

UNIVERSIDADE DO MINHO

Gestão de Espaços

MIEI

Engenharia de Aplicações

(2.º Semestre/4.º Ano)

**A73831 João Barreira
PG38419 Miguel Brito
A78434 Pedro Silva**

Braga
Julho 2019

Índice

1	Introdução	3
2	Especificação do sistema	3
2.1	Levantamento de Requisitos	3
2.1.1	Requisitos Funcionais	3
2.1.2	Requisitos Não Funcionais	4
2.2	Perfis de Utilizador	4
2.2.1	Visitante	4
2.2.2	Utilizador Comum	4
2.2.3	Utilizador com Privilégios de requisição de espaços	5
2.2.4	Gestor de Espaços	5
2.2.5	Administrador	5
2.3	Interfaces do Sistema	5
2.3.1	Ecrãs junto a cada espaço	5
2.3.2	Ecrãs em espaços comuns	5
2.3.3	Interface <i>web</i>	5
2.4	Mockups	6
2.5	Mapa de Navegação	11
3	Frameworks	12
4	Modelação	13
4.1	Modelo de Domínio	13
4.2	Modelação de Tarefas	14
4.3	Modelo de Casos de Uso	15
4.3.1	Especificação dos Use Case	17
4.3.2	Diagramas de Sequência de Sistema	18
4.4	Platform Independent Model	19
4.5	Platform Specific Models	21
5	Implementação do Sistema	24
5.1	Server-side	24
5.1.1	Model	24
5.1.2	Camada de Dados	24
5.1.3	Classes de Serviço	24
5.1.4	Web Services REST	26
5.1.5	Integração com API Externa	27
5.2	Client-side	27
6	Deployment da Aplicação	29
6.1	Docker	29
6.2	Docker-Compose	30
7	Testes de Carga	32
8	Conclusão	36
A	Modelos de Tarefas	38
A.1	Consultar Evento	38
A.2	Consultar Evento que Segue	38

B Especificação dos Use Case	39
B.1 Consultar Evento	39
B.2 Consultar Horário de um Espaço	39
B.3 Seguir Evento	39
B.4 Deixar de Seguir Evento	39
B.5 Consultar Eventos que está a Seguir	40
B.6 Consultar Notificações	40
B.7 Consultar Eventos pelos quais é Responsável	40
B.8 Efetuar pedido de alteração de Evento	41
B.9 Consultar os seus pedidos	41
B.10 Aprovar Pedido	42
B.11 Rejeitar Pedido	42
B.12 Cancelar Pedido	43
B.13 Definir um novo evento	43
B.14 Modificar Evento	44
B.15 Cancelar Evento	44
B.16 Definir Espaço Comum	45
B.17 Atualizar constituição de um Espaço Comum	45
B.18 Apagar Espaço Comum	46
B.19 Carregar Espaços	46
B.20 Carregar Horários	46
B.21 Registar Atores	47
B.22 Alterar Estatuto	47
B.23 Login	47
B.24 Logout	48
C Diagramas de Sequência de Sistema	48
C.1 Consultar Evento	48
C.2 Consultar Horário de um Espaço	48
C.3 Seguir Evento	48
C.4 Deixar de Seguir Evento	48
C.5 Consultar Eventos que está a Seguir	49
C.6 Consultar Notificações	49
C.7 Consultar Eventos pelos quais é Responsável	49
C.8 Efetuar pedido de alteração de Evento	50
C.9 Consultar os seus pedidos	50
C.10 Aprovar Pedido	50
C.11 Rejeitar Pedido	51
C.12 Cancelar Pedido	51
C.13 Definir um novo evento	52
C.14 Modificar Evento	52
C.15 Cancelar Evento	53
C.16 Definir Espaço Comum	53
C.17 Atualizar constituição de um Espaço Comum	54
C.18 Apagar Espaço Comum	54
C.19 Carregar Espaços	55
C.20 Carregar Horários	55
C.21 Registar Atores	55
C.22 Alterar Estatuto	55
C.23 Login	56
C.24 Logout	56

Resumo

O presente relatório diz respeito ao processo de desenvolvimento de uma aplicação *web* que tem como objetivo a gestão dos espaços de um edifício e seus eventos.

Para tal, em primeiro lugar, foi realizada a especificação do sistema onde se idealizou o mesmo a vários níveis, desde os requisitos, os perfis de utilizador, as interfaces (e respetivas *mockups*) e o mapa de navegação.

Tendo como base um trabalho anterior realizado no âmbito da Unidade Curricular de Arquiteturas Aplicacionais, foi efetuada, de seguida, a escolha das *frameworks* que seriam utilizadas para a conceção da aplicação.

Efetuou-se, depois, a modelação do sistema a ser implementado tanto do ponto de vista abstrato, como adaptando o processo tendo em consideração as tecnologias escolhidos e que viriam a ser utilizadas.

Todo este processo culminou com o desenvolvimento da aplicação com o *back-end* assente em Java (Spring), sendo a camada de persistência composta pela integração do *Hibernate* em cima de uma base de dados *MySQL*. Já o *front-end* foi desenvolvido em *React*. A comunicação entre ambos é realizada através de uma API que disponibiliza *RESTful web services*, sendo as páginas geradas do lado cliente.

Em relação à fase de *deployment* da aplicação concebida, foi tirado partido das funcionalidades de virtualização disponibilizadas pelo *Docker*.

1 Introdução

No âmbito das unidades curriculares Arquiteturas Aplicacionais e Sistemas Interativos do perfil de Engenharia de Aplicações, foi nos proposto pela equipa docente que desenvolvéssemos uma aplicação Web, desenvolvida em Java, escalável, responsiva e de alta disponibilidade.

O sistema desenvolvido aborda a gestão e disponibilização de horários de espaços, fornecendo capacidades desde definir a alocação de eventos aos espaços de diferentes edifícios até ao agrupamento de espaços com características comuns.

A interação com a nossa aplicação poderá efetuado de três modos:

- Através de ecrãs públicos colocados junto a cada espaço, que irão disponibilizar o horário associado ao último;
- A partir de ecrãs públicos colocados em espaços comuns, com os quais o utilizador poderá interagir para consultar informação de um espaço em particular;
- Acesso web (por exemplo, via *smartphone*), permitindo procura de eventos/espaços

Ora, começamos então por efetuar a especificação geral do nosso sistema, começando por determinar e analisar os requisitos da mesma. De seguida, identificamos os perfis de utilizadores que vão interagir com a nossa aplicação; abordar com mais detalhe os modos de acesso à aplicação; apresentação dos *mockups* desenvolvidos e respetivo mapa de navegação. De acordo com o contexto da nossa aplicação, determinamos as *frameworks* que mais se adequam às nossas necessidades para o desenvolvimento da primeira. Posteriormente, a partir da especificação efetuada, passamos a modelar o nosso modelo, desde a vertente estrutural até comportamental, construindo modelos cada vez mais especializados para tecnologia adotada. Passamos então à descrição da implementação do nosso sistema e respetivo modo de instalação. Por fim, abordamos a análise de carga que a aplicação desenvolvida suporta.

2 Especificação do sistema

2.1 Levantamento de Requisitos

2.1.1 Requisitos Funcionais

1. O sistema deve permitir o registo de utilizadores e a sua autenticação;
2. Os eventos associados aos espaços poderão ser periódicos ou *one-time events*;
3. Não é permitida a sobreposição de eventos para um dado espaço;
4. Qualquer utilizador poderá consultar uma lista de eventos futuros e a decorrer;
5. Qualquer utilizador poderá consultar os horários relativos aos espaços disponíveis;
6. Os horários dos espaços poderão ser apresentados em diversos formatos, como diariamente, semanalmente, ou mensalmente;
7. Os utilizadores registados devem poder seguir eventos e ser notificados de acordo com o progresso do evento(início, alteração de hora, alteração de local, etc.);
8. A criação/alteração de um evento exigirá uma revisão por parte do gestor de espaços caso o horário de inicio do mesmo seja superior a 1 hora. Para intervalos inferiores a 1 hora, e caso não existam conflitos, o sistema deve automaticamente efetiva a criação/alteração;
9. Ao requisitar alteração da especificação de um dos seus eventos, por parte do utilizador com privilégios de requisição, este último deve ser capaz de modificar toda a informação inerente ao evento, como nome, descrição, agendamento, periodicidade, local, entre outras características;
10. Quando o gestor de espaços altera a informação associada a um evento, o respetivo responsável e seguidores do evento são notificados;

11. O responsável pelo evento pode cancelar o mesmo, sendo os utilizadores interessados notificados;
12. O gestor de espaços poderá gerir os espaços e seus horários, tal como os pedidos de reserva realizados por outros utilizadores;
13. O administrador deverá ter permissões globais e gerir os utilizadores com permissões especiais, nomeadamente os gestores de espaços;
14. O administrador do sistema é responsável por carregar a informação associada aos espaços do contexto onde a aplicação é inserida. Para além disso, poderá também carregar a especificação de eventos, caso exista necessidade;
15. O sistema terá de possibilitar a consulta de um horário específico e interativo para disponibilização junto às salas, que apenas poderá ser configurado por um utilizador com privilégios especiais;
16. O gestor de espaços poderá criar grupos de espaços para que os seus horários sejam disponibilizados em espaços comuns;
17. O utilizador é capaz de navegar nos horários dos espaços contidos num dado espaço comum.

2.1.2 Requisitos Não Funcionais

1. Devem ser colocados ecrãs juntos a cada espaço, apresentando o horário associado ao respetivo espaço;
2. Para espaços comuns, devem ser afixados ecrãs que mostram os horários associados aos primeiros;
3. Idealmente, os ecrãs disponibilizados nos espaços comuns devem permitir a interação por parte do utilizador, para que este consiga tirar partido de todas as funcionalidades associadas à navegação no agrupamento de espaços em questão;
4. O sistema deve apresentar uma taxa de disponibilidade na ordem dos 99.999%.

2.2 Perfis de Utilizador

A partir dos requisitos especificados anteriormente, somos capazes de identificar os diversos tipos de utilizador que fazem parte do nosso sistema, que passamos a descrever de seguida. Por exemplo, tomando como cenário concreto de aplicação do sistema um *Departamento Académico*, somos capazes de estabelecer a seguinte correspondência:

- Visitante - utilizadores que não fazem parte do departamento;
- Utilizador Comum - alunos;
- Utilizador com Privilégios de requisição - docentes do departamento;
- Gestor de espaços - funcionário do departamento responsável pela logística do mesmo;
- Administrador - pessoa pertencente à administração/direção/secretaria do departamento.

2.2.1 Visitante

Podemos considerar o Visitante quem acede por web, mas não está autenticado, e tira partido das funcionalidades de pesquisa disponibilizadas ou quem interage com os ecrãs afixados junto a espaços e espaços comuns.

2.2.2 Utilizador Comum

O utilizador acede e autentica-se na aplicação a partir da Web. Tem acesso às mesmas funcionalidades disponibilizadas para o Visitante e, para além dessas, é capaz de especificar os eventos que lhe interessam, através do mecanismo de *follow*, sendo notificado sempre que um evento muda de lugar/agendamento/é cancelado/etc. .

2.2.3 Utilizador com Privilégios de requisição de espaços

Este tipo de utilizador herda as capacidades descritas para o Utilizador Comum, sendo também capaz de:

- Alocar espaço para novo evento através do preenchimento de um formulário que depois deve ser aprovada pelo Gestor de Espaços (ou pelo sistema nas condições estabelecidas anteriormente);
- Requisitar a alteração da informação associada a um evento que já existe e pelo qual ele é responsável;
- Consultar os seus pedidos efetuados. Para os pedidos que ainda não foram atendidos pelo gestor, é capaz de os anular;
- Possui total liberdade para cancelar qualquer um dos eventos que está a gerir.

2.2.4 Gestor de Espaços

O sistema possui um gestor que dispõem de total autoridade para gerir os eventos dos diversos espaços. É capaz de, livremente, alocar um espaço para novo evento, modificar a especificação de um evento já existente e cancelar qualquer evento.

É o Gestor de Espaços que tem o encargo de atender todos os pedidos efetuados pelos Utilizadores com Privilégios de requisição. Ao aceitar um pedido, o sistema é responsável por torná-lo efetivo, resolvendo quaisquer conflitos; se rejeitar um pedido, o que foi proposto neste último não é concretizado.

Por fim, será o gestor que irá definir os agrupamentos de espaços que constituem os diversos espaços comuns, sendo capaz de modificar a sua especificação sem qualquer restrição.

2.2.5 Administrador

O Administrador do sistema será responsável por instanciar a aplicação para o contexto onde ela vai ser inserida, carregando a informação associada aos espaços, eventuais horários e registar atores do sistema, nomeadamente, definir Utilizadores com Privilégios de requisição de espaços e o Gestor de Espaços.

2.3 Interfaces do Sistema

Passamos então a descrever mais detalhadamente os pontos de acesso da nossa aplicação e respectivas características.

2.3.1 Ecrãs junto a cada espaço

Ecrãs ligados à página web, responsáveis por apresentar o horário semanal de um dado espaço, ou seja, os respetivos eventos, atualizando com o passar das semanas e com eventuais atualização que ocorram aos eventos pertencentes ao mesmo.

2.3.2 Ecrãs em espaços comuns

Ecrãs ligados à pagina web, onde num primeiro modo destaca alguns dos eventos que estão a decorrer atualmente nos espaços que pertencem ao agrupamento, e eventos que irão ocorrer “mais tarde”. Interagindo com o ecrã, o utilizador é capaz de aceder a outro painel, onde será capaz de navegar nos horários dos diferentes espaços que pertencem ao espaço comum, selecionando um espaço em específico.

2.3.3 Interface web

Visitante

- Página inicial;
- Páginas de registo e de autenticação;

- Páginas associadas às funcionalidades de pesquisa de informação de eventos/espaços.

Utilizador Comum

- Listagem dos eventos e espaços existentes, com a capacidade de efetuar *follow* ou *unfollow* de eventos;
- *Dashboard* com informações dos eventos que está a seguir;
- Listagem de todas as notificações recebidas.

Utilizador com Privilégios de requisição de espaços

- Listagem com todos os espaços e possibilidade de requisitar através de preenchimento de um formulário com detalhes sobre o evento;
- Listagem com todos os eventos que criou (e respetivos espaços requisitados), com possibilidade de edição dos detalhes;
- Listagem dos seus pedidos atendidos e pendentes, sendo capaz de cancelar os últimos;
- Páginas acessíveis pelo Utilizador Comum.

Gestor de Espaço

- Página para pesquisar e gerir os diversos eventos existentes no sistema; pesquisar espaços e alocar os mesmo para criar novos eventos;
- Página com listagem pedidos de requisição/alteração pendentes que pode aceitar/recusar;
- Página para visualizar/criar/atualizar/apagar espaços comuns.

Administrador

- Página com listagem de atores do sistema, com capacidade de adicionar novos e gerir respetivos privilégios;
- Página para listagem de espaços/horários, permitindo carregar novos dados.

2.4 Mockups

Ora, esboçamos então as interfaces descritas anteriormente.

Apresentamos na Figura 1 o horário associado a um espaço, a ser exposto num ecrã junto ao espaço em questão.

Gestão de Espaços - Departamento de Informática						09:03 Terça-feira, 25 Março 2018
						DI-0.05
	Segunda-Feira	Terça-Feira	Quarta-Feira	Quinta-Feira	Sexta-Feira	
9:00	Computação Natural T	Arquiteturas Aplicacionais T		Arquitetura e Cálculo T	Interoperabilidade Semânti..	
10:00	Computação Natural TP	Arquiteturas Aplicacionais TP		Arquitetura e Cálculo TP	Interoperabilidade Semânti..	
11:00	Computação Natural TP	Arquiteturas Aplicacionais TP		Arquitetura e Cálculo TP	Interoperabilidade Semânti..	
12:00						
13:00						
14:00	Sistemas Autónomos T	Sistemas Interativos T		Verificação Formal T	Descoberta do Conhecimen..	
15:00	Sistemas Autónomos TP	Sistemas Interativos TP		Verificação Formal TP	Descoberta do Conhecimen..	
16:00	Sistemas Autónomos TP	Sistemas Interativos TP		Verificação Formal TP	Descoberta do Conhecimen..	

Departamento de Informática - Universidade do Minho © 2019

Figura 1: Ecrã junto a espaço

Para o ecrã afixado num espaço comum temos dois modos de apresentação: o destaque de eventos que estão a decorrer no momento e os que vão ocorrer de seguida, como se mostra na Figura 2; o modo de navegação nos horários dos espaços contidos no espaço comum, apresentado na Figura 3.

