de utilizador(técnico, docente, aluno). Como resposta, o servidor valida o registo do novo utilizador, em caso de sucesso, o utilizador é adicionado à base de dados e é direcionada para a página de login, caso contrario irá aparecer um pop up de insucesso.

GET /users

Esta rota trata de um pedido, no qual todos os dados dos utilizadores (*_id*, *username*, *numMecanografico*, *email*, *password* e *type*) são devolvidos como resposta através do microsserviço.

GET /users/alunos/verify

Esta rota recebe uma lista de números mecanográficos de Alunos e verifica se todos se encontram registados no Sistema.

GET /users/alunos/:numMecanografico

Por fim, esta rota trata de um pedido de GET, cujo objetivo é verificar a existência de um dado aluno a partir do seu número mecanográfico.

10.4 Gestão de Notificações

O microsserviço de notificações rege o sistema com dois tipos de notificação: (1) ativa, e (2) passiva.

No que diz respeito à notificação ativa, esta engloba avisos em tempo real acerca das burocracias relacionadas com uma prova: (1) criação, (2) alteração, (3) disponibilização, e (4) correção. Estas notificações são feitas através de dois mecanismos: (1) de *push*, e (2) de *pull*.

O mecanismo de *push* é utilizado aquando de do momento da gestão ou criação de uma prova. Ao passo que o de *pull* é utilizado aquando de um pedido, explícito, de acesso às suas notificações.

Quanto às notificações passivas, estas realizam-se através de um protocolo SMTP, e têm como destino os próprios *e-mails* das contas dos utilizadores.

Para além do envio das notificações supra mencionadas, relacionadas com as provas, acresce ainda as notificações (sob a forma de *e-mail*) inerentes às contas da plataforma, quer relacionadas com a criação destas, quer relacionadas com a gestão destas (recuperação e alteração da palavra-passe).

O envio de notificações realizou-se através da utilização de **WebSockets**. De forma, a que fosse possível despoletar a notificação após um dado evento, foi necessário para essa dada interface criar uma instância de um socket usando a função **io** fornecida pelo Socket.IO. Essa instância se conecta a um servidor na porta 8877 e vamos passar o número mecanográfico correspondente a esse aluno. Do lado do servidor, vamos armazenando as conexões dos observadores num dicionário que será identificado pelo seu número e terá como valor um socket que é obtida através da promessa devolvida em **io.on**. A partir desse socket iremos posteriormente enviar a notificação(após ter sido computada/construída) a um dado user através do método **s.emit**. Para isso iremos fazer uma travessia das chaves do dicionário e quando o número do destinatário de uma notificação for igual à chave, enviámos para esse user a notificação a partir do socket que estava armazenado.

10.4.1 Rotas do microsserviço de notificações

POST /notifications/docente

Rota para guardar e enviar a notificação do registo de um novo docente(s).

POST /notifications/aluno

Rota para guardar e enviar a notificação do registo de um novo aluno(s).

POST /notifications/newprova

Rota para guardar e enviar as notificações da inscrição de alunos numa prova.

POST /notifications/editprova

Rota para guardar e enviar as notificações da alteração da prova aos alunos inscritos.

GET /notifications/unread/:id

Rota para obter as notificações não lidas de um dado utilizador.

GET /notifications/:id

Rota para obter as notificações de um dado utilizador.

POST /notifications/grades

Rota para guardar e enviar as notificações do lançamento das notas da prova aos alunos inscritos na mesma.

POST /notifications/unavailableroom

Rota para guardar e enviar as notificações da remoção de uma dada sala.

11. Grau de completude da solução

Nesta secção indicamos quais os requisitos funcionais e não funcionais que foram possíveis de ser cumpridos na solução de modo a avaliar o grau de completude do produto final.

11.1 Requisitos funcionais

O documento de requisitos que serviu de base ao desenvolvimento do projecto tem um total de 68 requisitos funcionais. Com as funcionalidades implementadas nesta fase fomos capazes de cumprir os seguintes:

