

##### Relatório Pet Clinic

Engenharia Informática

## CIAI

# *Keiven Marques nº44677 Tomás Pessanha nº41774*

# *Dezembro de 2019*

# Client Side

O client-side do projeto está bastante incompleto, iremos descrever aqui o objetivo final do cliente.

## Tecnologias

Foram usadas as seguintes tecnologias para o client-side:

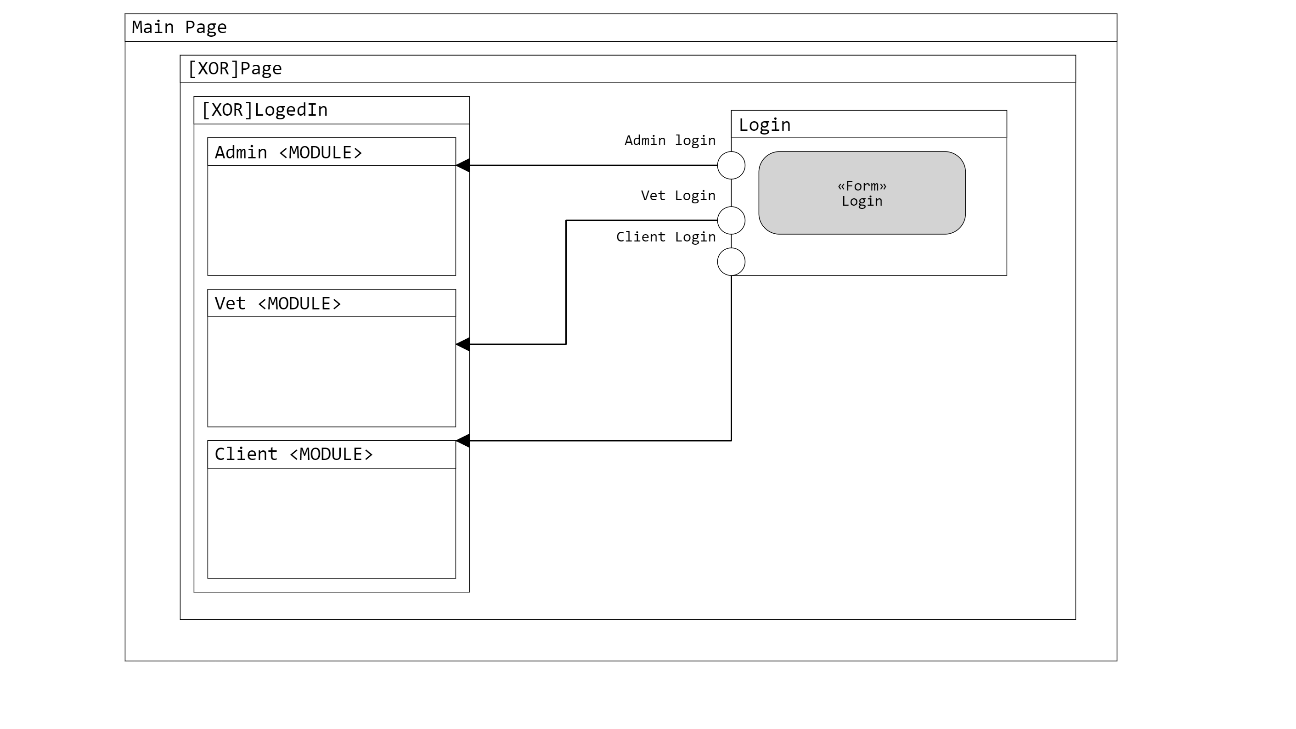
* React
* Typescript
* Redux
* React-Router / Hook routes
* Bootstrap