Gestão de Espaços - Departamento de Informática						09:03 Terça-feira, 25 Março 2018																											
						Eventos																											
						Horários																											
A decorrer																																	
<table border="1"> <tbody> <tr> <td>Programação Imperativa</td> <td>DI-0.05</td> <td>9:00 - 11:00</td> <td>Responsável A</td> </tr> <tr> <td>Sistemas Operativos</td> <td>DI-0.06</td> <td>9:00 - 11:00</td> <td>Responsável B</td> </tr> <tr> <td>Sistemas de Computação</td> <td>DI-0.09</td> <td>9:00 - 11:00</td> <td>Responsável C</td> </tr> <tr> <td>Comunicações por Computador</td> <td>DI-1.03</td> <td>9:00 - 11:00</td> <td>Responsável D</td> </tr> <tr> <td>Programação Orientada aos Objetos</td> <td>DI-1.05</td> <td>9:00 - 11:00</td> <td>Responsável E</td> </tr> <tr> <td>Cálculo de Programas</td> <td>DI-1.09</td> <td>9:00 - 11:00</td> <td>Responsável F</td> </tr> <tr> <td>Laboratórios de Informática IV</td> <td>DI-1.11</td> <td>9:00 - 11:00</td> <td>Responsável G</td> </tr> </tbody> </table>						Programação Imperativa	DI-0.05	9:00 - 11:00	Responsável A	Sistemas Operativos	DI-0.06	9:00 - 11:00	Responsável B	Sistemas de Computação	DI-0.09	9:00 - 11:00	Responsável C	Comunicações por Computador	DI-1.03	9:00 - 11:00	Responsável D	Programação Orientada aos Objetos	DI-1.05	9:00 - 11:00	Responsável E	Cálculo de Programas	DI-1.09	9:00 - 11:00	Responsável F	Laboratórios de Informática IV	DI-1.11	9:00 - 11:00	Responsável G
Programação Imperativa	DI-0.05	9:00 - 11:00	Responsável A																														
Sistemas Operativos	DI-0.06	9:00 - 11:00	Responsável B																														
Sistemas de Computação	DI-0.09	9:00 - 11:00	Responsável C																														
Comunicações por Computador	DI-1.03	9:00 - 11:00	Responsável D																														
Programação Orientada aos Objetos	DI-1.05	9:00 - 11:00	Responsável E																														
Cálculo de Programas	DI-1.09	9:00 - 11:00	Responsável F																														
Laboratórios de Informática IV	DI-1.11	9:00 - 11:00	Responsável G																														
Mais tarde																																	
<table border="1"> <tbody> <tr> <td>Programação Imperativa</td> <td>DI-0.05</td> <td>14:00</td> </tr> <tr> <td>Sistemas Operativos</td> <td>DI-0.08</td> <td>14:00</td> </tr> <tr> <td>Sistemas de Computação</td> <td>DI-0.09</td> <td>14:00</td> </tr> <tr> <td>Comunicações por Computador</td> <td>DI-1.01</td> <td>14:00</td> </tr> <tr> <td>Programação Orientada aos Objetos</td> <td>DI-1.03</td> <td>15:00</td> </tr> <tr> <td>Cálculo de Programas</td> <td>DI-1.05</td> <td>15:00</td> </tr> <tr> <td>Laboratórios de Informática IV</td> <td>DI-1.09</td> <td>15:00</td> </tr> </tbody> </table>							Programação Imperativa	DI-0.05	14:00	Sistemas Operativos	DI-0.08	14:00	Sistemas de Computação	DI-0.09	14:00	Comunicações por Computador	DI-1.01	14:00	Programação Orientada aos Objetos	DI-1.03	15:00	Cálculo de Programas	DI-1.05	15:00	Laboratórios de Informática IV	DI-1.09	15:00						
Programação Imperativa	DI-0.05	14:00																															
Sistemas Operativos	DI-0.08	14:00																															
Sistemas de Computação	DI-0.09	14:00																															
Comunicações por Computador	DI-1.01	14:00																															
Programação Orientada aos Objetos	DI-1.03	15:00																															
Cálculo de Programas	DI-1.05	15:00																															
Laboratórios de Informática IV	DI-1.09	15:00																															
Amanhã																																	
<table border="1"> <tbody> <tr> <td>Programação Imperativa</td> <td>DI-0.05</td> <td>14:00</td> </tr> <tr> <td>Sistemas Operativos</td> <td>DI-0.08</td> <td>14:00</td> </tr> <tr> <td>Sistemas de Computação</td> <td>DI-0.09</td> <td>14:00</td> </tr> <tr> <td>Comunicações por Computador</td> <td>DI-1.01</td> <td>14:00</td> </tr> <tr> <td>Programação Orientada aos Objetos</td> <td>DI-1.03</td> <td>15:00</td> </tr> <tr> <td>Cálculo de Programas</td> <td>DI-1.05</td> <td>15:00</td> </tr> <tr> <td>Laboratórios de Informática IV</td> <td>DI-1.09</td> <td>15:00</td> </tr> </tbody> </table>							Programação Imperativa	DI-0.05	14:00	Sistemas Operativos	DI-0.08	14:00	Sistemas de Computação	DI-0.09	14:00	Comunicações por Computador	DI-1.01	14:00	Programação Orientada aos Objetos	DI-1.03	15:00	Cálculo de Programas	DI-1.05	15:00	Laboratórios de Informática IV	DI-1.09	15:00						
Programação Imperativa	DI-0.05	14:00																															
Sistemas Operativos	DI-0.08	14:00																															
Sistemas de Computação	DI-0.09	14:00																															
Comunicações por Computador	DI-1.01	14:00																															
Programação Orientada aos Objetos	DI-1.03	15:00																															
Cálculo de Programas	DI-1.05	15:00																															
Laboratórios de Informática IV	DI-1.09	15:00																															

Departamento de Informática - Universidade do Minho © 2019

Figura 2: Ecrã em Espaço Comum(Eventos)

Gestão de Espaços - Departamento de Informática						09:03 Terça-feira, 25 Março 2018
		Eventos	Horários			
DI-01		Segunda-Feira	Terça-Feira	Quarta-Feira	Quinta-Feira	Sexta-Feira
DI-02	9:00	Computação Natural T	Arquiteturas Aplicac...		Arquitetura e Cálcul...	Interoperabilidade S...
DI-03	10:00	Computação Natural...	Arquiteturas Aplicac...		Arquitetura e Cálcul...	Interoperabilidade S...
DI-04	11:00	Computação Natural...	Arquiteturas Aplicac...		Arquitetura e Cálcul...	Interoperabilidade S...
DI-05	12:00					
DI-06	13:00					
DI-07	14:00	Sistemas Autónomo...	Sistemas Interativos T		Verificação Formal T	Descoberta do Conh...
DI-08	15:00	Sistemas Autónomo...	Sistemas Interativos...		Verificação Formal TP	Descoberta do Conh...
DI-09	16:00	Sistemas Autónomo...	Sistemas Interativos...		Verificação Formal TP	Descoberta do Conh...
DI-10						
DI-11						

Departamento de Informática - Universidade do Minho © 2019

Figura 3: Ecrã em Espaço Comum(Horários)

Na Figura 4 está apresentada a página inicial da nossa aplicação. É apresentado ao visitante um excerto dos eventos que estão a decorrer no momento que ele acede ao sistema, podendo seguir para o Login, Registo, ou para as funcionalidades de pesquisa.

The screenshot shows the main interface of the application. At the top, there is a header with the title "Gestão de Espaços - Departamento de Informática" and navigation links for "Login" and "Registrar".

The main content is divided into three sections:

- A decorrer:** Shows events currently happening, such as "Programação Imperativa" at 9:00 - 11:00, "Desenvolvimento de Sistemas de Software" at 9:00 - 11:00, "Sistemas de Computação" at 9:00 - 11:30, and "Comunicações por Computador" at 9:00 - 11:00. Each event has a location (e.g., DI-1.09, DI-0.05, DI-0.11, DI-1.12) and a responsible person (e.g., Responsável X, Responsável Y, Responsável Z, Responsável W).
- Mais tarde:** Shows events scheduled for later, such as "Programação Imperativa" at 14:00, "Sistemas Operativos" at 14:00, "Sistemas de Computação" at 14:00, "Comunicações por Computador" at 14:00, "Programação Orientada aos Objetos" at 15:00, "Cálculo de Programas" at 15:00, and "Laboratórios de Informática IV" at 15:00. It includes a navigation bar with pages 1 through 5.
- Amanhã:** Shows events scheduled for the next day, such as "Programação Imperativa" at 9:00, "Sistemas Operativos" at 9:00, "Sistemas de Computação" at 9:00, "Comunicações por Computador" at 9:00, "Programação Orientada aos Objetos" at 11:00, "Cálculo de Programas" at 11:00, and "Laboratórios de Informática IV" at 11:00. It includes a navigation bar with pages 1 through 5.

At the bottom, there is a footer with the text "Departamento de Informática - Universidade do Minho © 2019".

Figura 4: Página Inicial

O visitante pode pesquisar por eventos/espaços, como se mostra na Figura 5, podendo (des)ativar filtros Eventos/Espaços para auxiliar na pesquisa.

Gestão de Espaços - Pesquisar

Gestão de Espaços - Departamento de Informática

Login | Registar

Eventos

Espaços

Programação Imperativa	⌚ DI - 1.09	⌚ 9:00 - 11:00	👤 Responsável X	⭐
Sistemas Operativos	⌚ DI - 0.05	⌚ 9:00 - 11:00	👤 Responsável Y	⭐
Sistemas de Computação	⌚ DI - 1.11	⌚ 9:00 - 11:00	👤 Responsável Z	⭐
Departamento de Informática - 0.05				
Desenvolvimento de Sistemas de Software	⌚ DI - 1.08	⌚ 8:00 - 10:00	👤 Responsável W	⭐
Processamento de Linguagens	⌚ DI - 1.09	⌚ 11:00 - 13:00	👤 Responsável X	⭐
Departamento de Informática - 1.10				
Departamento de Informática - 1.11				
Departamento de Informática - 0.07				
Cálculo de Programas	⌚ DI - 0.07	⌚ 9:00 - 11:00	👤 Responsável A	⭐
Departamento de Informática - 0.09				
Comunicações por Computador	⌚ DI - 0.06	⌚ 14:00 - 16:00	👤 Responsável B	⭐
Departamento de Informática - 1.06				
Laboratórios de Informática IV	⌚ DI - 0.06	⌚ 16:00 - 18:00	👤 Responsável Z	⭐
Programação Orientada aos Objetos	⌚ DI - 1.09	⌚ 9:00 - 11:00	👤 Responsável C	⭐
Departamento de Informática - 2.04				
Programação Funcional	⌚ DI - 1.08	⌚ 10:00 - 12:00	👤 Responsável D	⭐
Laboratórios de Informática I	⌚ DI - 0.04	⌚ 15:00 - 17:00	👤 Responsável D	⭐
Departamento de Informática - 0.06				
Departamento de Informática - 0.02				

Departamento de Informática - Universidade do Minho © 2019

Figura 5: Pesquisar: Visitante

Na página apresentada na Figura 6 o utilizador é capaz de, além de pesquisar por evento/espaço, seguir, ou deixar de seguir, eventos a partir de \star , tendo acesso a mais um filtro pelos eventos que está a seguir.

Gestão de Espaços - Pesquisar

Gestão de Espaços - Departamento de Informática

A seguir

Pesquisar

Notificações

Username

Eventos

Espaços

A Seguir

Programação Imperativa	⌚ DI - 1.09	⌚ 9:00 - 11:00	👤 Responsável X	⭐
Sistemas Operativos	⌚ DI - 0.05	⌚ 9:00 - 11:00	👤 Responsável Y	⭐
Sistemas de Computação	⌚ DI - 1.11	⌚ 9:00 - 11:00	👤 Responsável Z	⭐
Departamento de Informática - 0.05				
Desenvolvimento de Sistemas de Software	⌚ DI - 1.08	⌚ 8:00 - 10:00	👤 Responsável W	⭐
Processamento de Linguagens	⌚ DI - 1.09	⌚ 11:00 - 13:00	👤 Responsável X	⭐
Departamento de Informática - 1.10				
Departamento de Informática - 1.11				
Departamento de Informática - 0.07				
Cálculo de Programas	⌚ DI - 0.07	⌚ 9:00 - 11:00	👤 Responsável A	⭐
Departamento de Informática - 0.09				
Comunicações por Computador	⌚ DI - 0.06	⌚ 14:00 - 16:00	👤 Responsável B	⭐
Departamento de Informática - 1.06				
Laboratórios de Informática IV	⌚ DI - 0.06	⌚ 16:00 - 18:00	👤 Responsável Z	⭐
Programação Orientada aos Objetos	⌚ DI - 1.09	⌚ 9:00 - 11:00	👤 Responsável C	⭐
Departamento de Informática - 2.04				
Programação Funcional	⌚ DI - 1.08	⌚ 10:00 - 12:00	👤 Responsável D	⭐
Laboratórios de Informática I	⌚ DI - 0.04	⌚ 15:00 - 17:00	👤 Responsável D	⭐
Departamento de Informática - 0.06				
Departamento de Informática - 0.02				

Departamento de Informática - Universidade do Minho © 2019

Figura 6: Pesquisar: Utilizador

Ora, a página de pesquisa para um Utilizador com Privilégios de Requisição de espaços apresenta-se na Figura 7, tendo acesso a *shortcuts* para funcionalidades como a alocação de um espaço ou alteração/cancelamento de um dos seus eventos.

Figura 7: Pesquisar: Utilizador com Privilégios de Requisição

Um responsável deve preencher os formulários apresentados nas Figuras 8 e 9 para efetuar o pedido de alocação de espaço e alteração de um dos seus eventos, respetivamente.

Figura 8: Alocar Espaço

Gestão de Espaços

Gestão de Espaços - Departamento de Informática

Pedidos Pesquisar Espaços Comuns Gestor

Editar evento

Nome	Programação Imperativa																																										
Espaço	DI-0.05																																										
Data	<input type="text" value="April 2019"/> <table border="1"> <tr> <th>Mo</th><th>Tu</th><th>We</th><th>Th</th><th>Fr</th><th>Sa</th><th>Su</th></tr> <tr> <td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr> <td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td></tr> <tr> <td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td></tr> <tr> <td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td></tr> <tr> <td>29</td><td>30</td><td></td><td></td><td></td><td></td><td></td></tr> </table>	Mo	Tu	We	Th	Fr	Sa	Su	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30					
Mo	Tu	We	Th	Fr	Sa	Su																																					
1	2	3	4	5	6	7																																					
8	9	10	11	12	13	14																																					
15	16	17	18	19	20	21																																					
22	23	24	25	26	27	28																																					
29	30																																										
Repete	<input type="checkbox"/> Diariamente <input checked="" type="checkbox"/> Semanalmente <input type="checkbox"/> Mensalmente <input type="checkbox"/> Anualmente																																										
Hora de Início	09 : 00																																										
Hora de Fim	11 : 00																																										
Descrição	Aula prática da U.C. de Programação Imperativa do 1.º ano do Mestrado Integrado em Engenharia Informática.																																										

Guardar alterações

Departamento de Informática - Universidade do Minho © 2019

Figura 9: Alteração de Evento

Por fim, o gestor tem acesso aos pedidos efetuados pelos utilizadores com privilégios de requisição, como se mostra na Figura 10, podendo a partir daí consultar a informação associada a cada um deles e aceitar ou rejeitar os mesmos.

Gestão de Espaços

Gestão de Espaços - Departamento de Informática

Pedidos Pesquisar Espaços Comuns Gestor

Gestão de pedidos de requisição

Requisição	Horário	Espaço	Responsável	Aceite	Rejeite
Programação Imperativa	9:00 - 11:00	DI-0.05	Responsável A	✓	✗
Sistemas Operativos	9:00 - 11:00	DI-0.06	Responsável B	✓	✗
Sistemas de Computação	9:00 - 11:00	DI-0.09	Responsável C	✓	✗
Comunicações por Computador	9:00 - 11:00	DI-1.03	Responsável D	✓	✗
Programação Orientada aos Objetos	9:00 - 11:00	DI-1.05	Responsável E	✓	✗
Cálculo de Programas	9:00 - 11:00	DI-1.09	Responsável F	✓	✗

Departamento de Informática - Universidade do Minho © 2019

Figura 10: Gestão de pedidos

2.5 Mapa de Navegação

Desenvolvemos então o seguinte Mapa de Navegação para a nossa aplicação:

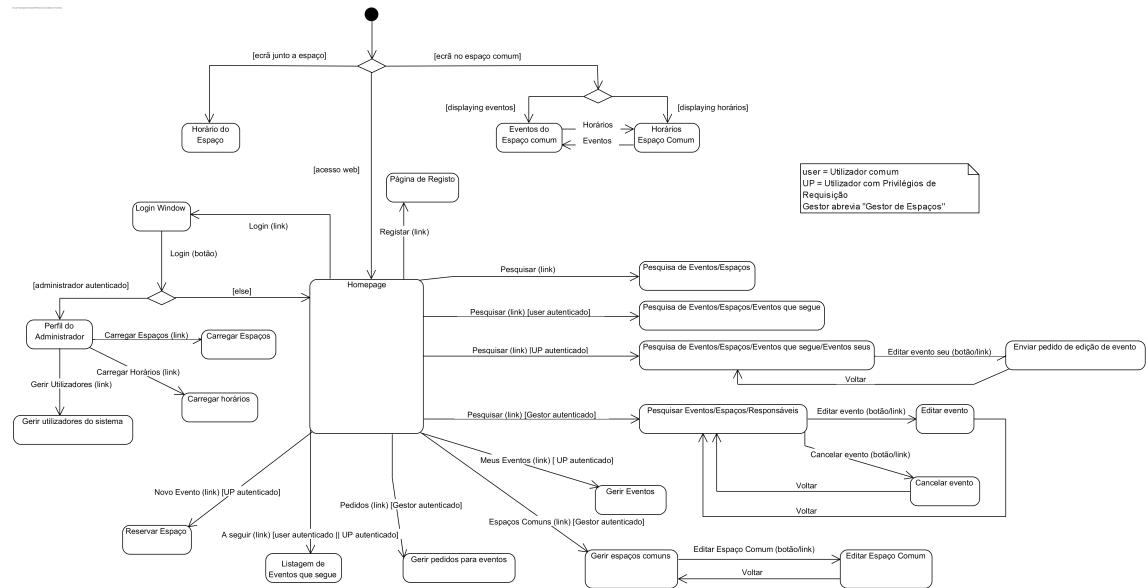


Figura 11: Mapa de Navegação

3 Frameworks

De acordo com o estudo das *frameworks* efetuado anteriormente, bem com o contacto que tivemos com as mesmas ao longo do semestre, terminamos com a seguinte seleção como se apresenta na Figura 12.

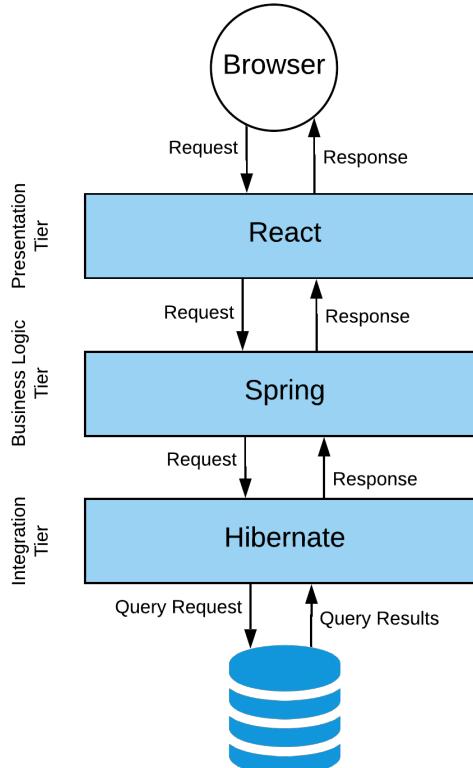


Figura 12: Arquitetura tipo inicial.

Relativamente à camada de dados e de apresentação não mudamos de ideias relativamente à

utilização de *Hibernate* e *Spring*. Em particular, para *Spring*, somos capazes de destacar os seguintes módulos dos quais pretendemos tirar partido:

Spring Data - somos capazes de integrar *Hibernate* às componentes de *Spring* a partir do *JPA* do *Spring Data*;

Spring Container - o **IoC Container** a partir do qual é feita a gestão dos nossos *Spring Beans*;

Spring MVC - permite a definição de *Controladores REST* para construirmos os nossos *Web Services RESTful*;

Spring Security - para tratar da autenticação dos utilizadores, tirando partido de mecanismos como *JWT Tokens*.