- RF7: O Utilizador autentica-se com o email e password
- RF8: O Sistema informa no caso do email não existir ou a password estar errada
- RF14: O Docente cria uma prova de avaliação com os seguintes parâmetros: nome da prova, ficheiro com as informações dos alunos a participar na prova, data, hora preferencial, tempo de admissão e duração
- RF15: O Sistema calcula a disponibilidade das salas existentes
- RF16: O Sistema informa sobre as salas e respetivos horários indisponíveis
- RF17: O Sistema faz uma calendarização inteligente, indicando as salas, horários e alunos alocados a cada sala
- RF19: O Docente inscreve Alunos numa prova de avaliação
- RF20: O Sistema notifica, por email e pela plataforma, os Alunos que o Docente inscreveu numa prova de avaliação com os detalhes da Prova
- RF21: O Docente cria diferentes versões de uma prova de avaliação e associa cada versão às salas e horários calendarizados
- RF22: O Docente adiciona Questões de escolha múltipla a uma prova de avaliação
- RF23: O Docente adiciona Questões de Verdadeiro ou Falso a uma prova de avaliação
- RF26: O Docente adiciona um enunciado para cada questão
- RF29: O Sistema disponibiliza a lista de provas a que um Utilizador tem acesso
- RF30: O Utilizador acede aos detalhes de uma Prova, contendo o nome, duração, salas, horários e alunos alocados a cada ronda
- RF48: O Aluno tem uma opção para responder a Provas
- RF49: O Sistema mostra ao Aluno uma questão de cada vez, indicando o enunciado da mesma e um espaço para responder

- RF50: O Aluno seleciona a questão que pretende responder
- RF51: O Aluno submete uma resposta
- RF52: O Aluno termina a Prova a qualquer momento
- RF56: O Sistema apresenta uma confirmação aquando da submissão da Prova
- RF62: O Aluno tem acesso a uma secção com todas as Provas que já realizou
- RF63: O Aluno consulta a sua classificação de uma Prova
- RF64: O Aluno consulta as questões, respostas dadas e pontuação atribuída a cada questão de uma Prova
- RF68: O Utilizador acede a uma lista de notificações

Foi satisfeito um total de 24 requisitos funcionais dos 68 existentes, o que leva a um grau de cobertura de aproximadamente 35%.

Este valor não é muito elevado, uma vez que a equipa de desenvolvimento aplicou mais esforços e tempo na implementação das funcionalidades solicitadas no MVP.

11.2 Requisitos não funcionais

Relativamente aos requisitos não funcionais, o documento de requisitos possuía 24, sendo que a equipa considera ter satisfeito os seguintes:

- RNF1: O Sistema mostra um cronómetro com o tempo restante até ao final da Prova
- RNF2: As questões são facilmente legíveis
- RNF3: O produto é fácil de usar por pessoas que frequentam o ensino superior, ou seja, com capacidades robustas para lidar com tecnologia
- RNF4: O produto pode ser utilizado por pessoas com algum tipo de deficiência visual
- RNF5: O formato dos ficheiros submetidos deverá ser JSON
- RNF9: Qualquer interação entre o utilizador e o Sistema tem um tempo de resposta inferior a 2 segundos
- RNF12: O sistema deve correr em computadores de gama baixa
- RNF18: O Sistema informa se o utilizador já estiver registado
- RNF19: O Sistema informa se o ficheiro submetido for inválido, indicando a razão
- RNF21: O Sistema restringe o acesso a funcionalidades de acordo com o utilizador
- RNF22: O Sistema guarda e valida palavras-passe através de técnicas criptográficas
- RNF24: Os dados de cada utilizador não devem ser partilhados com terceiros sem autorização

Foi satisfeito um total de 12 requisitos não funcionais dos 24 existentes, o que leva a um grau de cobertura de 50%.

12. Contributos individuais na implementação

Nesta secção iremos apresentar todos os pontos da implementação do projecto nos quais cada elemento da equipa contribuiu.

Microsserviço	Número	Contributos
Provas	PG54123	<p>Funcionalidades implementadas: Criação de prova; responder a prova; calendarização automática.</p> <p>Microsserviços envolvidos: Gestão de provas; Gestão de salas</p> <p>Rotas implementadas (gestão de provas):</p> <ul style="list-style-type: none">- POST /provas: Para criar/inserir uma prova no sistema- GET /provas/checkname: Para verificar a validade dum novo nome de uma prova.- GET /provas/:idProva/resolucoes/correcaoAuto: Despoleta a correcção automática de uma prova- GET /provas/alunos/:numMecAluno/naoRealizadas: Obtém as provas (não realizadas) em que um aluno está inscrito- GET /provas/alunos/:numMecAluno/realizadas: Obtém todas as provas que o aluno já realizou- POST /provas/resolucoes: Regista a resolução de um aluno a uma dada prova <p>Rotas implementadas (gestão de salas):</p> <ul style="list-style-type: none">- POST /salas/calendarizacao: Para o cálculo de calendarização da prova dados os alunos e as salas disponíveis. <p>Páginas implementadas (frontend):</p> <ul style="list-style-type: none">- Login- Criação da prova- Realização da prova