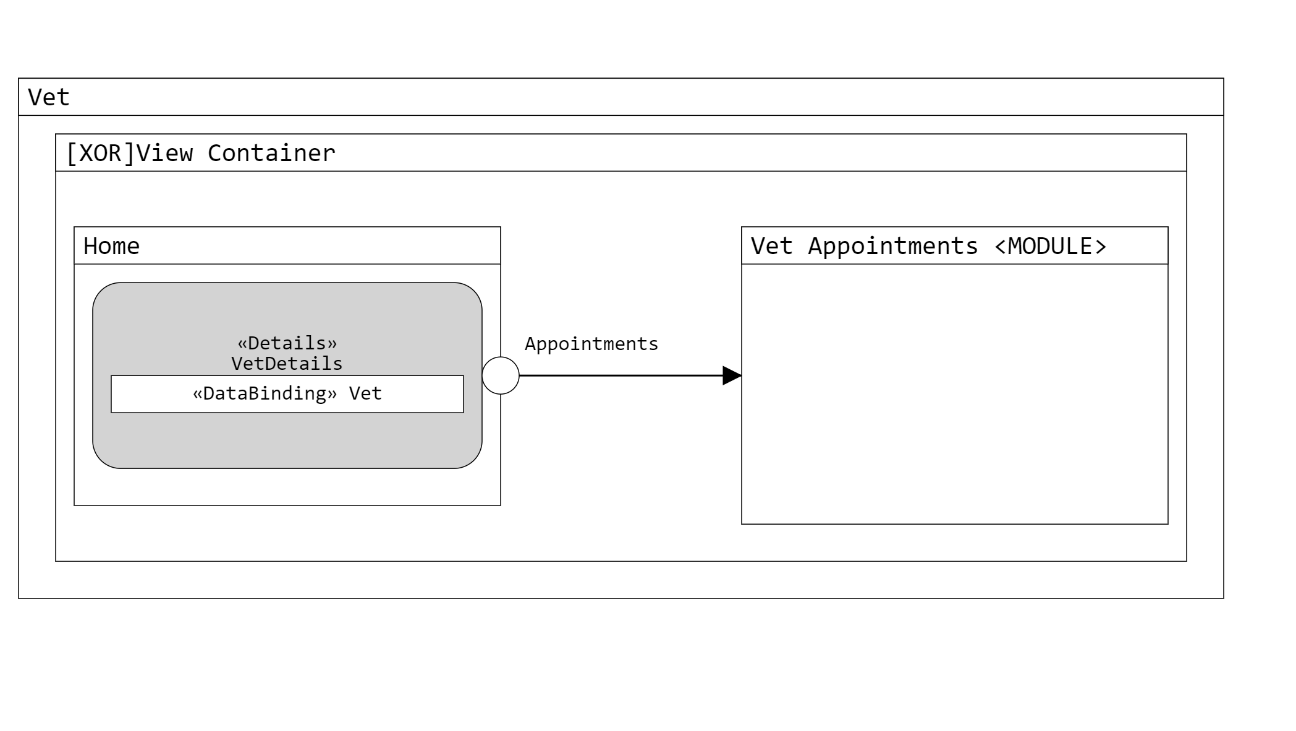
## UI e IFML

O IFML do projeto pode ser encontrado na pasta “docs/ifml”, foi dividido em vários módulos para ficar mais organizado.

Foi utilizada a ferramenta online para editar IFML (“https://editor.ifmledit.org”)

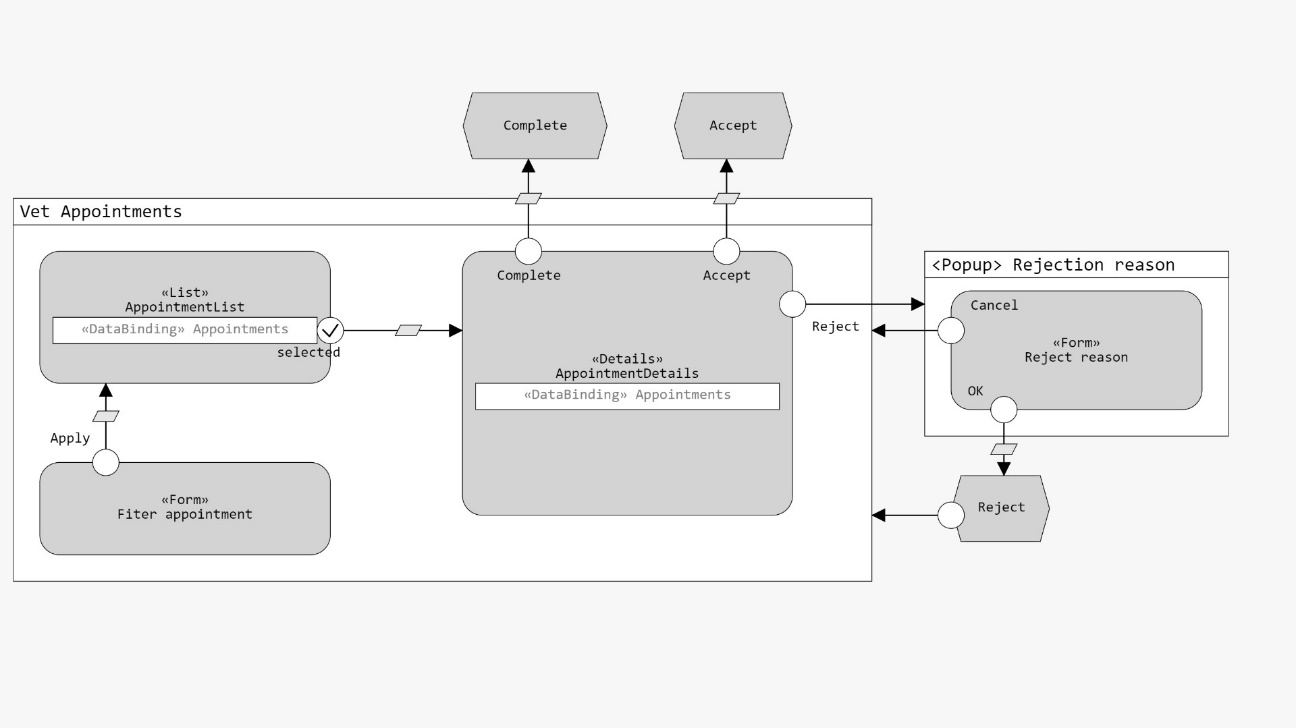
A página principal contem o login para o utilizador ou um dos vários tipos de páginas principais dependendo do role do utilizador.