Inicialmente, a tinhamos optado por utilizar *Vue* para desenvolver a camada de apresentação, devido à crescente popularidade e baixa curva de aprendizagem. No entanto, após entrar em contacto com *Vue* e *React*, chegamos à conclusão que a curva de aprendizagem de ambas são bastante semelhantes pelo que, dado que *React*, ainda assim, é bastante mais popular que *Vue*, existindo mais suporte da comunidade, a sua utilização tornou-se mais atraente do que o uso de *Vue*, pelo que acabamos por optar por utilizar *React* no nosso projeto. Esta é também uma das *frameworks* com mais *packages* disponíveis no *npm* o que permite que haja uma maior reutilização de funcionalidade já implementada e não obrigue à criação de certos componentes de raiz, como é o caso de gestor de rotas, formulários, notificações, etc.

A nível de funcionamento, cada componente implementa um método *render* que retorna o que se pretende que seja visualizado. Para definição dos elementos visuais recorre-se a *JSX* (*JavaScript XML*), uma combinação de *JavaScript* com *XML*. O uso de *JSX* é opcional mas o seu uso é padrão. Cada componente pode receber dados de outros componentes, dados estes denominados de *props*, pode também manter o seu estado interno que a cada alteração resultará num *trigger* para execução do método de *renderização* do componente. Os componentes são compostos por um grupo de métodos que podem ser implementados para serem ativados ao longo do seu ciclo de vida.

Destes métodos destacam-se os seguintes:

- ***shouldComponentUpdate*** - Permite ao desenvolvedor adicionar lógica para definir se o componente deve ser re-renderizado ou não.
- ***componentDidMount*** - Método chamado assim que o componente é criado e associado ao DOM. Normalmente utilizado para fazer *fetch* de dados de uma API.
- ***componentWillUnmount*** - Método chamado imediatamente antes do componente ser removido do DOM. Pode ser utilizado para forçar a remoção/limpeza de dependências associadas ao componente que não o são automaticamente.

A construção de aplicações modulares é um conceito nuclear do *React*. Esta modularidade pode ser obtida recorrendo à partição bem definida de componentes e permite que recorrendo a uma Virtual DOM apenas os componentes que são alterados e os seus filhos são renderizados de novo, algo que não acontece com o típico DOM, onde cada mudança exige uma renderização completa da página o que implica um maior uso de recursos e maior tempo de processamento.

Assim sendo, adotamos *Hibernate*, *Spring* e *React* como as nossas principais ferramentas auxiliares de desenvolvimento.

4 Modelação

4.1 Modelo de Domínio

Assim, a partir do trabalho desenvolvido para a especificação do sistema, somos capazes de definir o domínio da aplicação através do diagrama apresentado na Figura 13, no qual estão identificados os elementos-chave do nosso sistema e as respetivas relações.

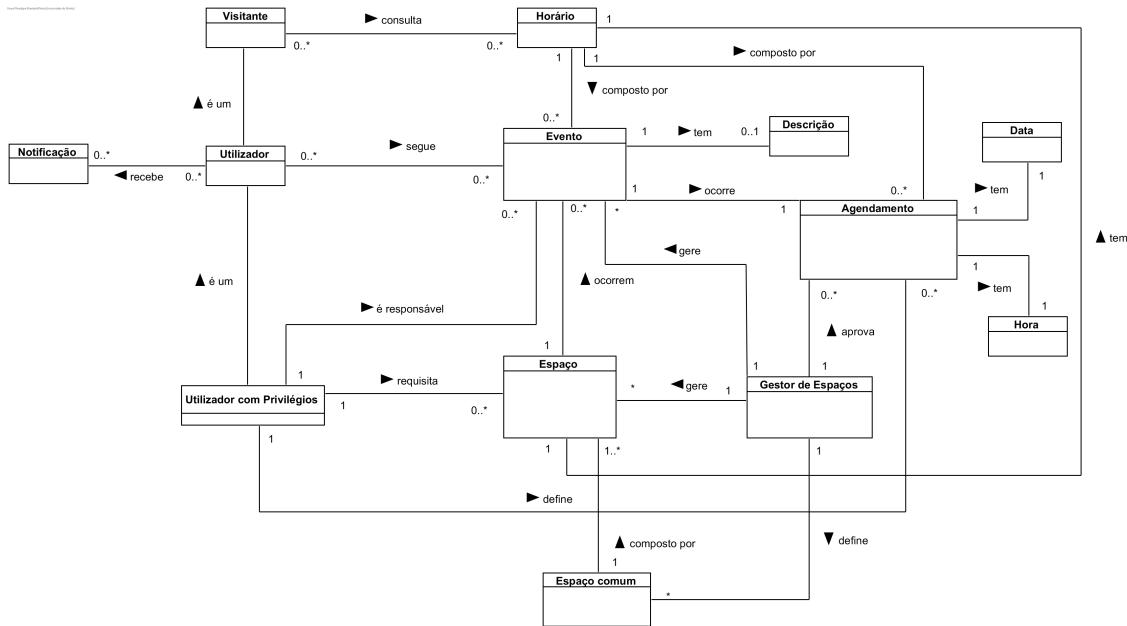


Figura 13: Modelo de Domínio

4.2 Modelação de Tarefas

Das diferenças tarefas possíveis de identificar para o nosso sistema em particular, destacamos algumas delas e construímos o modelo de tarefas associado, recorrendo à abordagem *Hierarchical Task Analysis*.

Apresentamos na Figura 14 o modelo resultante para a tarefa de alocação de um espaço por parte de um utilizador com privilégios de requisição, desde a navegação até encontrar o espaço que pretende alojar até ao envio do pedido construído.

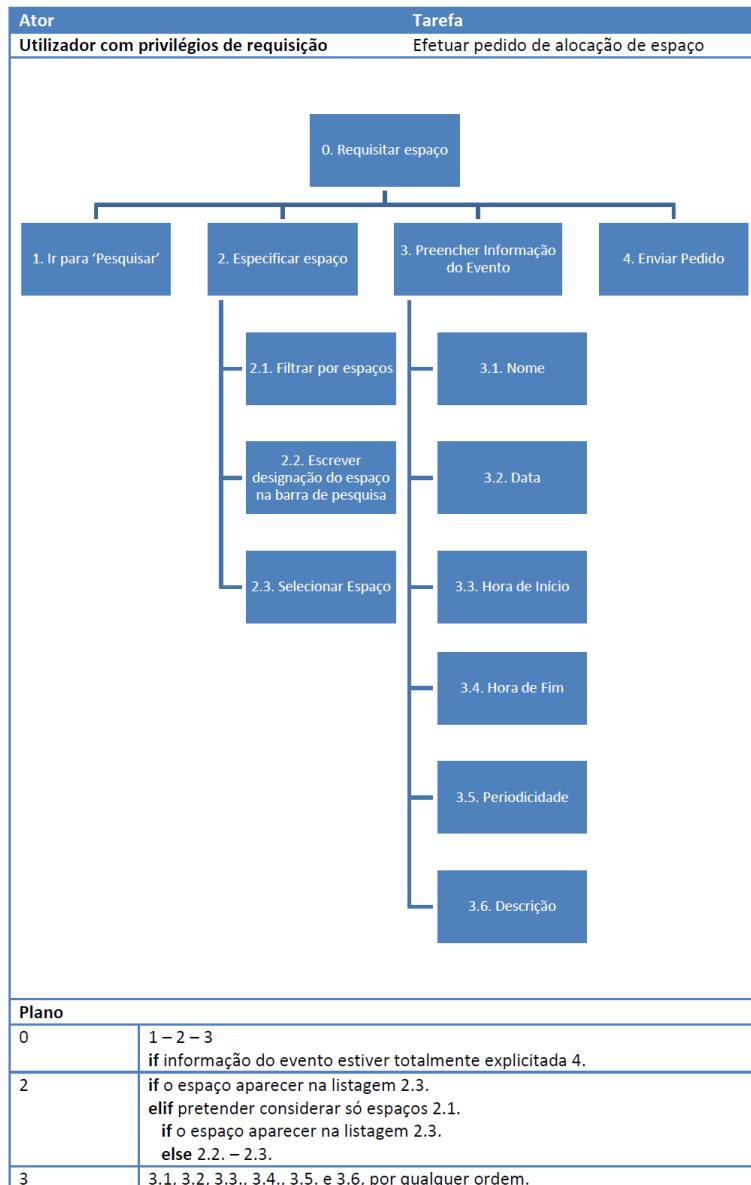


Figura 14: Tarefa: Alocar Espaço

De destacar que no Anexo A estão apresentados os modelos de tarefas para as *Consultar Evento*, por parte de um Visitante, e *Consultar Evento*, efetuada por um Utilizador.

4.3 Modelo de Casos de Uso

Desenvolvemos um diagrama de Use Case, cujos respetivos casos de uso fora diretamente derivados a partir dos requisitos levantados na secção 2.1.1.



Figura 15: Diagrama de Use Case

Consultar Evento (RF4) - obter informação associada um determinado evento;

Consultar Horário de Espaço (RF5) - consultar a agenda de eventos de um dado espaço;

Seguir Evento (RF7) - ficar alerta para obter notificações de um dado evento;

Deixar de seguir Evento (RF7);

Consultar Eventos que segue (RF4, RF7) - listagem de eventos que estão a ser seguidos por este utilizador;

Consultar Notificações (RF7) - observar as notificações recebidas;

Alocar Espaço (RF8) - utilizador com privilégios de requisição aloca um espaço para novo evento;

Requisitar alteração de Evento (RF9) - utilizador com privilégios de requisição efetua um pedido para modificar as informações associadas a um dos seus eventos;

Consultar Pedidos (RF12) - obter a listagem dos pedidos associados a este ator, i.e., pedidos efectuados por ele, no caso de ser um utilizador com privilégios de requisição, ou os pedidos por atender, no caso de se tratar do gestor de espaços;

Cancelar Pedido (RF8, RF9) - utilizador com privilégios de requisição anula um pedido pendente efectuado por ele;

Aprovar Pedido (RF12) - gestor aprova o pedido, o sistema efetiva as alterações, e os interessados são notificados;

Rejeitar Pedido (RF12) - gestor rejeita o pedido, sendo o utilizador que efetuou o pedido notificado;

Novo Evento (RF12) - gestor aloca um espaço para ser realizado um novo evento especificado por ele;

Modificar evento (RF12) - o gestor de espaços altera as informações associadas a um evento já existente;

Consultar Eventos que gere (RF4, RF8, RF12) - listagem dos eventos que estão a ser coordenados por esse ator;

Cancelar Evento (RF11, RF12) - o responsável pelo evento/gestor cancela o evento e o sistema notifica os interessados;

Definir espaço comum (RF16) - gestor define um agrupamento de espaços;

Modificar espaço comum (RF16) - alteração da composição de um espaço comum;

Apagar espaço comum (RF16) - anular o agrupamento de espaços;

Carregar Horários (RF14) - administrador carrega para o sistema informação relativamente aos horários de espaços;

Carregar Espaços (RF14) - é carregada para o sistema a informação associada aos espaços sobre os quais o sistema se insere;

Registrar Atores (RF13) - registar atores no sistema;

Alterar estatuto (RF13) - especificação dos privilégios associados a determinada conta de utilizador;

Efetuar Autenticação (RF1) - autenticação no sistema;

Logout (RF1).

4.3.1 Especificação dos Use Case

Para cada um dos use case identificados anteriormente, desenvolvemos a respetiva especificação, tendo por base o funcionamento especificado nos requisitos que lhe deram origem em conjunto com toda a especificação e modelação efetuada até ao momento.

Por exemplo, consideremos o use case *Alocar Espaço*, destacado na Figura 16, onde se retrata todo o processo de efetuar um pedido de alocação de espaço, tendo em conta todos os cenários estipulados pelos seus requisitos. Em particular, inicialmente o utilizador especifica os detalhes relativamente ao evento que pretende criar, especificando se o evento é periódico ou não, e o sistema adequa-se às diferentes possibilidades, validando as informações obtidas; no momento da confirmação do pedido, se faltar menos de 1 hora para o evento que se pretende agendar, e este não se sobreponha a outros eventos que podem estar a ocorrer nesse espaço, o evento fica efetivamente criado, e o utilizador é notificado da sua criação; senão, o pedido ficará pendente, à espera de ser atendido pelo Gestor de Espaços.

Brief Description	Requisita um espaço.	
Preconditions	Espaço existe.	
Post-conditions		
Flow of Events	Actor Input	System Response
	1 Indica que pretende alocar o espaço para um novo evento.	
	2 Especifica nome e descrição do evento.	
	3	Regista informação.
	4	Pede para indicar se o evento é periódico.
	5 Especifica a periodicidade do evento.	
	6	Processa escolha.
	7	Requer dias e horas em que o evento irá decorrer.
	8 Fornecer agendamento desejado.	
	9	Valida informação
	10	Determina quanto tempo falta para o evento ocorrer.
	11	Regista pedido.
	12	Informa que o pedido foi efetuado com sucesso.
Alternativa 1 [evento periódico] (passo 6)	Actor Input	System Response
	1	Requer os dias de referência, o respetivo horário, e a periodicidade associada.
	2 Especifica agendamento.	
Exception 1 [agendamento inválido] (passo 9)	Actor Input	System Response
	1	Informa que os horários especificados são inválidos.
Alternativa 2 [menos de 1 hora para o evento && não existe conflito com outro evento marcado] (passo 10)	Actor Input	System Response
	1	Regista alocação.
	2	Informa que o agendamento está efetivado.

Figura 16: Especificação do UC: Alocar Espaço

A especificação dos restantes casos de uso encontra-se no Anexo B.

4.3.2 Diagramas de Sequência de Sistema

Derivamos cada diagrama de sequência de sistema associado à respetiva especificação, destacada na secção anterior, obtendo, assim, noção temporal no proceder das ações. Podem ser consultados no Anexo C.

Considerando ainda o use case *Alocar Espaço*, verificamos que a sua especificação deu origem ao diagrama de sequência de sistema apresentado na Figura 17.

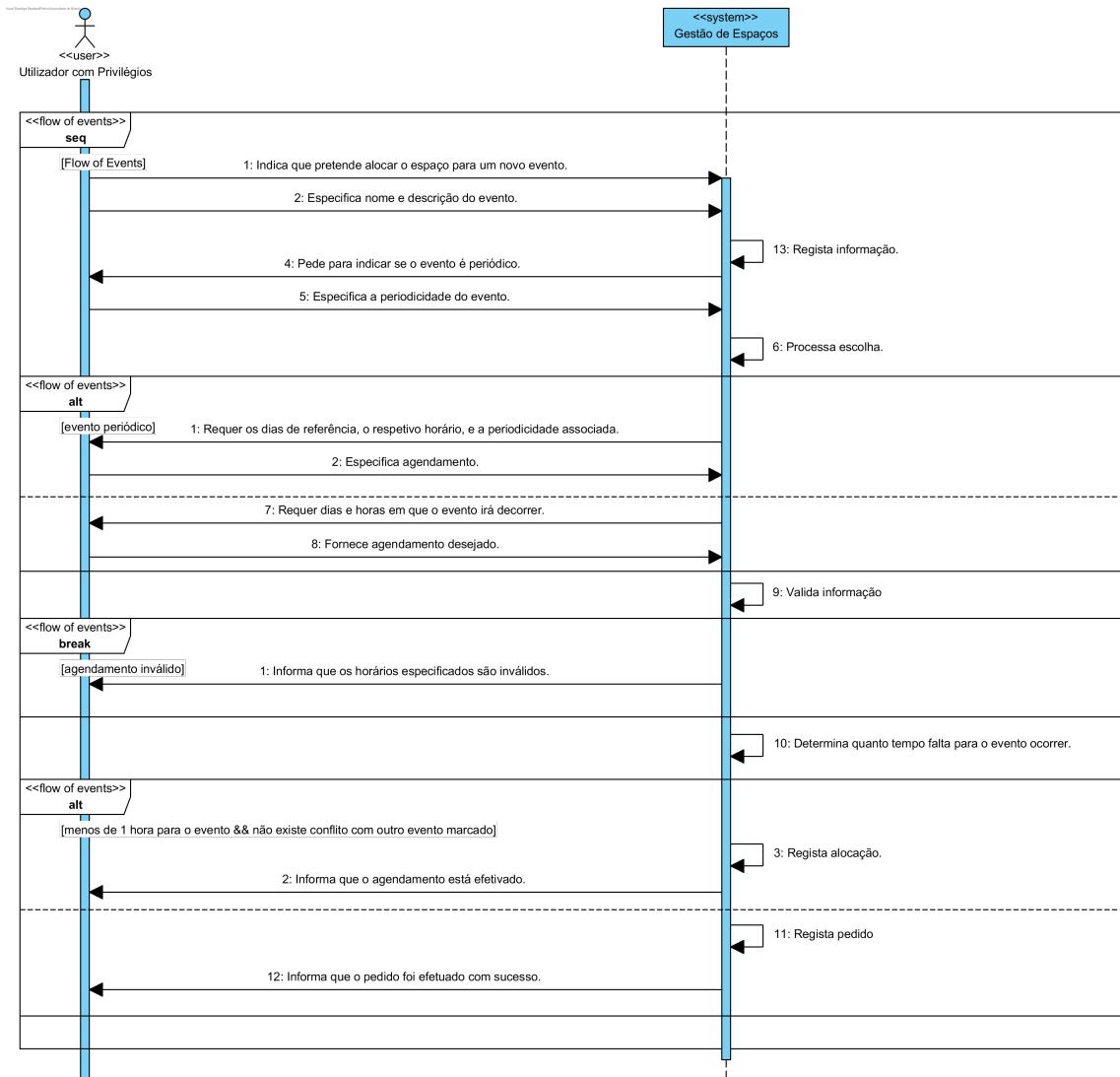


Figura 17: Diagrama de Sequência de Sistema: Alocar Espaço

Fornecendo uma melhor visão relativamente ao fluxo da interação entre o cliente e o sistema, explicitando temporalmente a ocorrência dos patamares de decisão quanto à especificação de um evento periódico ou não e relativamente à condição temporal, menos de 1 hora até ao agendamento e não causa conflito, da alocação automática.