Microsserviço	Número	Contributos
Provas	PG52682	<p>Funcionalidades implementadas: Classificação automática de uma Prova.</p> <p>Microsserviços envolvidos: Gestão de Provas.</p> <p>Rotas implementadas (gestão de provas):</p> <ul style="list-style-type: none"> - GET /provas/:idProva/resolucoes/correcaoAuto: Realiza a correção automática de uma prova - GET /provas/docente/:idDocente: Obtém todas as provas às quais um docente tem acesso. <p>Páginas implementadas (frontend):</p> <ul style="list-style-type: none"> - Classificação Automática.
Provas	PG52705	<p>Funcionalidades implementadas: Devolução de todas as Provas.</p> <p>Microsserviços envolvidos: Gestão de Provas.</p> <p>Rotas implementadas (gestão de provas):</p> <ul style="list-style-type: none"> - GET /provas: Retorna todas as provas
Provas	A98728	<p>Pop up da notificação</p> <p>Página das provas realizadas</p> <p>Página das classificações</p>

Microsserviço	Número	Contributos
Salas	A93318	<p>Funcionalidades implementadas: <i>homepage</i> dos técnicos para gestão de salas; Remoção de salas.</p> <p>Microsserviços envolvidos: Gestão de salas</p> <p>Rotas implementadas (gestão de salas)</p> <ul style="list-style-type: none"> - DELETE /salas/:idSala Remove a sala pretendida. <p>Páginas implementadas (frontend):</p> <ul style="list-style-type: none"> - Homepage dos técnicos para gestão de salas - Listagem e remoção de salas disponíveis
Salas	PG53825	<p>Rotas implementadas (gestão de salas):</p> <ul style="list-style-type: none"> - GET /salas/: Retorna todas as salas disponíveis. - POST /salas/: Adiciona salas à base de dados - POST /salas/:idSala/locar: Aloca uma sala para um certo intervalo de tempo - POST /salas/locar: Aloca várias salas para um certo intervalo de tempo

Microsserviço	Número	Contributos
Utilizadores	PG53932	<p>Funcionalidade implementadas: Login e Register de utilizadores</p> <p>Microsserviços envolvidos: Gestão de utilizadores</p> <p>Rotas implementadas (API Gateway):</p> <ul style="list-style-type: none"> - POST /login: Login de utilizador - POST /register: Regista utilizador na base de dados - GET /users : Devolve todos os utilizadores presentes na base de dados - GET /users/alunos/:numMecanografico: Verifica se aluno existe pelo número mecanográfico <p>Rotas implementadas (gestão de utilizadores):</p> <ul style="list-style-type: none"> - POST /users/login - POST /users/register - GET /users - GET /users/alunos/verify: Faz a verificação de uma lista de alunos na base de dados, pelos números mecanográficos. - GET /users/alunos/:numMecanografico
Utilizadores	PG54004	<p>Funcionalidade implementadas: Login e Register do utilizadores</p> <p>Microsserviços envolvidos: Gestão de utilizadores</p> <p>Rotas implementadas (API Gateway)</p> <ul style="list-style-type: none"> - POST /login: Login de utilizador - POST /register: Regista utilizador na base de dados <p>Rotas implementadas (gestão de utilizadores):</p> <ul style="list-style-type: none"> - POST /users/login - POST /users/register
Utilizadores	PG53840	<p>Rotas implementadas (API Gateway)</p> <ul style="list-style-type: none"> - POST /login: Login de utilizador - POST /register: Regista utilizador na base de dados <p>Rotas implementadas (gestão de utilizadores):</p> <ul style="list-style-type: none"> - POST /users/login - POST /users/register

Microsserviço	Número	Contributos
Notificações	PG53929	<ul style="list-style-type: none"> - Rotas; - App; - Index; - Service (integração); - SMTPMailer; - Modelo de Dados; Frontend: - HomePage do aluno; - Pop up da notificação; - Página das provas realizadas; - Página das classificações;
Notificações	PG54232	<ul style="list-style-type: none"> - Logger; - MessageComposer; - SMTPMailer; - Service (esqueleto).