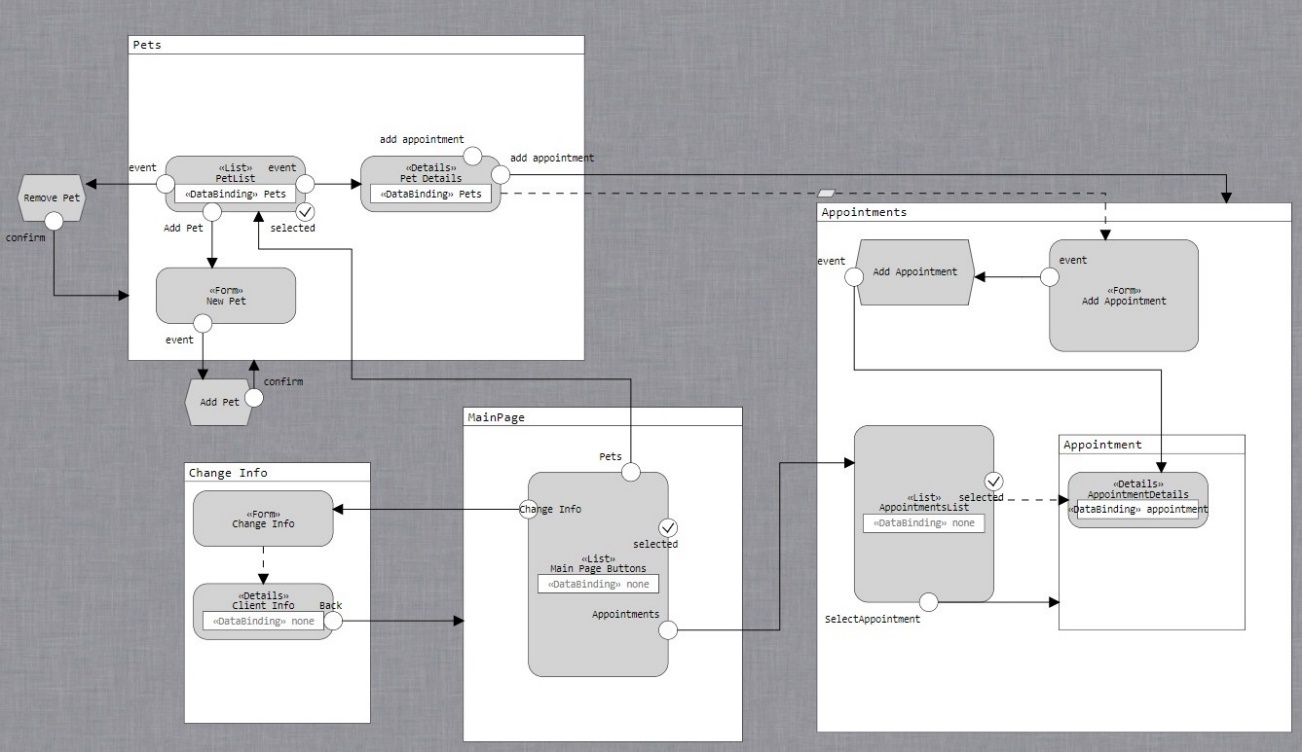
Se o utilizador tiver o role de veterinário então será apresentado o seguinte UI.