4.4 Platform Independent Model

Com o estudo efetuado até ao momento, estamos então em condições de desenhar o nosso *Platform Independent Model* para suportar a lógica de negócio do nosso sistema, independentemente da tecnologia adotada na fase de desenvolvimento.

Apresentamos o **PIM** desenvolvido na Figura 18, constituído por:

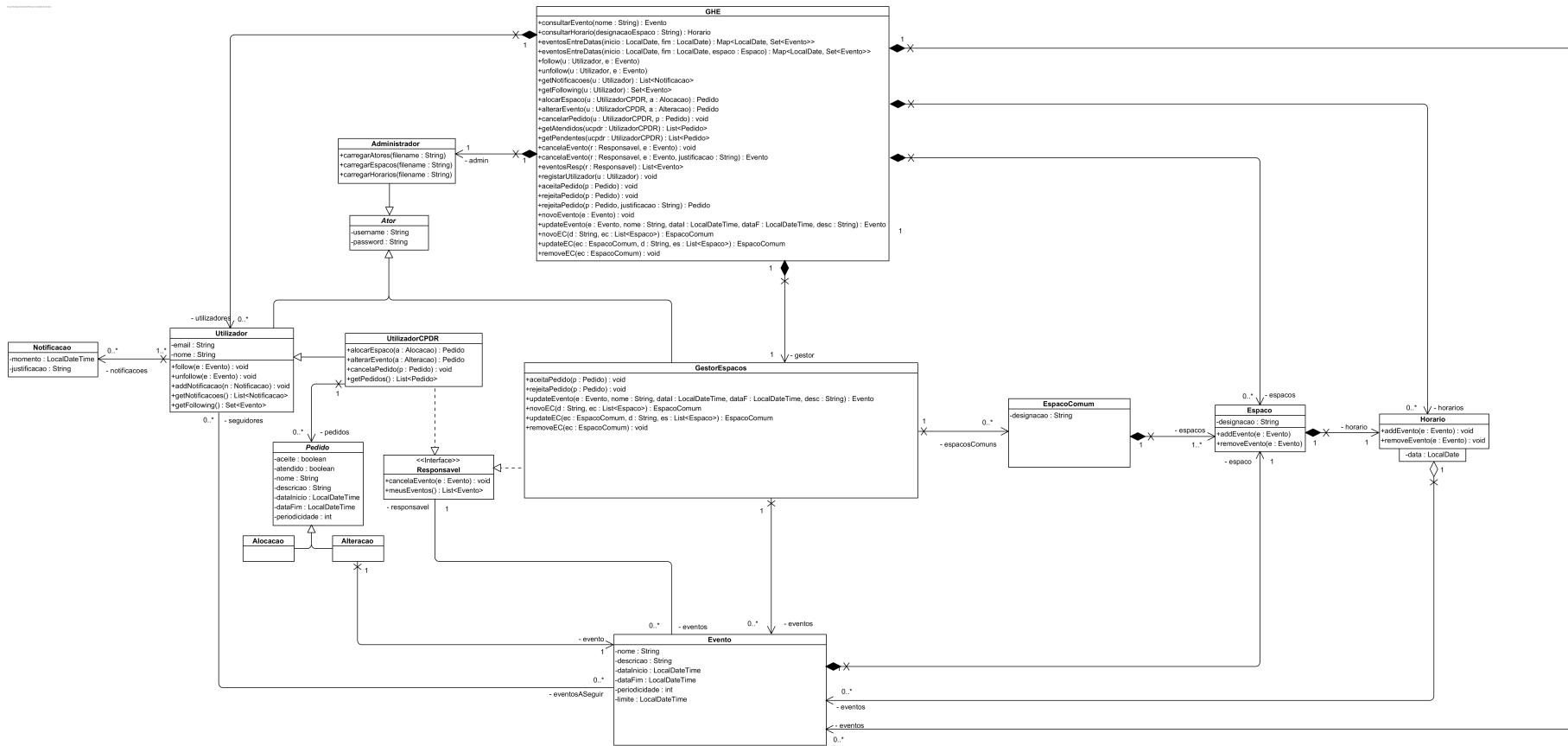


Figura 18: PIM

- **Autor** - para representar os diferentes tipos de utilizador que interagem com o nosso sistema, identificado por `username` e `password`;
 - **Administrador** - onde definimos o comportamento permitido pelo administrador do sistema;
 - **Responsavel** - interface para especificar operações comuns a **UtilizadorCPDR** e **GestorEspacos** que decorrem de forma distinta, face à diferença de permissões. Consideramos que cada responsável tem acesso aos **Evento** pelos quais é responsável;
 - **GestorEspacos** - o gestor de espaços, que tem acesso a todos **Evento** existente na aplicação, bem como os **EspacoComum** definidos, possuindo capacidades para os manipular;
 - **Utilizador** - Utilizador Comum, representado também pelo seu `nome` e `email`. Consideramos o relacionamento `eventosASeguir` para referenciar os eventos que este utilizador está a seguir e `notificacoes` para identificar as suas `Notificacao`;
 - **UtilizadorCPDR** - Utilizador com Privilégios de requisição de espaço, para além do que herda de **Utilizador**, tem acesso aos **Pedido** efetuados por ele, e aos **Evento** pelos quais é responsável (de acordo com a interface **Responsavel**).
- **Espaco** - identificado pela sua `designacao` e respetivo **Horario**;
- **Horario** - define o horário de um dado espaço, sendo constituído pelos **Evento** que ocorrem nesse **Espaco**;
- **Evento** - identifica um evento, contendo toda a sua informação associada, `nome`, `descricao`, data de inicio, de fim, período, e data limite até quando o evento se repete. É sempre composto por um **Espaco**, local onde ocorre, tendo acesso ao **Responsavel** por ele, e aos seus seguidores.
- **Pedido** - um pedido dispõe de toda a informação necessária para definir um evento, podendo ser de um de dois tipos concretos:
 - **Alocacao** - se for um pedido de criação de novo evento;
 - **Alteracao** - que representa um pedido de alteração de um evento já existe, pelo que possui uma referência ao **Evento** que irá, potencialmente, ter as suas informações atualizadas.
- **EspacoComum** - definido pelo gestor, tem uma `designacao` e é composto por diversos **Espaco**;
- **Notificacao** - contém o conteúdo da notificação e o momento em que foi enviada.
- **GHE** - corresponde à nossa *Facade*, constituída por todos os métodos que tiveram origem dos *use case* definidos na secção 4.3. Esta é composta por todos os utilizadores, administrador, gestor, eventos, espaços e horários do sistema.

4.5 Platform Specific Models

Ora, tendo em conta a tecnologia adotada descrita na secção 3, começamos por construir um **PSM** tendo em conta *Hibernate*, anotando as devidas classes a persistir com *ORM Persistable*, adaptando os métodos e referências para se adaptarem ao modelo *ORM*, resultando no modelo apresentado na Figura 19.

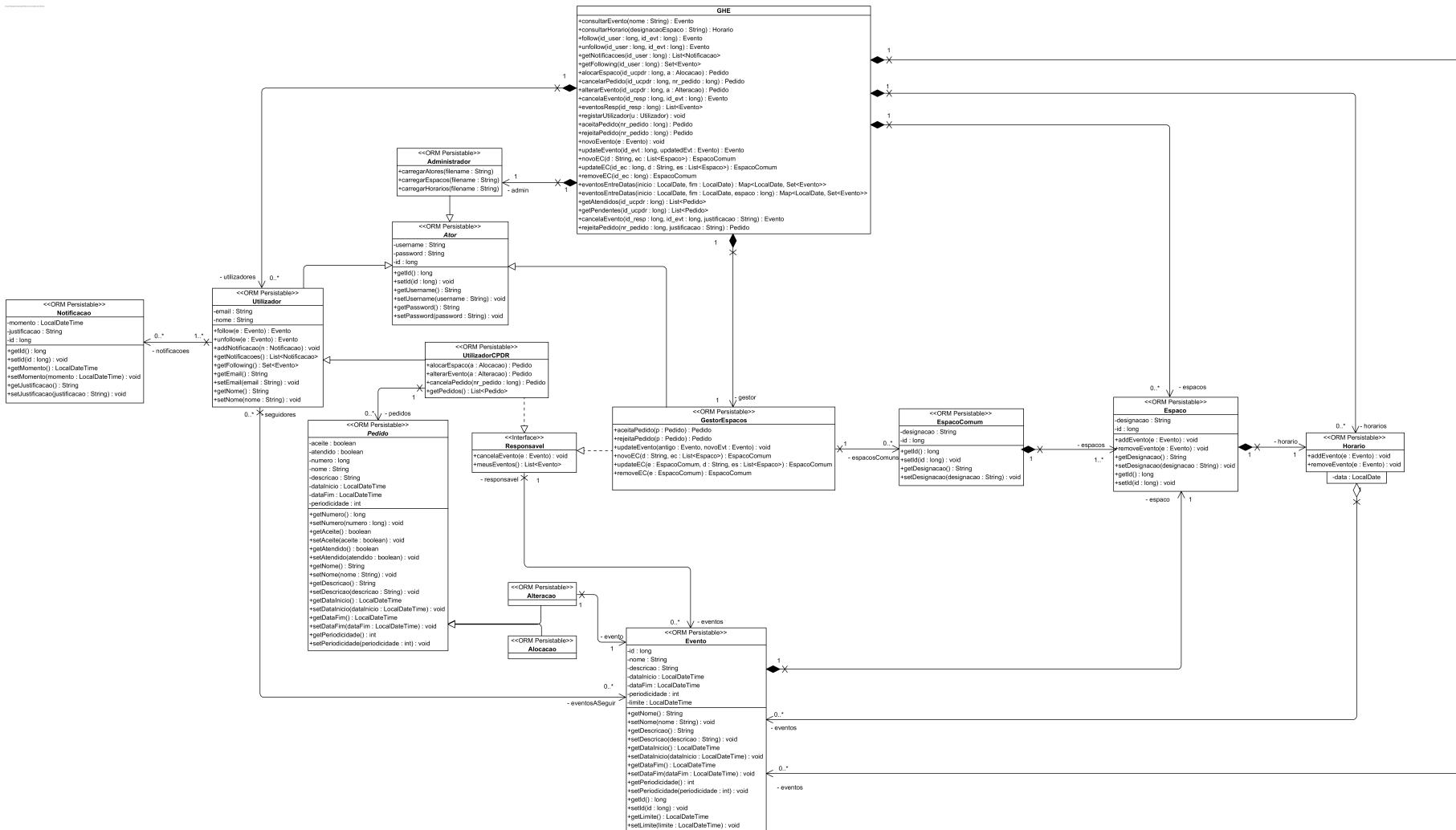


Figura 19: Primeira versão do PSM, identificando classes a persistir.

Por outro lado, estamos interessados em definir os nossos *Stateless Session Spring Beans* de forma a tirar partido do *Spring Container* para promover a escalabilidade da nossa aplicação. Para tal, definimos quatro tipos de *Spring Beans*:

VisitanteBean - responsável por tratar das funcionalidades associadas à pesquisa e consulta de eventos, espaços e seus horários;

UtilizadorBean - para implementar o mecanismo de *follow*, notificações e registo dos utilizadores no sistema;

ResponsavelBean - suporta a lógica associada às funcionalidades da construção e gestão de pedidos por parte do utilizador com privilégios de requisição, bem como funcionalidades inerentes ao **Responsavel**, como gerir os pedidos pelos quais é responsável e cancelar os seus eventos;

GestorBean - contém as funcionalidades associadas à gestão e administração da aplicação. Responde aos pedidos do gestor para aceitar/rejeitar pedidos, criar/modificar/cancelar eventos e agrupar/atualizar/apagar espaços comuns.

Assim sendo, GHE passa a tirar partido desses *beans* para fornecer a API do nosso sistema. Chegamos, assim, ao PSM apresentando na Figura 20.

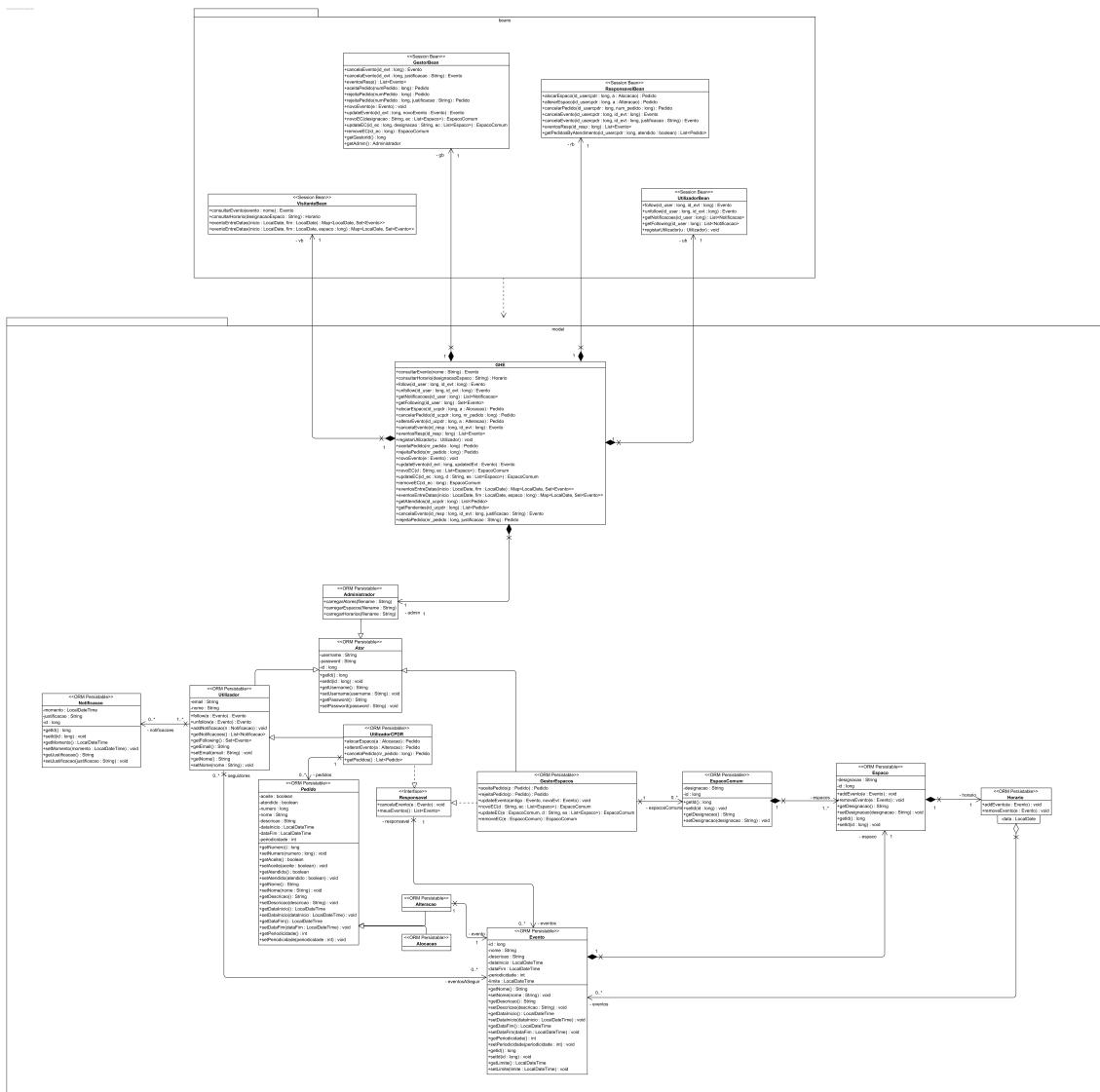


Figura 20: PSM de acordo com as *frameworks* utilizadas.

5 Implementação do Sistema

Terminada a especificação e modelação do sistema, estamos, então, em condições de desenvolver o nosso sistema recorrendo às ferramentas estipuladas na secção 3. Passamos, então a descrever a implementação da aplicação nas vertentes consideradas.

5.1 Server-side

5.1.1 Model

A partir do PSM, ver Figura 20, derivamos de imediato as classes que definem a estrutura da nossa aplicação e respetivos métodos.

5.1.2 Camada de Dados

De seguida, optamos por utilizar *Anotações Hibernate* para especificar a camada de dados. Preferimos esta abordagem, pois permitiu mais facilmente integrar *Hibernate* com *Spring*, recorrendo aos *JPARepository* para servir de ponte entre as duas camadas.

Ora, para cada classe a persistir anotamos com `@Entity` e especificamos o respetivo `@Id`. A presença de herança fez com que tivéssemos de adotar um mecanismo para lidar com ela:

- Para `Autor` e respetivas subclasses, optamos pela estratégia `InheritanceType.TABLE_PER_CLASS`, i.e., uma relação por classe concreta;
- Relativamente a `Pedido`, escolhemos utilizar `InheritanceType.SINGLE_TABLE` com `DiscriminatorType.CHAR`, ou seja, `Alocacao` e `Alteracao` são armazenados na mesma tabela, tal que, "A" identifica `Alocacao` e "U" representa `Alteracao`.

