Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores 2022/2023 Unidade Curricular de Laboratórios de Software Docente Felipe Freitas

Área Departamental de Engenharia de Eletrónica e Telecomunicações e de Computadores

Fase 4

Autores: 48269 José Borges
49487 Ricardo Rovisco
49508 João Mota

Relatório para a Unidade Curricular de Laboratórios de Software
Licenciatura em Engenharia Informática e de Computadores

10 – 06 – 2023

Índice

Introdução

O objetivo deste projeto é implementar um sistema de informação para gerenciamento de tarefas inspirado na aplicação *Trello*. O domínio da aplicação é baseado em 4 entidades diferentes:

1. User: Um user é caracterizado por um número único, um nome e um email único.

2. Board: Uma board representa um projeto ou atividade semelhante e é composto por um conjunto de tarefas. Cada board tem um nome único e uma descrição e está associado a um ou vários users. O nome vem da analogia com uma board física, contendo cards fixas que descrevem tarefas. Cada board é dividida numa sequência de lists, que correspondem a colunas numa board física.

3. List: Cada list representa uma fase na execução de uma tarefa (por exemplo, "para fazer", "fazendo" e "feito"). A posição relativa de uma list numa board pode mudar, no entanto, uma list pertence sempre à mesma board. Uma list tem um nome, que deve ser único na board. Cada lista contém uma sequência de cards.

4. Card: Uma card representa uma tarefa e está sempre associada à mesma board. Uma card deve estar sempre dentro de uma list, exceto se estiver arquivada. Uma card pode ser movida entre lists da mesma board. Cada card é caracterizada pelo menos por: um nome, uma descrição, a data de criação e a data de conclusão da tarefa.

Modelo Conceptual

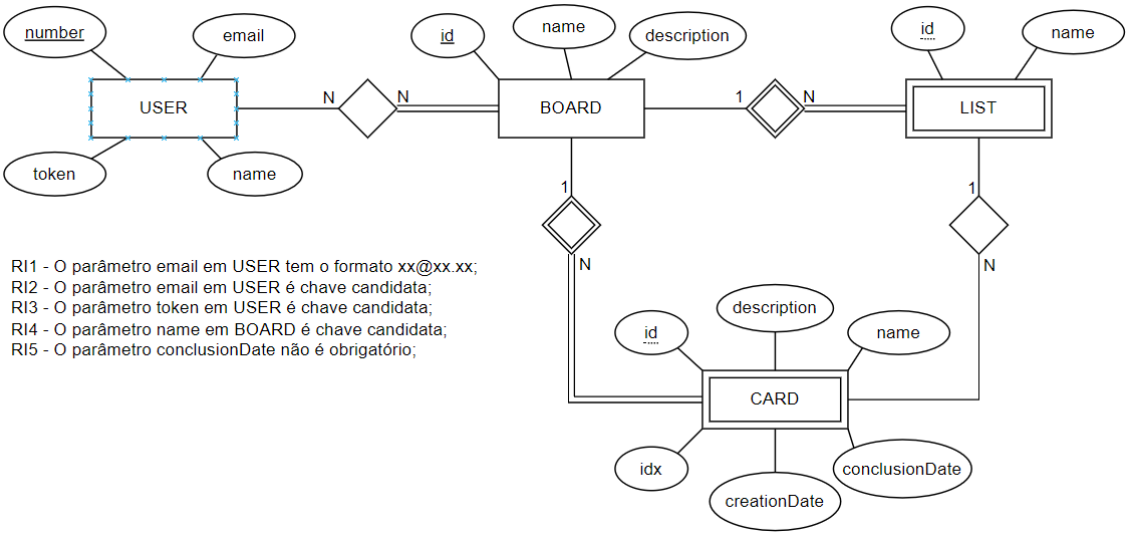

Figura 1 modelo EA

Como referido anteriormente o nosso modelo é composto por 4 entidades das quais 2 delas são fracas.
Como demonstrado na figura 1, a entidade List é fraca da entidade Board tal como a entidade Card d visto que quando não existe uma Board não é possível que exista uma Card. É criada uma 5ª entidade a partir da associação N para N entre User e Board que vai armazenar as chaves primárias da entidade User e da entidade Board, representando os Users que pertencem a uma determinada Board.

Modelo Físico

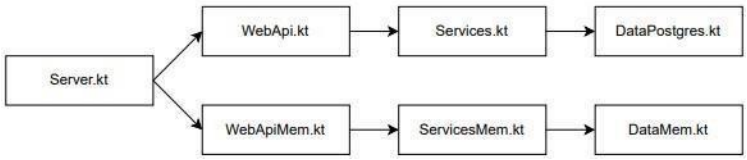

Figura 2 modelo físico

Para a implementação do modelo físico recorremos a REST API que consiste na criação de serviços web com o objetivo de facilitar a comunicação e a integração entre sistemas distribuídos e heterogêneos. Para os módulos WebApi, Services e Data foram feitas duas implementações, uma em memória (WebApiMem.kt, ServicesMem.kt e DataMem.kt) e uma que acede a uma base de dados local (WebApi.kt, Services.kt e DataPostgres.kt).