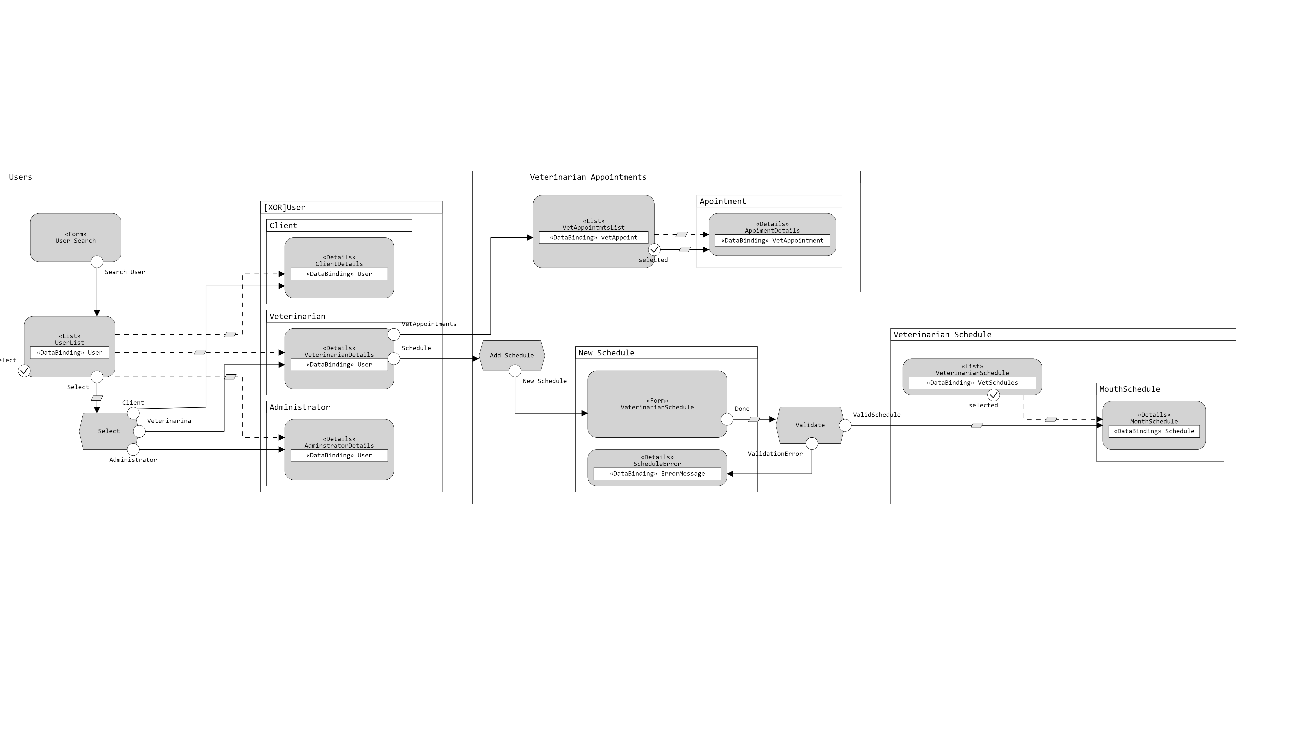
Onde lhe é mostrado detalhes da sua conta, e um botão que o leva a ver as suas marcações.

Na página das marcações, é mostrado a lista de marcações onde ao selecionar uma delas mais informação sobre ela é proporcionada, aqui o utilizador pode então mudar o estado da marcação selecionada.



Se o utilizador tiver o role de cliente então será apresentado o seguinte UI.



Se o utilizador tiver o role de administrador então será apresentado o seguinte UI.

## User Stories

Aqui estão as 10 user stories utlizados.