5.1.3 Classes de Serviço

Estamos em condições de implementar a lógica da aplicação, tirando partido de `Model` e das classes da camada de dados, especificando as nossas classes de serviço como *Stateless Session Spring Beans*, conforme se mostra para `VisitanteBean`:

```
@Service  
@Scope(value= ConfigurableBeanFactory(SCOPE_SINGLETON)  
public class VisitanteBean{...}
```

Tendo desenvolvido os nossos *Spring Beans*, estamos em condições de disponibilizar a nossa API, recorrendo à nossa facade `GHE`, que passamos a descrever:

- `Evento consultarEvento(String nome)` - Obter um evento com o nome fornecido, a existir;
- `Horario consultarHorario(String designacaoEspaco)` - Obter o horário associado ao espaço com determinada designação, a existir;
- `Map<LocalDate, Set<Evento>> eventosEntreDatas(LocalDate inicio, LocalDate fim)` - Obter todos os eventos entre datas, agrupados por data;
- `Map<LocalDate, Set<Evento>> eventosEntreDatas(LocalDate inicio, LocalDate fim, long espaco)` - Obter todos os eventos de um espaço entre duas datas, agrupados por data;
- `Evento follow(long id_user, long id_evt)` - Utilizador passa a seguir o evento;
- `Evento unfollow(long id_user, long id_evt)` - Utilizador deixa de seguir o evento, se o estava a seguir previamente;
- `Set<Notificacao> getNotificacoes(long id_user)` - Obter as notificações recebidas pelo utilizador;

- `Set<Evento> getFollowing(long id_user)` - Determina o conjunto de eventos que o utilizador especificado segue;
- `Pedido alocarEspaco(long id_usercpdr, Alocacao a)` - O utilizador com privilégios de requisição efetua o pedido de alocação de espaço, a ser aprovado pelo gestor. Caso falte menos de 1 hora para o inicio do evento e este último não causar conflito com outros, será automaticamente efetivado;
- `Pedido cancelarPedido(long id_usercpdr, long nr_pedido)` - Responsável decidiu cancelar um pedido pedente efetuado por ele;
- `Pedido alterarEvento(long id_usercpdr, Alteracao a)` - O utilizador com privilégios de requisição efetua um pedido de alteração de um evento pelo qual é responsável. Se faltar menos de 1 hora para o inicio proposto deste evento, e a sua nova especificação não causar conflito com outros, será automaticamente aceite. Senão, ficará à espera de atendimento por parte do gestor;
- `List<Pedido> getAtendidos(long id_usercpdr)` - Determina os pedidos atendidos efetuados por este utilizador c/ privilégios. Se não existir nenhum utilizador com esse identificador associado, é retornada a lista vazia;
- `List<Pedido> getPendentes(long id_usercpdr)` - Determina os pedidos pendentes efetuados por este utilizador c/ privilégios. Se não existir nenhum utilizador com esse identificador associado, é retornada a lista vazia;
- `Evento cancelaEvento(long id_responsavel, long id_evt, String justificacao)` - O responsável pelo evento cancela o mesmo, notificando os seguidores;
- `Evento cancelaEvento(long id_responsavel, long id_evt)` - O responsável pelo evento cancela o mesmo, notificando os seguidores. Não fornece uma justificação personalizada;
- `Evento cancelaEvento(long id_evt)` - O gestor cancela o evento, notifica o responsável e seguidores desse evento, não fornecendo uma justificação personalizada.
- `Evento cancelaEvento(long id_evt, String justificacao)` - O gestor cancela o evento, notifica o responsável e seguidores desse evento;
- `List<Evento> eventosResp(long id_resp)` - Determina os eventos que são geridos pelo responsável especificado;
- `void registarUtilizador(Utilizador u)` - Regista utilizador no sistema;
- `Pedido aceitaPedido(long nr_pedido)` - Gestor aprova um pedido, sendo efetuadas as devidas alterações. Qualquer evento que cause conflito com o novo, é cancelado e os interessados notificados;
- `Pedido rejeitaPedido(long nr_pedido)` - Gestor rejeita o pedido e notifica o responsável, sem justificar;
- `Pedido rejeitaPedido(long nr_pedido, String justificacao)` - Gestor rejeita o pedido e notifica o responsável que o efetuou;
- `void novoEvento(Evento e)` - Gestor cria um novo evento. Qualquer evento que entre em conflito com o novo, é cancelado;
- `Evento updateEvento(long id_evt, Evento novoEvento)` - Gestor atualiza um evento já existente. Qualquer evento que entre em conflito com o eventual novo agendamento, é cancelado. Notifica os interessados das alterações;
- `EspacoComum novoEC(String designacao, List<Espaco> ec)` - Define um novo espaço comum;
- `EspacoComum updateEC(long id_ec, String designacao, List<Espaco> ec)` - Atualiza a composição de um espaço comum;

- `EspacoComum removeEC(long id_ec)` - Descarta o agrupamento de espacos especificado;
- `void carregarAtores(String filename)` - Administrador regista atores no sistema;
- `void carregarEspacos(String filename)` - Administrador carrega especificação dos espaços sobre os quais o sistema está a ser aplicado;
- `void carregarHorarios(String filename)` - São carregados para o sistema horários associados a espaços, pelo administrador.

5.1.4 Web Services REST

Vamos definir *Controladores RESTful*, aos quais o *front-end* deve enviar os pedidos para tirar partido das capacidades do servidor. Para tal, tiramos partido de `@RestController` do *Spring* para os definir, estes serão identificados pelo respetivo URL, especificado com `@RequestMapping`, onde disponibilizamos a nossa API a partir de pedidos GET e POST, especificados por `@GetMapping` e `@PostMapping`, respetivamente.

Ora, definimos os seguintes controladores:

VisitanteController - Estão disponibilizados os recursos do servidor para os quais não é necessário o ator estar autenticado para tirar partido dos mesmos, entre eles:

- Registo de conta de utilizador;
- Autenticação no sistema;
- Consulta de eventos, horários, espaços, espaços comuns, etc..

UtilizadorController - Funcionalidades associadas a um Utilizador Comum, ou com Privilépios de Requisição:

- Seguir ou deixar de seguir eventos;
- Consultar eventos que está a seguir;
- Consultar as suas notificações recebidas.

UtilizadorCPDRController - Aqui são fornecidas as funções associadas ao Utilizador com Privilépios de Requisição de espaços, ou seja:

- Efetuar pedido de alocação de espaço;
- Efetuar pedido de alterar de um dos seus eventos;
- Consultar os seus pedidos pendentes/atendidos;
- Cancelar um pedido pendente;
- Obter os eventos pelos quais é responsável;
- Cancelar um evento que gere.

GestorController - Neste ponto são disponibilizadas as funcionalidades para o gestor administrar os espaços:

- É capaz de criar/alterar/cancelar um evento;
- Obter os pedidos pendentes;
- Aceitar/Rejeitar um dado pedido;
- Agrupar/Alterar/Desagrupar um espaço comum.

De notar que para os pedidos a qualquer um dos controladores (exceto `VisitanteController`) serem atendidos, os atores devem estar devidamente autenticados. Para tal, tiramos partido de *Spring Security* para estabelecer uma ligação segura entre cliente e servidor a partir de *JSON Web Tokens*. No momento que o ator se autentica com sucesso, efetuando pedido de *login* ao `VisitanteController`, é produzido um *JWT Token* que deverá ser utilizado para garantir acesso a funcionalidades fornecidas pelos restantes controladores.

5.1.5 Integração com API Externa

A API externa que decidimos integrar no nosso projeto corresponde à Google Calendar API[2].

Assim, disponibilizamos a utilizador a capacidade de, caso esteja interessado, sincronizar o agendamento dos eventos que está a seguir com o calendário associado à sua conta Google.

Para tal, devemos seguir o *workflow* apresentado na Figura 21[3].

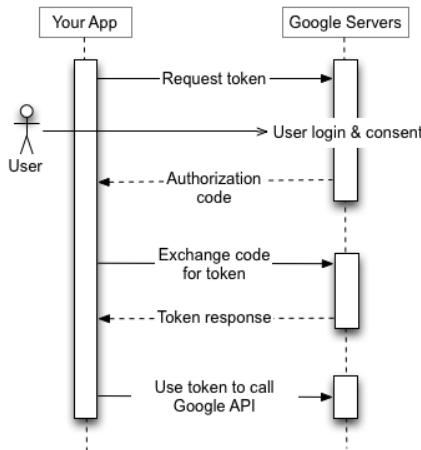


Figura 21: Workflow de acesso à Google API.

Assim sendo, inicialmente o utilizador terá de se autenticar na sua conta Google, e dar-nos permissões para aceder aos seus calendários. Recebido o código de autorização, deverá fornecer à nossa aplicação, do lado do cliente, que posteriormente enviará o pedido ao servidor com esse código. A partir desse código somos capazes de ganhar acesso ao calendário desse utilizador. De seguida, obtemos os eventos que este utilizador está a seguir e, para cada um deles, criamo um evento no seu *Google Calendar* com informação do evento em questão, recorrendo à API. No final, o calendário do utilizador irá conter os agendamentos de todos os eventos que estava a seguir no momento da sincronização.

Assim sendo, adicionamos à nossa *Facade* o método `boolean syncCalendar(long id, String code)` e ao Web Service `UtilizadorController` um recurso para aceitar pedidos com essa informação.

5.2 Client-side

Tal como indicado anteriormente, a implementação do *client-side* foi feita recorrendo ao *React*. A melhor forma de tirar partido desta *framework* é recorrendo a uma implementação de componentes pequenos e bem definidos relativamente à sua função. Componentes pequenos permitem uma maior reutilização e configuração facilitando assim o processo de desenvolvimento.

A interação com o servidor de *backend* por *REST* foi feita recorrendo ao *axios*, uma dependência que permite a realização de pedidos *HTTP*, sendo estes pedidos organizados por vários utilitários de acordo com a sua funcionalidade.

Segue-se um exemplo da definição de um método na `ApiPedidos` que para além do pedido *GET* envia também o *token* para autenticação do utilizador.

Exemplo ApiPedidos

```
const token = {
  "headers": {
    "Authorization": `Bearer ${UserHandler.getToken()}`
  }
}

ApiPedidos.fetchPedidosPendentes = async () => {

  try {
    const req = await axios.get(`.${HOST}/usercpdr/pedidos/
pendentes/${userData.id}`, token);
    return req.data;

  } catch (e) {
    console.error(e);

    return {
      success: false,
    }
  }
}
```

Sendo a aplicação composta por várias interfaces relacionadas com apresentação de horários, a utilização de um componente já existente para a apresentação de diferentes vistas do horário foi crucial para acelerar o processo de desenvolvimento. Os horários são estruturas complexas e adicionando o fator interface e necessidade de se ajustar a ecrãs de diferentes resoluções tornam-se ainda mais complexas.

Desta forma, o uso do *FullCalendar*, um componente que permite uma enorme configuração: desde a sua interface, às vistas e eventos por ele apresentados, permitiu a apresentação de uma interface consistente e responsiva.

Segue-se o exemplo da criação de um *wrapper* para apresentação de uma vista semanal com eventos:

Vista semanal

```
class WeeklyCalendar extends Component {
    constructor(props) {
        super(props);
        this.state = {}
    }

    render() {
        return (
            <div style={{ marginBottom: '25px' }}>
                <FullCalendar defaultView="timeGridWeek"
                    allDaySlot={false}
                    plugins={[timeGridPlugin]}
                    slotLabelFormat={{
                        hour: '2-digit',
                        minute: '2-digit',
                        omitZeroMinute: false
                    }}
                    minTime={'08:00:00'}
                    maxTime={'20:00:00'}
                    events={this.props.events}
                    locale='pt'
                    contentHeight='auto'
                    eventColor="#ddd"
                    slotEventOverlap={true}
                /> </div>
        );
    }
}

export default WeeklyCalendar;
```

Na implementação do *client-side* existiram essencialmente 2 fases:

- Implementação de Interface:
 - Nesta fase recorreu-se à implementação da estrutura da interface e do seu estilo recorrendo a dados fictícios;
- Integração com dados:
 - Já com a interface implementada realizou-se a criação dos métodos para interação com o *backend* e realizaram-se as adaptações necessárias aos dados recebidos;

6 Deployment da Aplicação

Um dos principais fatores que se tem em conta na realização do *deploy* é como se pode criar consistência neste processo, ou seja, espera-se que o sistema em modo de desenvolvimento quando realizado o *deploy* tenha exatamente o mesmo comportamento. Uma forma de obter esta consistência é recorrer a tecnologias de virtualização que permitam criar ambientes de raiz apenas com as dependências estritamente necessárias.

6.1 Docker

Para criar tal consistência recorremos ao *Docker*, tecnologia que fornece uma camada de abstração e automatização de virtualização ao nível do sistema operativo, recorrendo a características do *kernel Linux*, como o *cgroups* para a realização do isolamento dos *containers*.

A criação destes *containers* dá-se inicio pela criação de um ficheiro de configuração onde o programador define um conjunto de instruções a serem realizadas, denominado de *Dockerfile*. Estas instruções permitem definir que dependências/*software* instalar, configurações da máquina e que serviço será exposto.

Realizada a definição do conjunto de instruções e o respetivo *build*, é obtida uma imagem. A imagem refere-se ao produto final obtido pela execução e configuração definida no *Dockerfile*, podendo esta ser reutilizada como imagem base para outras *builds* ou para inicialização de múltiplos *containers*.

Os principais processos a necessitar de virtualização para este projeto são o *backend*, o *frontend* e a base dados *MySQL*

Começando pela definição do *Dockerfile* para o *backend*, recorreu-se a uma imagem base do *ubuntu* onde são instalados os restantes componentes ao funcionamento do servidor como é o caso do *Maven*, para instalação das dependências e inicialização do processo e as bibliotecas de *Java*.

```
Dockerfile Servidor Backend
FROM ubuntu:18.04

WORKDIR /server

COPY . .
COPY ./wait-for-it.sh .
RUN chmod +x wait-for-it.sh

RUN apt-get update
RUN apt-get install openjdk-8-jdk-headless -y
RUN apt-get install openjdk-8-jre-headless -y
RUN apt-get install maven -y
```

Relativamente à dockerização do *Frontend* recorreu-se a uma imagem base já composta pelo *node* e o *npm*, sendo assim apenas necessário copiar o projeto para o *container*, realizar a instalação das dependências e iniciar o servidor.

```
Dockerfile Frontend
FROM node:8

RUN apt-get update
WORKDIR /app

COPY . .

ENV PATH /app/node_modules/.bin:$PATH
RUN npm install

EXPOSE 3000
CMD ["npm", "start"]
```

A *dockerização* do *MySQL* é realizada no *docker-compose* dada a facilidade com que é aí feita, recorrendo a uma imagem base disponibilizada no *DockerHub*.

6.2 Docker-Compose

O foco do *Docker* passa pela criação de ambientes isolados e consistentes para disponibilização de serviços, no entanto, após esta criação é necessário realizar-se a orquestração entre os vários serviços expostos.

É neste processo de orquestração que o *docker-compose* atua, permitindo definir que *containers* serão inicializados e a partir de que imagens, e configurar a interação entre os mesmos, tipicamente recorrendo a variáveis de ambiente que farão o *overwrite* de configurações de ligação (ex.: *string de conexão do mysql*)

De forma homóloga ao *Docker* o *Docker-Compose* recorre a um ficheiro *yml* para definição dos *containers* a executar e as suas configurações. Para cada um dos *containers* que se pretende executar define-se um serviço. A cada serviço está associado o local do respetivo *Dockerfile* para *build* ou alternativamente uma imagem já existente localmente ou num repositório *online*, tipicamente o *DockerHub*.

As variáveis de ambiente permitem definir a interligação entre os componentes uma vez que o *Docker* faz o mapeamento entre os nomes dos serviços e os respetivos *IPs* para as máquinas que se encontram na mesma rede.

É também importante definir as dependências entre serviços para controlar a ordem de criação e inicialização dos *containers*, recorrendo ao *depends_on*. Apesar do *depends_on* criar esta dependência e realizar a execução pela ordem das dependências, este não garante que quando o serviço que cria a dependência é exposto, os seus dependentes já se encontrem prontos. É assim necessário recorrer a um *script*, *wait-for-it.sh*, que espera até que o serviço se encontre pronto, executando de seguida o comando para inicialização do serviço atual.

Segue-se o *docker-compose* definido para o *deploy* de toda a infraestrutura:

```

docker-compose
version: '3.3'

services:

  db_gestorespacos:
    image: mysql:5.7.26
    environment:
      MYSQL_DATABASE: db_gestorespacos
      MYSQL_USER: gestorespacos
      MYSQL_PASSWORD: gestorespacos
      MYSQL_ROOT_PASSWORD: root-password
    ports:
      - 3300:3306
    networks:
      - network_backend

  frontend:
    build: ./client
    ports:
      - 3333:3000

  backend:
    build:
      context: ./server
      command: ["./wait-for-it.sh", "db_gestorespacos:3306",
                "--", "mvn", "install"]
      command: ["mvn", "spring-boot:run"]

    environment:
      WAIT_HOSTS: mysql:3306
      SPRING_DATASOURCE_URL: jdbc:mysql://db_gestorespacos:
            3306/db_gestorespacos?useUnicode=true&
            useJDBCCompliantTimezoneShift=true&
            useLegacyDatetimeCode=false
            &serverTimezone=Europe/Lisbon
      SPRING_DATASOURCE_USERNAME: gestorespacos
      SPRING_DATASOURCE_PASSWORD: gestorespacos
    ports:
      - 8080:8080
    networks:
      - network_backend
    restart: always

networks:
  network_backend:
    driver: "bridge"

```

7 Testes de Carga

Recorremos ao *JMeter* para testar as capacidades do nosso servidor aplicacional. Construímos então um teste aos pedidos à API que podem ser efetuados pelos visitantes, ou seja, as funcionalidades de pesquisa e consulta de eventos, horários e espaços comuns, como se apresentam desde a Figura 22 até à Figura 25.

HTTP Request

Name: Consultar Evento
Comments:

Web Server
Server Name or IP: localhost Port Number: 8080

HTTP Request

Implementation: Protocol [http]: http Method: POST Content encoding:

Path: /public/users/evento

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data for POST Browser-compatible headers

Parameters Body Data

```
1 {"*nome*": "EventoTeste33"}
```

Figura 22: Consultar Evento por nome.

HTTP Request

Name: Consultar Horário
Comments:

Web Server
Server Name or IP: localhost Port Number: 8080

HTTP Request

Implementation: Protocol [http]: http Method: POST Content encoding:

Path: /public/users/horario

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data for POST Browser-compatible headers

Parameters Body Data

```
1 {"*espaco*": "Sala 41"}
```

Figura 23: Consultar Horário por nome.

HTTP Request

Name: Consultar Eventos Entre Data
Comments:

Web Server
Server Name or IP: localhost Port Number: 8080

HTTP Request

Implementation: Protocol [http]: http Method: POST Content encoding:

Path: /public/users/eventos

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data for POST Browser-compatible headers

Parameters Body Data

```
1{
2    "inicio": "2017-01-01",
3    "fim": "2023-01-01"
4 }
```

Figura 24: Consultar eventos entre duas datas.

HTTP Request

Name: Consultar um Espaço Comum em Particular
Comments:

Web Server
Server Name or IP: localhost Port Number: 8080

HTTP Request

Implementation: Protocol [http]: http Method: GET Content encoding:

Path: /public/users/ecs/view/31

Figura 25: Consultar um Espaço Comum em particular.

HTTP Request

Name: Consultar Espacos Comuns
Comments:

Web Server
Server Name or IP: localhost Port Number: 8080

HTTP Request

Implementation: Protocol [http]: http Method: GET Content encoding:

Path: /public/users/ecs/viewAll

Figura 26: Consultar todos os espaços comuns.

Efetuamos o teste para 1, 10, 100, 1000 e, finalmente, 2000 utilizadores, tendo obtido os resultados apresentados desde a Figura 27 até à Figura 31.