- Server.kt – Constitui o ponto de entrada para a aplicação.
- WebApi.kt e WebApiMem.kt – Implementação das rotas HTTP que compõem a REST API da aplicação.

- Services.kt e ServicesMem.kt – Implementação da lógica de cada funcionalidade da aplicação.

- DataPostgres.kt e DataMem.kt – Acesso à informação da aplicação.

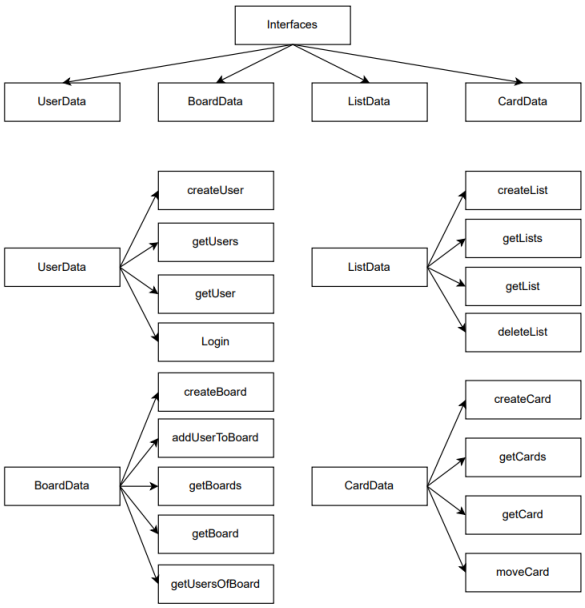

Figura 3 Interfaces

Para garantir que as funcionalidades da aplicação são executadas corretamente foram implementadas 4 interfaces: UserData, BoardData, ListData e CardData.

Especificação Open-API

Toda a informação sobre a aplicação pode ser encontrada aqui.

Navegação pela Aplicação

Foi implementada uma SPA (Single Page Application) que carrega o conteúdo à medida que o utilizador interage com a aplicação fazendo com que apenas sejam atualizadas as partes da página que assim o necessitam. A estrutura de navegação da aplicação pode ser observada na figura 4. Utilizando a plataforma Render, lançámos a aplicação na web.

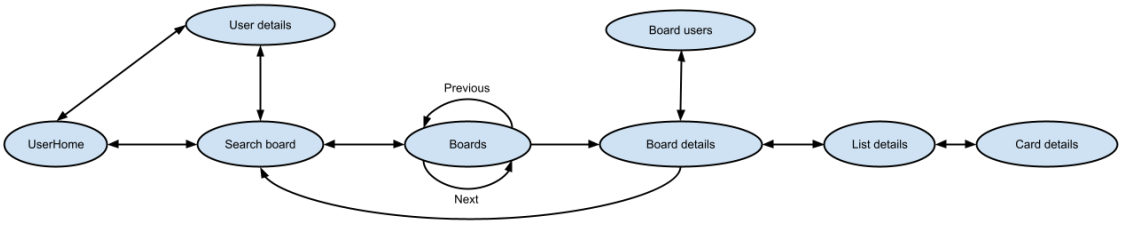

Figura 4 Estrutura de navegação da SPA

Detalhes do Pedido

Quando um utilizador efetua um pedido HTTP na aplicação, o Server envia a informação com a rota e método utilizados para que na WebApi possa executar a funcionalidade correta com a informação passada no path, query ou no body enviando essa informação para os Services que efetuam as verificações necessárias para garantir que a informação seja enviada para a database local ou para memória efetuando a funcionalidade da aplicação no módulo DataPostgres ou DataMem respetivamente. Certas funcionalidades apenas podem ser efetuadas se o User estiver com sessão iniciada na aplicação, para isso desenvolvemos funcionalidades de login e signin para posteriormente efetuarmos essa verificação pelo requestAuthHandler no módulo WebApi.

Gerenciamento da Conexão

Utilizando a função setUrl() definimos o caminho para a base de dados, seguidamente com a função connection.use() estabelecemos a conexão com a base de dados e enviamos as queries necessárias para realizarmos a respetiva funcionalidade que desejamos. Essa conexão é descartada assim que é finalizada, uma vez que o método de conexão que utilizamos a termina automaticamente.

Acesso a Dados

Para aceder à informação da base de dados foi criada uma classe DataPostgres que possui 4 inner classes que acedem às interfaces anteriormente mencionadas na figura 3. Cada inner class dá override às funções da respetiva interface estabelecendo uma ligação com a base de dados e enviando queries anteriormente criadas para que possam ser efetuadas as funcionalidades da aplicação.

Processamento/Tratamento de Erros

Caso aconteça algum erro, este é tratado pelo requestHandler ou pelo requestWithAuthHandler que, apenas escrevem na consola a exceção originada pelo erro e o respetivo código.

Avaliação Crítica

Na realização desta fase melhorámos algum código feito nas fases anteriores, evitando assim a repetição de código, corrigindo também alguns erros de fases anteriores.