1. Um cliente abre a sua página, clica no botão ‘Change Info’ para mudar a sua informação de contacto, introduz o seu novo email e numero, guarda as mudanças clicando no botão ‘Confirm’. O resultado pode ser visto ao clicar no seu perfile.
2. Um cliente abre a sua página, clica no botão ‘Appointments’ e na lista de marcação procura a que pretende verificar, clica na mesma para a selecionar.
3. Um cliente abre a sua página, clica no botão ‘Pets’ que o redireciona para uma página com a sua lista de animais, depois clica em ‘Add Pet’. O cliente fornece os detalhes do novo animal e confirma, adicionando o novo animal.
4. Um cliente abre a sua página, clica no botão ‘Pets’ que o redireciona para uma página com a sua lista de animais, seleciona um animal e clica ‘New Appointment’, que abre uma nova janela, o cliente depois preenche o formulário, e confirma ao clicar no botão ‘Confirm’.
5. Um cliente abre a sua página, clica no botão ‘Pets’ que o redireciona para uma página com a sua lista de animais, seleciona um dos seus animais e clica ‘Remove Pet’que remove o animal.
6. Um administrador abre a sua página e clica no botão ‘List of Employees’, uma lista dos funcionários é apresentada, depois o administrador verifica a lista, seleciona um veterinário e seleciona a opção de marcações, onde lhe é mostrado as marcações do veterinário.
7. Um administrador abre a sua página, clica no botão ‘List of Employees’, uma lista dos funcionários é apresentada. Depois o administrador seleciona um veterinário e clica no botão ‘Add Schedule’, o administrador preenche um formulário e clica ‘Done’ para confirmar e marcar o Schedule do veterinário.
8. Um administrador abre a sua página e clica no botão ‘List of Clients’, uma lista de clientes é apresentada, depois seleciona um cliente. A informação do mesmo é apresentada. Depois clica no botão ‘List Pets’ para ver a lista de animais do cliente.
9. Um veterinário abre a sua página, clica no botão ‘Appointments’ que mostra uma lista de marcações do veterinário, apos selecionar uma marcação a informação da marcação aparece ao lado.
10. Um veterinário abre a sua página, clica no botão ‘Appointments’ que mostra uma lista de marcações do veterinário, apos selecionar uma marcação, clica no botão ‘Complete’ para mudar o estado da marcação para ‘Completed’.

# Server Side

O servidor foi realizado com a framework Spring, e Hibernate como JPA provider

O server-side está muito mais complete completando os requisitos base do enunciado completos.

## Imagens

O servidor guarda as imagens de perfile dos utilizadores e animais no seu *file system*, as imagens destes dois grupos encontram-se em pastas separadas e são guardadas com o nome ‘{id}.jpg’ onde o id é o identificador do animal ou do utilizador.

Na base de dados as imagens são guardadas como um URI da mesma.

## Base De Dados

Foram tomadas algumas decisões sobre como a base de dados iria ficar nesta secção iremos explicar a razão destas decisões.

Foi utilizada a estratégia de herança do tipo ‘JOINED’, isto pois enquanto é mais lenta, minimiza a redundância da base de dados e não achamos necessário otimizar para velocidade num projeto deste tipo onde provavelmente não haverá milhões de utilizadores a utilizar um site de clínica veterinária a cada momento.

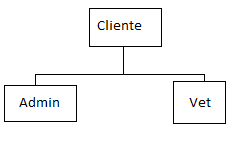
Isto levou a que tenhamos bastastes tabelas como por exemplo para um veterinário são utilizadas 3 tabelas (users, employees, veterinarians).

O local onde se guarda o URI das imagens de um utilizador à primeira vista deveria estar na tabela de ‘users’ no entanto não está, foi colocado em vez disso na tabela ‘clients’ e na tabela ‘employees’, isto para que a base de dados consiga estar coerente com o enunciado e o requisito de que a foto é obrigatória para trabalhadores mas não para clientes.

Os roles são guardados como strings em vez de números inteiros, pela facilidade de se entender rapidamente qual o role que é, isto pelo tradeoff de velocidade.

## Segurança

Foi utilizada uma estrutura de roles na camada de segurança da aplicação



Onde todos os utilizadores registados são clientes e depois pode opcionalmente ser um administrador ou um veterinário.

Isto permite que um administrador não tenha de criar outra conta para utilizar os serviços da clínica (fica aberto para discussão se é uma opção que se pretende ter).

# Conclusao

A primeira parte do trabalho foi realizada como pretendido embora achamos que 1 semana para seguranca seja pouco visto que tivemos que refazer os testes pois deixaram todos de funcionar, a segunda infelizmente não e muita coisa ficou em falta.

Nenhum membro do grupo estava a vontade com as tecnologias de React/Redux/Typescript, que tornou este trabalho bastante dificil que se viu com os resultados. Estas tecnologias não são obvias e demoram a entender o seu funcionamenteo, acreditamos que agora se podessemos comerçar de novo tinhamos o conhecimento necessario para realizar um bom trabalho, mas demoramos muito tempo a entender como React funciona ao ponte de ser preciso 3 dias para fazer um simples componente funcionar como pretendido.

O abandono de um dos membros do grupo no dia da entrega, não ajudou e ficamos com a parte dele por fazer e sem tempo para tal.

Para concluir o servidor está bom, o cliente está muito pobre.