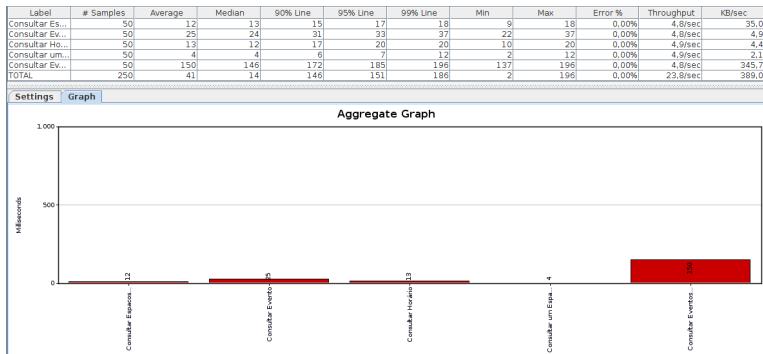


Figura 27: Resultados para $\#\text{sessões paralelas} = 1$.

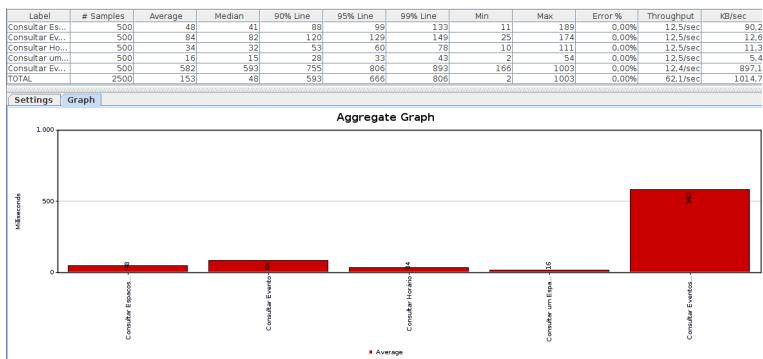


Figura 28: Resultados para $\#\text{sessões paralelas} = 10$.

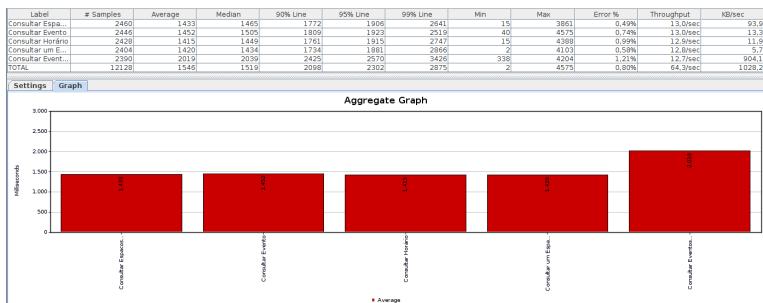


Figura 29: Resultados para $\#\text{sessões paralelas} = 100$.

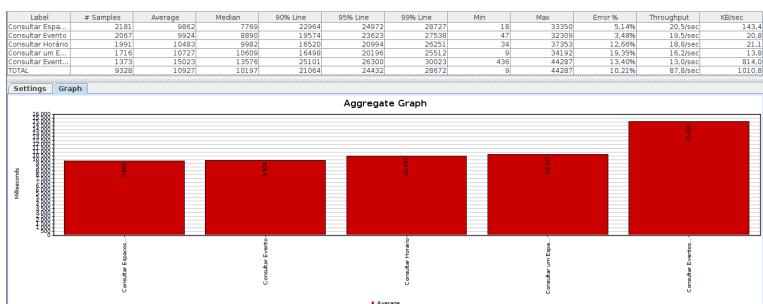


Figura 30: Resultados para $\#\text{sessões paralelas} = 1000$.

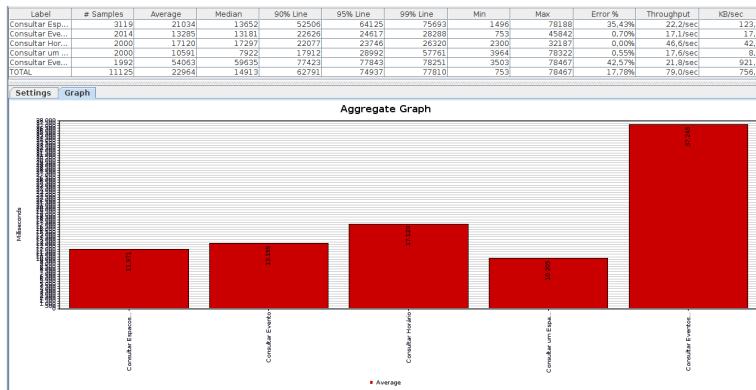


Figura 31: Resultados para #sessões paralelas = 2000.

Como podemos observar, o nosso servidor a partir de 1000 sessões paralelas deixa de conseguir responder em tempo útil a diversos pedidos, começando obter erros cada vez mais superiores a partir desse número.

8 Conclusão

Dado por concluído este trabalho, o grupo considera que realizou um bom trabalho. No entanto, para trabalho futuro, destaca-se a possibilidade de efetuar testes de carga mais completos, acabar de implementar todas as funcionalidades previstas, aumentar a complexidade do sistema adicionando-se novas funcionalidades, bem como explorar mais configurações das *frameworks* utilizadas e efetuar o *deployment* numa infraestrutura mais escalável.

Como forma de melhoria da infraestrutura de modo a conseguir suportar uma carga mais elevada, seria interessante adicionar um *container* intermediário contendo o *nginx*, para realizar a distribuição da carga entre múltiplos servidores aplicacionais que podem ser facilmente replicados uma vez que já se encontram em *containers*.

Em retrospectiva, este trabalho foi bastante interessante pois pudemos pôr em prática os conceitos dados nas aulas como, por exemplo, ao nível dos *design patterns* que eram implementados pelas ferramentas que utilizámos.

Além disso, foi também bastante importante pois pudemos ganhar alguma experiência prática nas *frameworks* utilizadas, através do desenvolvimento de uma aplicação de raiz.

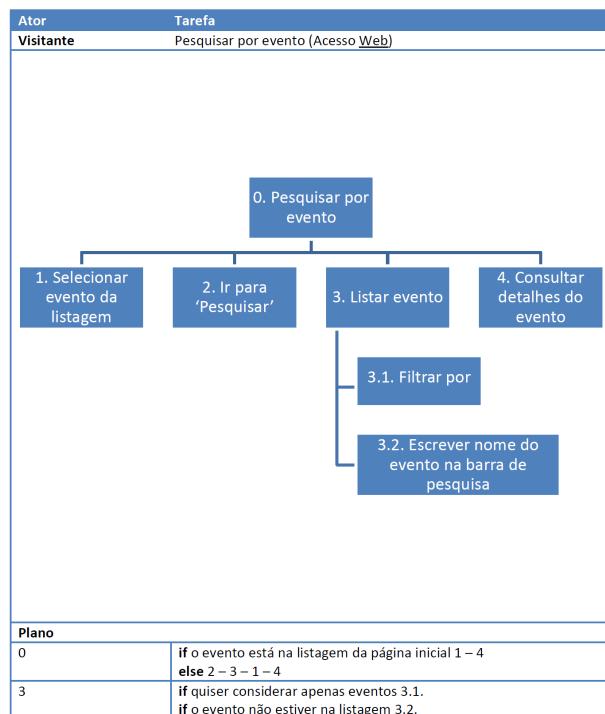
Referências

- [1] Spring, Core Technologies <https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#spring-core> Último acesso: 21/06/2019
- [2] Google Calendar API <https://developers.google.com/calendar/v3/reference/> Último acesso: 29/06/2019
- [3] <https://developers.google.com/identity/protocols/OAuth2>

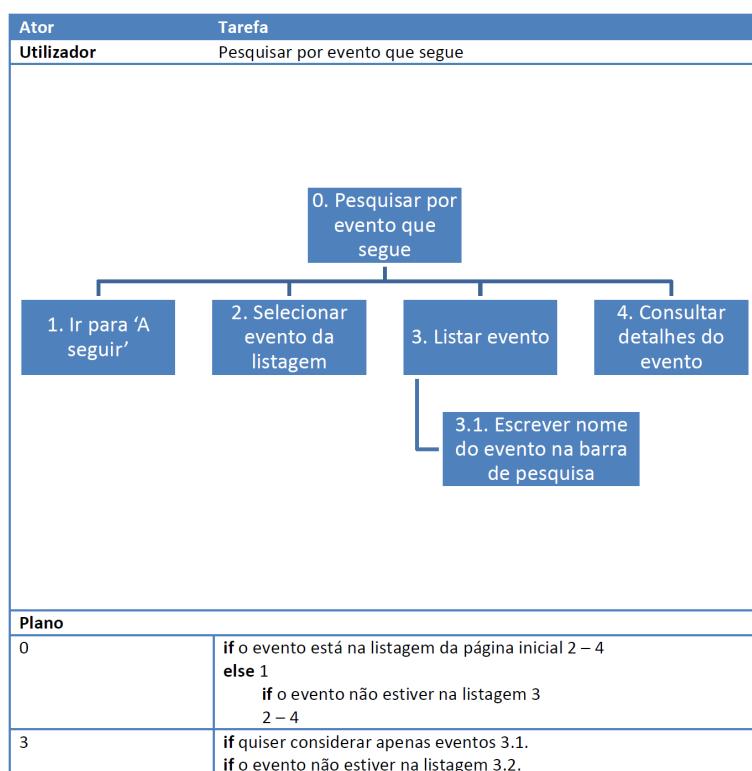
Anexos

A Modelos de Tarefas

A.1 Consultar Evento



A.2 Consultar Evento que Segue



B Especificação dos Use Case

B.1 Consultar Evento

Brief Description	Acesso à informação de um dado evento.	
Preconditions	Evento existe.	
Post-conditions	Apresentação da informação do evento especificado.	
Flow of Events	Actor Input	System Response
	1 Indica o evento que pretende consultar	
	2	Apresenta os detalhes do evento.

B.2 Consultar Horário de um Espaço

Brief Description	Acesso ao horário de um dado espaço.	
Preconditions	Espaço existe.	
Post-conditions	Apresentação do horário do especificado.	
Flow of Events	Actor Input	System Response
	1 Indica o espaço que pretende consultar	
	2	Apresenta o horário associado a esse espaço

B.3 Seguir Evento

Brief Description	Ficar à escuta de notificações associadas a um evento.	
Preconditions	Evento existe e o utilizador não o está a seguir.	
Post-conditions	Evento adicionado à lista de seguidos do utilizador.	
Flow of Events	Actor Input	System Response
	1 Especifica o evento que pretende seguir.	
	2	Associa o evento ao utilizador.
	3	Indica que o utilizador está a seguir o evento.

B.4 Deixar de Seguir Evento

Brief Description	Parar de seguir e receber notificações de um dado evento.	
Preconditions	Evento existe ; Utilizador está a seguir o evento.	
Post-conditions	Utilizador não segue o evento.	
Flow of Events	Actor Input	System Response
	1 Indica que pretende deixar de seguir o evento.	
	2	Remove o evento da lista de seguidos da respetiva conta.
	3	Informa que o unfollow foi realizado com sucesso.

B.5 Consultar Eventos que está a Seguir

Brief Description	Observar eventos que está a seguir.	
Preconditions		
Post-conditions	Apresentação da lista dos eventos seguidos pelo utilizador.	
	Actor Input	System Response
Flow of Events	1 Indica que pretende consultar os eventos que está a seguir.	
	2	Determina os eventos associados ao utilizador.
	3	Exibe os eventos.

B.6 Consultar Notificações

Brief Description	Apresenta a listagem das notificações mais recentes, relativamente aos eventos que segue.	
Preconditions		
Post-conditions	Apresentação da informação das notificações.	
	Actor Input	System Response
Flow of Events	1 Informa que pretende consultar as suas notificações.	
	2	Carrega a informação associada às notificações para este utilizador.
	3	Exibe a lista de notificações.
Exception 1 [não existem notificações associadas à conta] (passo 2)	Actor Input	System Response
	1	Alerta que no momento não existem notificações para apresentar.

B.7 Consultar Eventos pelos quais é Responsável

Brief Description	Observar os eventos pelos quais é responsável.	
Preconditions		
Post-conditions	Apresentação da lista dos eventos dirigidos pelo utilizador.	
	Actor Input	System Response
Flow of Events	1 Indica que pretende consultar os eventos que está a gerir.	
	2	Carrega os eventos que estão associados a este ator.
	3	Apresenta a listagem desses eventos.

B.8 Efetuar pedido de alteração de Evento

Brief Description	Pedir para alterar as informações associadas aos eventos	
Preconditions	O utilizador é o responsável pelo evento em questão	
Post-conditions	Pedido registado.	
Flow of Events	Actor Input	System Response
	1 Indica que pretende alterar os dados do evento.	
	2	Apresenta a informação atual associada a esse evento.
	3 Especifica alterações.	
	4	Verifica se o nome/descrição foram modificados.
	5	Valida informação.
	6	Determina quanto tempo falta para o evento ocorrer, de acordo com os novos dados.
	7	Regista pedido.
	8	Informa que o pedido foi efetuado com sucesso.
Exception 1 [dados inválidos] (passo 5)	Actor Input	System Response
Alternativa 1 [O nome e/ou descrição o evento foram alterados]	1	Indica que as alterações propostas são inválidas.
	2	Indica que a informação foi atualizada com sucesso.
	3	Regressa a 5.
Alternativa 2 [menos de 1 hora para o evento && não existe conflito com outro evento marcado] (passo 6)	Actor Input	System Response
	1	Regista alterações.
	2	Notifica os seguidores do evento.
	3	Informa que as alterações foram efetivadas.

B.9 Consultar os seus pedidos

Brief Description	Observar os seus pedidos pendentes e atendidos.	
Preconditions		
Post-conditions	Apresentação da informação associada aos pedidos que dizem respeito ao ator	
Flow of Events	Actor Input	System Response
	1 Indica que pretende consultar os seus pedidos.	
	2	Carrega os dados dos pedidos associados a este ator.
	3	Apresenta a listagem dos pedidos.

B.10 Aprovar Pedido

Brief Description	Aprovar um pedido efetuado (novo evento/alteração do horário/local de um evento já existente)	
Preconditions	Pedido está pendente.	
Post-conditions	Alterações ao evento são efetivadas.	
Flow of Events	Actor Input	System Response
	1 Indica que pretende aceitar o pedido.	
	2	Cancela qualquer evento que causasse conflito com o evento em questão e notifica o responsável e seguidores.
	3	Armazena informações do evento.
	4	Notifica o responsável pelo evento.
Alternativa 1 [não é um novo evento] (passo 4)	Actor Input	System Response
	1	Notifica o responsável e seguidores do evento.
	2	Regressa a 5.

B.11 Rejeitar Pedido

Brief Description	Declinar pedido efetuado por um utilizador com privilégios de requisição.	
Preconditions	Pedido não foi cancelado.	
Post-conditions	Pedido descartado ; Responsável notificado.	
Flow of Events	Actor Input	System Response
	1 Indica o pedido que pretende refusar.	
	2	Apresenta a opção de fornecer uma justificação para a interrupção do evento.
	3	Descarta o pedido.
	4	Notifica o requerente.
Alternativa 1 [pretende justificar] (passo 2)	Actor Input	System Response
	1 Indica a razão do pedido não poder ser aceite.	
	2	Regista justificação.
	3	Retorna a 3.

B.12 Cancelar Pedido

Brief Description	Cancelar um pedido pendente (reserva de espaço para novo evento / alteração de informação de um evento pelo qual é responsável)	
Preconditions	Pedido ainda não foi atendido.	
Post-conditions	Pedido foi cancelado.	
Flow of Events	Actor Input	System Response
	1 Especifica o pedido pendente que pretende cancelar.	
	2	Suspende o pedido indicado.
	3	Indica que o pedido foi eliminado com sucesso.

B.13 Definir um novo evento

Brief Description	Gestor aloca um espaço para um novo evento.	
Preconditions	Espaço existe no sistema	
Post-conditions	Evento fica registado.	
Flow of Events	Actor Input	System Response
	1 Indica que pretende definir um novo evento.	
	2 Especifica nome e descrição do evento.	
	3	Regista informação.
	4	Pede para indicar se o evento é periódico.
	5 Especifica a periodicidade do evento.	
	6	Processa escolha.
	7	Requer dias e horas em que o evento irá decorrer.
	8 Fornece agendamento desejado.	
	9	Valida informação
	10	Cancela qualquer evento que causasse conflito com o novo e notifica o respetivo responsável e seguidores.
	11	Regista novo evento.
	12	Informa que o espaço foi alocado com sucesso.
Alternativa 1 [evento periódico] (passo 6)	Actor Input	System Response
	1	Requer os dias de referência, o respetivo horário, e a periodicidade associada.
	2 Especifica agendamento.	
	3	Regressa a 9.
Exception 1 [agendamento inválido] (passo 9)	Actor Input	System Response
	1	Informa que os horários especificados são inválidos.

B.14 Modificar Evento

Brief Description	Alterar nome e/ou descrição e/ou momento e/ou local da ocorrência de um evento.	
Preconditions	Evento encontra-se no sistema.	
Post-conditions	Informação do evento atualizada ; Seguidores/Responsável notificados.	
Flow of Events	Actor Input	System Response
	1 Indica que pretenda alterar a informação de um dado evento.	
	2	Apresenta a especificação atual do evento.
	3 Fornece os novos dados do evento.	
	4	Valida informação.
	5	Cancela quaisquer eventos que causem conflito e notifica respetivo responsável e seguidores.
	6	Regista novas informações.
	7	Notifica o responsável/seguidores do evento que foi modificado.
	8	Informa que o evento foi atualizado com sucesso.
Exception 1 [Dados inválidos] (passo 4)	Actor Input	System Response
	1	Alerta que a nova informação fornecida não é válida.

B.15 Cancelar Evento

Brief Description	Suspender um evento que dirige.	
Preconditions	Utilizador é o responsável pelo evento.	
Post-conditions	Evento é removido.	
Flow of Events	Actor Input	System Response
	1 Indica o evento que pretende cancelar.	
	2	Carrega a informação associada ao evento.
	3	Apresenta a opção de fornecer uma justificação para a interrupção do evento.
	4	Suspende o evento.
	5	Notifica os seguidores do evento.
	6	Informa que o evento foi cancelado.
Alternativa 1 [Evento é periódico] (passo 2)	Actor Input	System Response
	1	Pede para especificar (s) intervalo(s) de tempo onde o evento não vai acontecer.
	2	Especifica os momentos pretendidos.
	3	Valida informação.
	4	Regressa a 3.
Exception 1 [Intervalo inválido] (passo 2.3.)	Actor Input	System Response
	1	Alerta que os momentos fornecidos são inválidos.
Alternativa 2 [pretende justificar] (passo 3)	Actor Input	System Response
	1 Descreve a razão do cancelamento do evento.	
	2	Regista justificação.
	3	Retorna ao passo 4.

B.16 Definir Espaço Comum

Brief Description	Agrupar espaços.	
Preconditions	Espaços existem no sistema.	
Post-conditions	Grupo de espaços formado.	
Flow of Events	Actor Input	System Response
	1 Indica que pretende definir um espaço comum.	
	2	Apresenta espaços do sistema.
	3 Especifica os espaços que irão formar o grupo.	
	4	Valida agrupamento.
	5	Regista grupo.
Exception 1 [não foi selecionado pelo menos 1 espaço] (passo 4)	Actor Input	System Response
	1	Informa que não é possível formar um grupo com os espaços selecionados.

B.17 Atualizar constituição de um Espaço Comum

Brief Description	Alterar a informação - designação e/ou constituição - de um espaço comum	
Preconditions	O espaço comum existe.	
Post-conditions	Informação do agrupamento é atualizada.	
Flow of Events	Actor Input	System Response
	1 Especifica que pretende alternar a informação do espaço comum.	
	2	Carrega informação associada a esse espaço comum.
	3	Apresenta especificação do espaço comum.
	4	Fornece opções de alteração.
	5 Indica a nova designação do espaço comum.	
	6	Valida informação.
	7 Especifica a nova constituição de espaços.	
	8	Valida grupos.
	9	Atualiza informação do espaço comum.
	10	Informa que o espaço comum foi atualizado com sucesso.
Alternativa 1 (passo 4) [não pretende alterar designacao]	Actor Input	System Response
1 Informa que não pretende alterar a designação do agrupamento.		
	2	Retorna a 7.
Exception 1 (passo 6) [dados inválidos]	Actor Input	System Response
1		Informa que a informação fornecida não é válida.
	1	
Exception 2 (passo 8) [agrupamento vazio]	Actor Input	System Response
1		Informa que um espaço comum tem de ser constituído por pelo menos um espaço.
	1	

B.18 Apagar Espaço Comum

Brief Description	Cancelar agrupamento de espaços.	
Preconditions	Espaço comum existe.	
Post-conditions	Espaço comum removido do sistema.	
Flow of Events	Actor Input	System Response
	1 Especifica o espaço comum que pretende desagrupar.	
	2	Descarta informação que define o agrupamento dos espaços.
	3	Informa que o espaço comum foi apagado com sucesso.

B.19 Carregar Espaços

Brief Description	Adição de espaços ao sistema.	
Preconditions		
Post-conditions		
Flow of Events	Actor Input	System Response
	1 Indica que pretende definir novos espaços.	
	2	Pede para especificar de onde deve carregar a informação.
	3 Explicita fonte dos dados.	
	4	Carrega espaços válidos e ignora os restantes.
	5	Fornece log que descreve os espaços carregados com sucesso.

B.20 Carregar Horários

Brief Description	Carregar eventos associados a espaços do sistema.	
Preconditions	Espaços estão registados no sistema.	
Post-conditions		
Flow of Events	Actor Input	System Response
	1 Indica que pretende carregar eventos.	
	2	Pede para especificar de onde deve carregar a informação.
	3 Explicita fonte dos dados.	
	4	Armazena eventos válidos e ignora os restantes.
	5	Apresenta log que descreve os eventos carregados com sucesso.

B.21 Registar Atores

Brief Description	Registrar novas contas de utilizador/utilizador com privilégios/gestores de espaços	
Preconditions		
Post-conditions		
Flow of Events	Actor Input	System Response
	1 Indica que pretende registrar novas contas de utilizador.	
	2	Pede para especificar de onde deve carregar a informação.
	3 Explicita fonte dos dados.	
	4	Carrega contas válidas e ignora as restantes.
	5	Fornece log que descreve os atores registados com sucesso.

B.22 Alterar Estatuto

Brief Description	Modificar o estatuto de um dado ator(utilizador, utilizador com privilégios ou gestor de espaços).	
Preconditions	Ator está registado no sistema.	
Post-conditions	Ator tem o novo estatuto atribuído.	
Flow of Events	Actor Input	System Response
	1 Identifica o ator a alterar de estatuto.	
	2	Lista os estatutos possíveis.
	3	Pede para selecionar o novo estatuto.
	4 Especifica o estatuto pretendido.	
	5	Adapta os privilégios associados ao ator de acordo com a opção escolhida.
	6	Informa que o ator teve a sua posição alterada.

B.23 Login

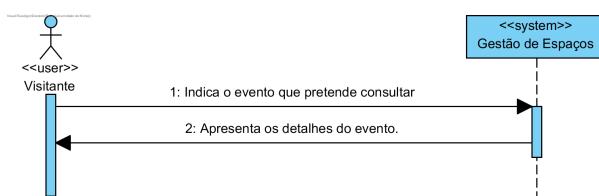
Brief Description	Autenticação do utilizador.	
Preconditions	Utilizador já está registado na aplicação.	
Post-conditions	Utilizador autenticado.	
Flow of Events	Actor Input	System Response
	1 Indica que pretende efetuar o login.	
	2	Pede e-mail e password
	3 Insere os dados	
	4	Valida Informação
	5	Informa que o utilizador está autenticado
Exception 1 (passo 4) [e-mail inválido]	Actor Input	System Response
1	Indica que o e-mail introduzido não consta no sistema	
Exception 2 (passo 4) [password errada]	Actor Input	System Response
1	Informa que a password fornecida não é válida para o e-mail especificado.	

B.24 Logout

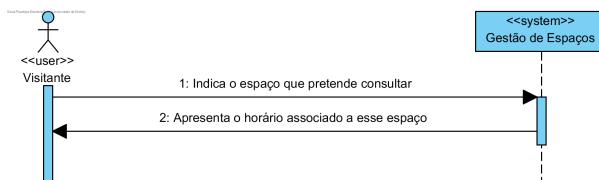
Preconditions	Ator autenticado.	
Post-conditions	Ator com sessão terminada no sistema.	
Flow of Events	Actor Input	System Response
	1 Indica que pretende efetuar logoff.	
	2	Termina a sessão associada a esta conta.
	3	Informa que o utilizador foi deslogado com sucesso.

C Diagramas de Sequência de Sistema

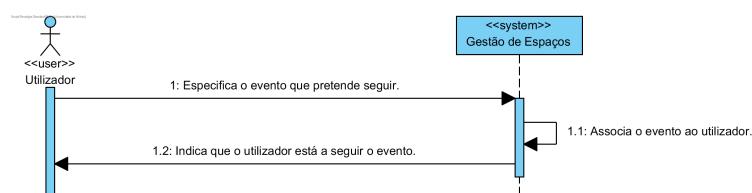
C.1 Consultar Evento



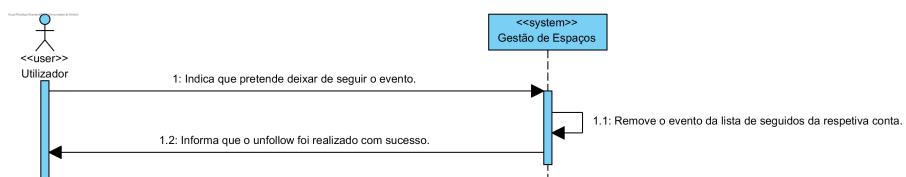
C.2 Consultar Horário de um Espaço



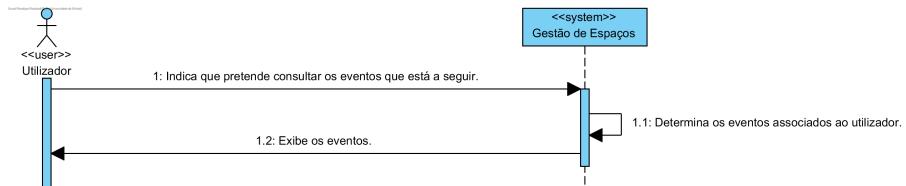
C.3 Seguir Evento



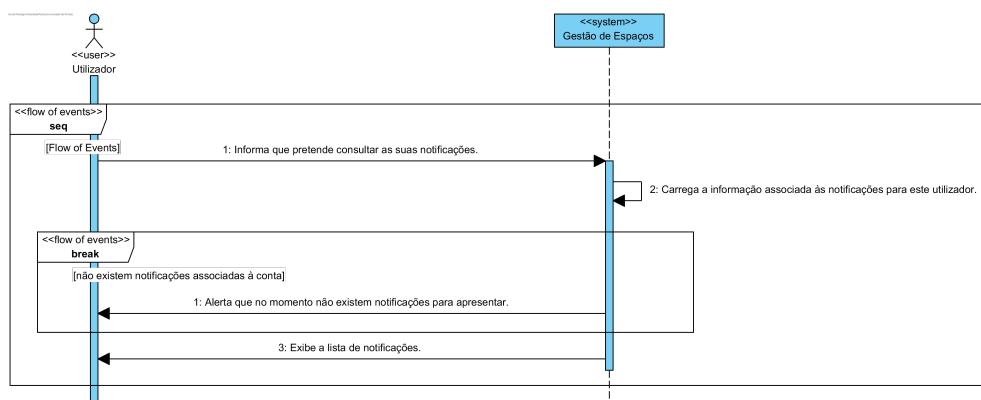
C.4 Deixar de Seguir Evento



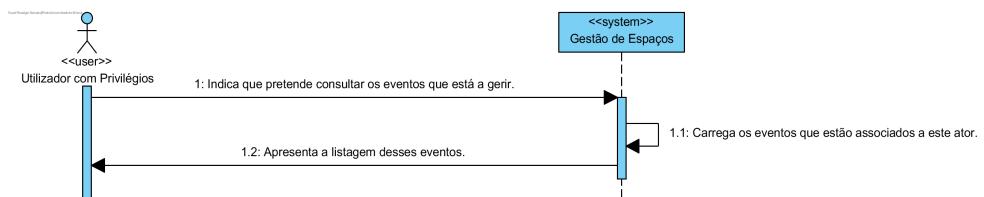
C.5 Consultar Eventos que está a Seguir



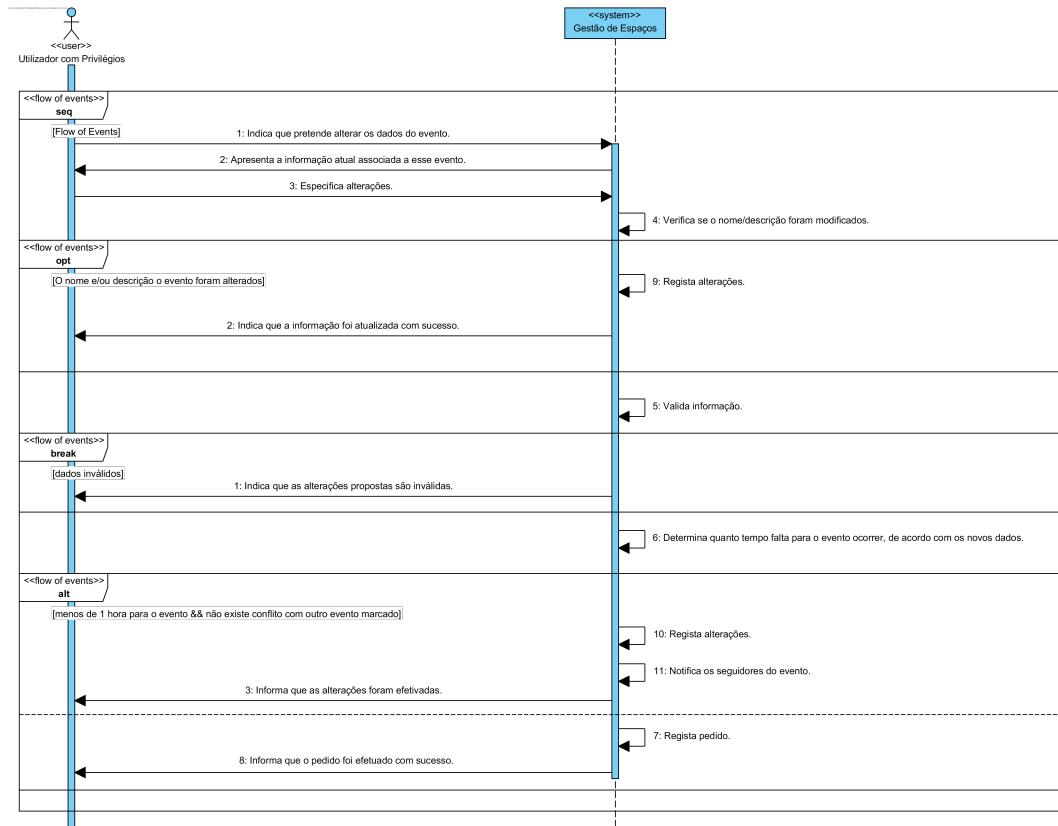
C.6 Consultar Notificações



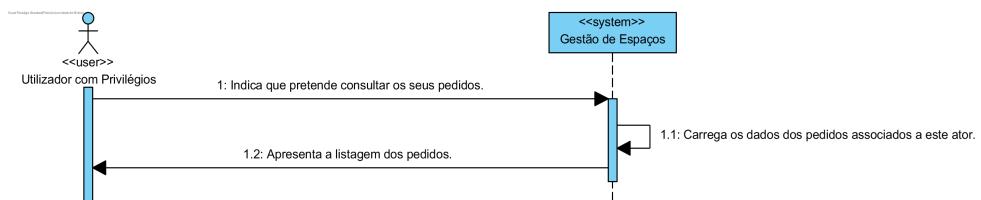
C.7 Consultar Eventos pelos quais é Responsável



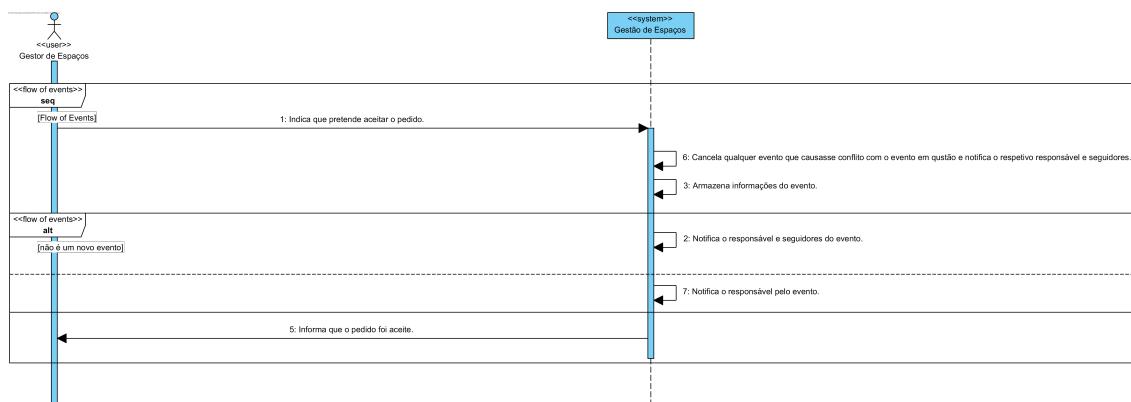
C.8 Efetuar pedido de alteração de Evento



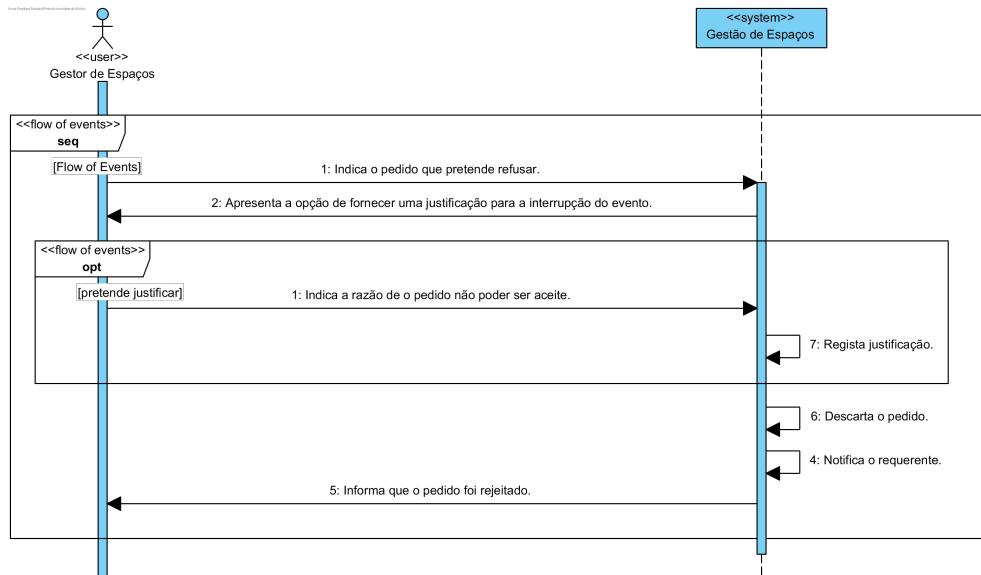
C.9 Consultar os seus pedidos



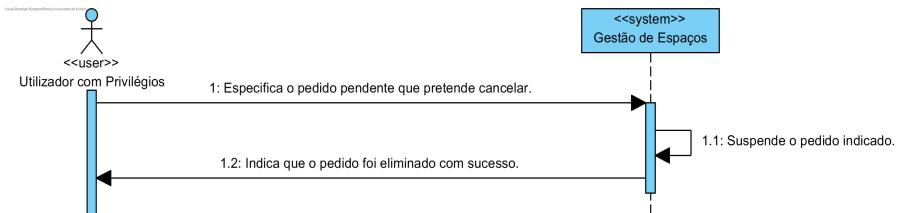
C.10 Aprovar Pedido



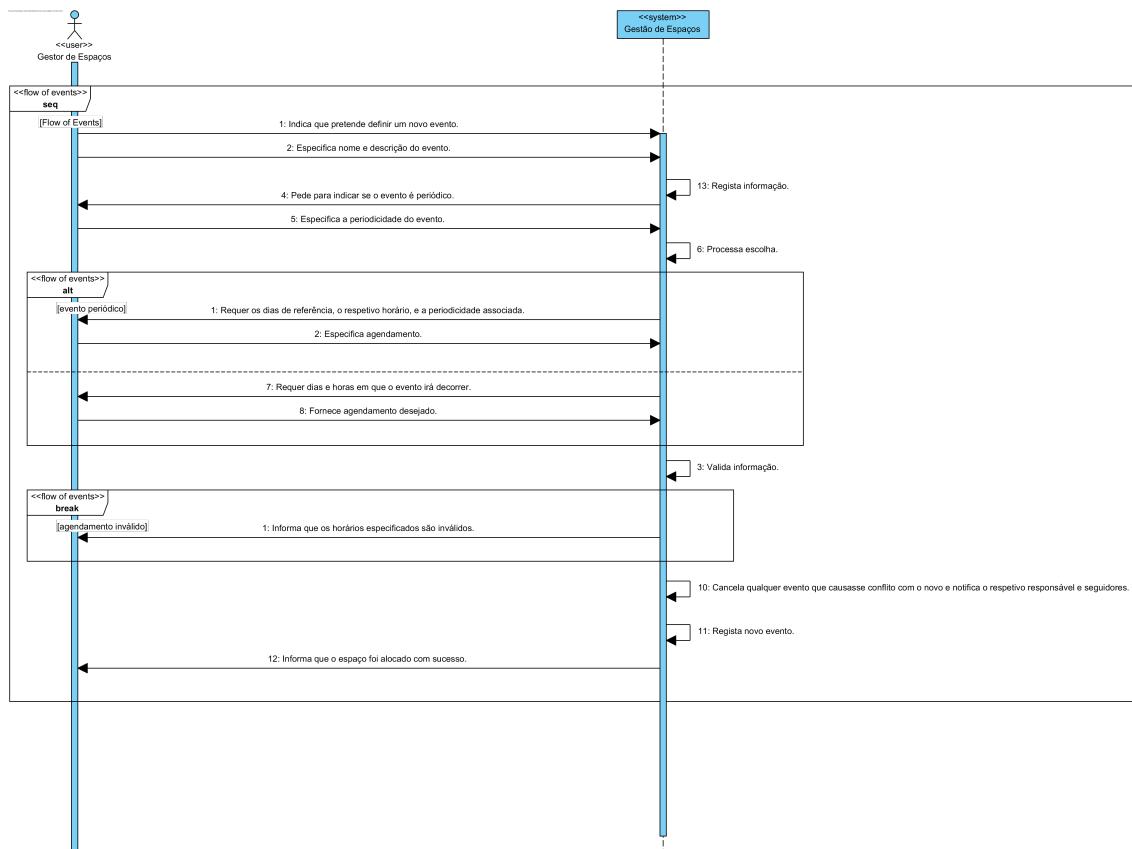
C.11 Rejeitar Pedido



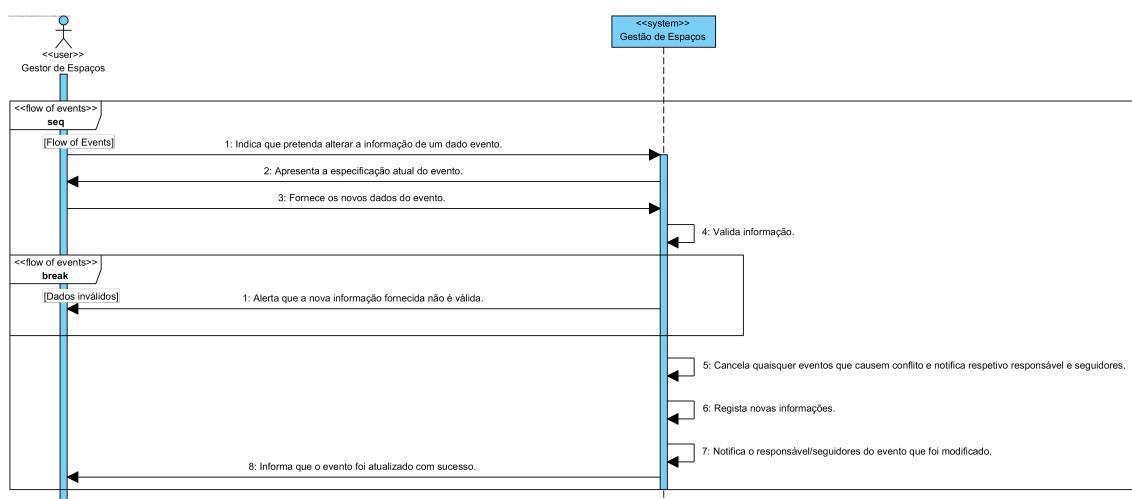
C.12 Cancelar Pedido



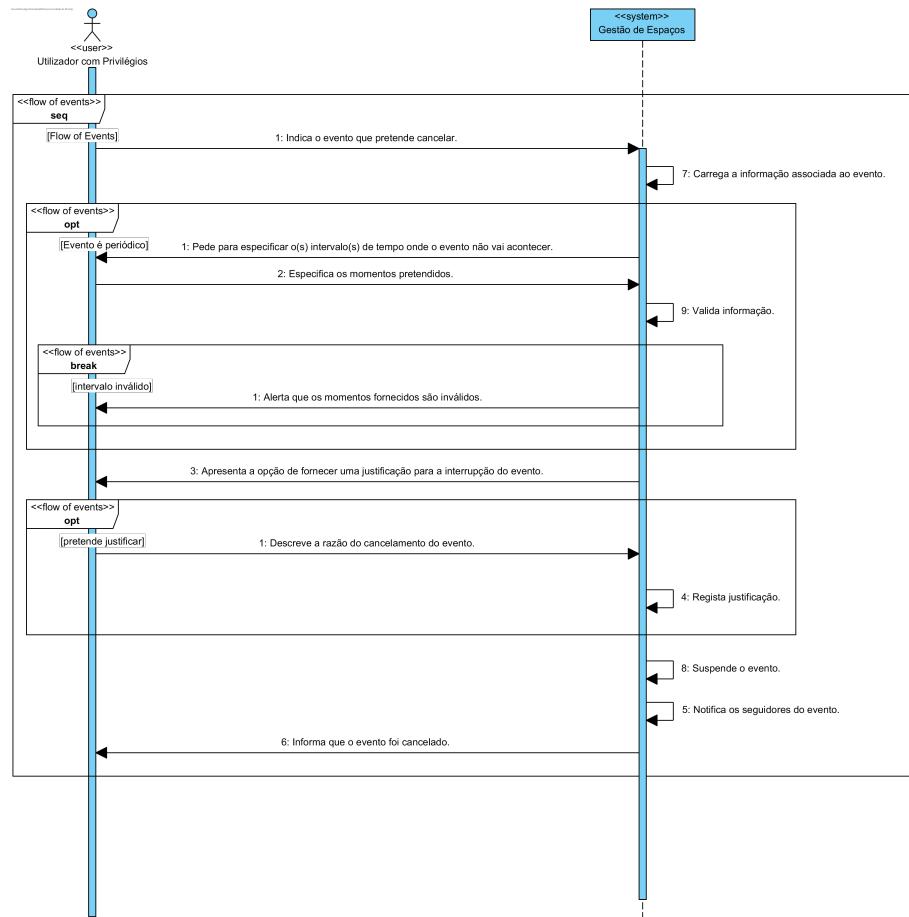
C.13 Definir um novo evento



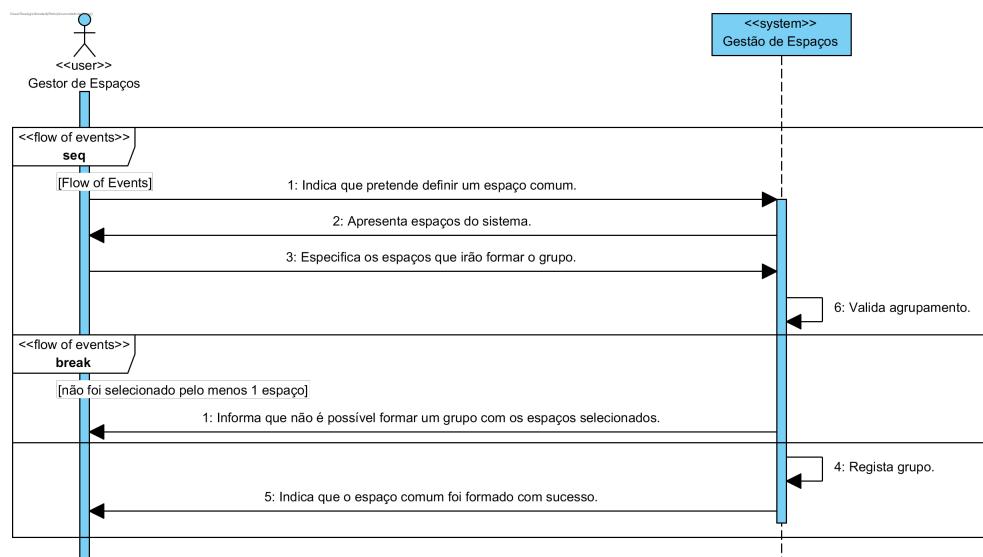
C.14 Modificar Evento



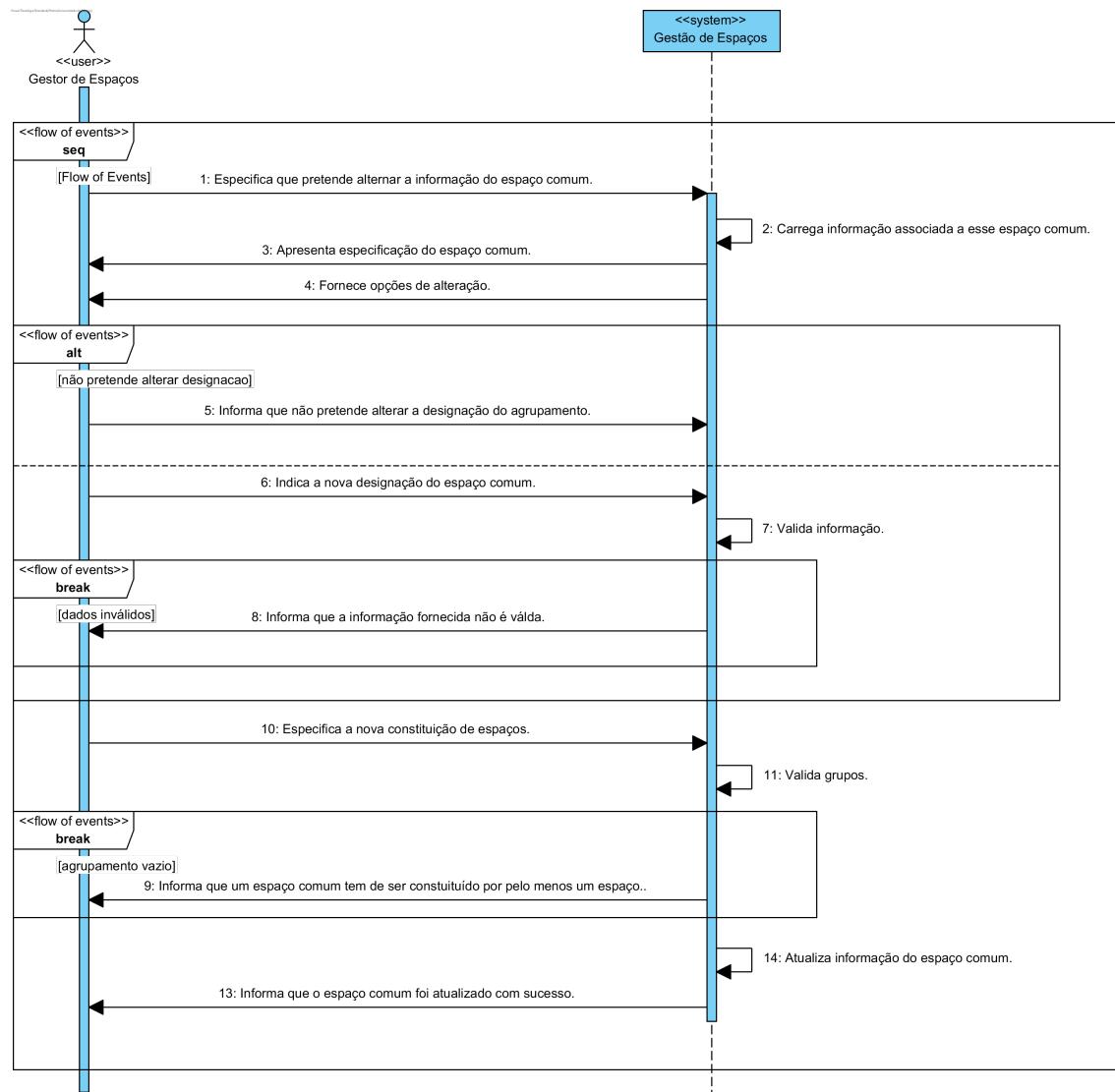
C.15 Cancelar Evento



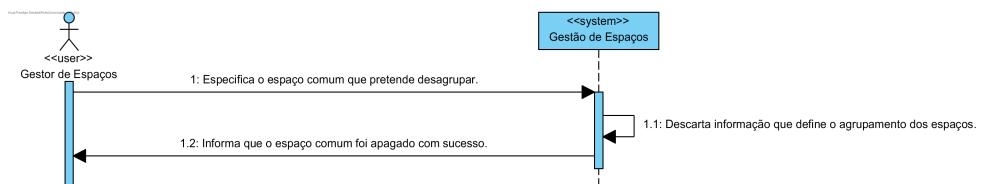
C.16 Definir Espaço Comum



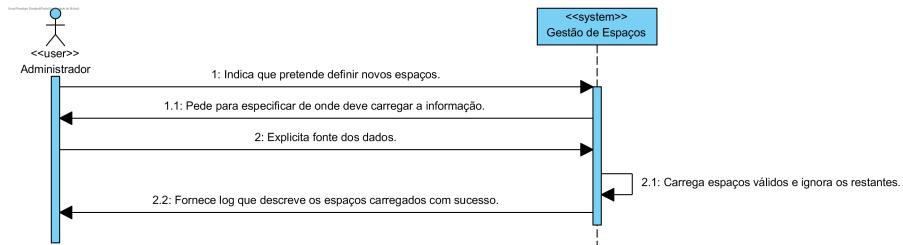
C.17 Atualizar constituição de um Espaço Comum



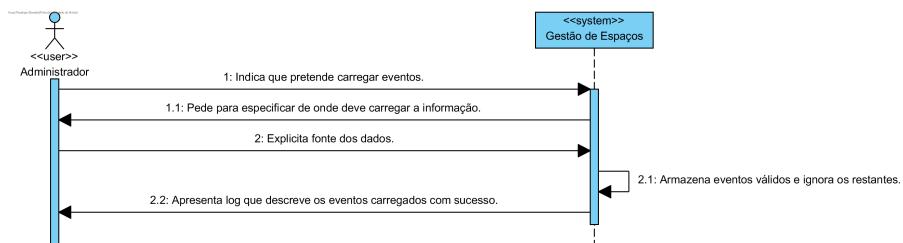
C.18 Apagar Espaço Comum



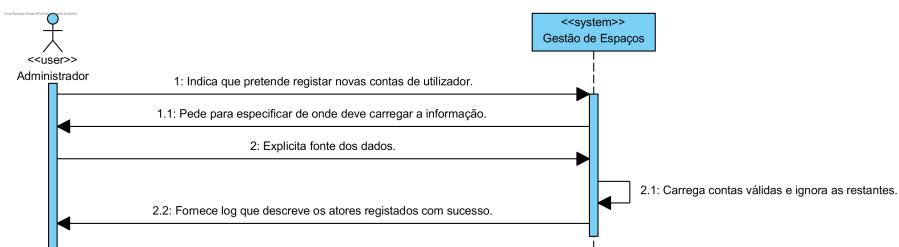
C.19 Carregar Espaços



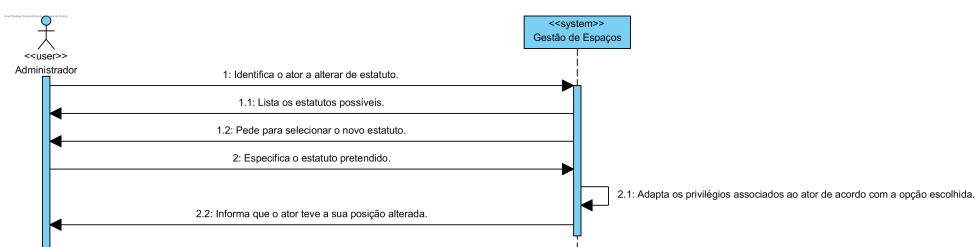
C.20 Carregar Horários



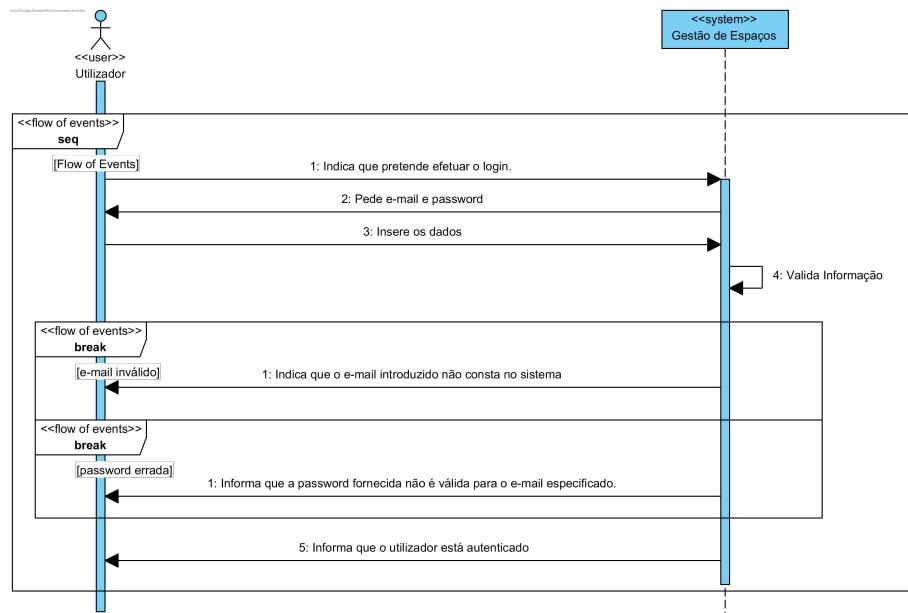
C.21 Registar Atores



C.22 Alterar Estatuto



C.23 Login



C.24 Logout

