Universidade Federal do Rio de Janeiro

Universidade Federal
do Rio de Janeiro
Escola Politécnica

Plataforma SIGADica

Alunos

Patrick Franco Braz
Yuri Mendes Moreira
Guilherme Souza Sobrinho

Professor

Claudio Miceli de Farias

Rio de Janeiro, 01 de Setembro de 2021

Conteúdo

Lista de Tabelas

Lista de Figuras

1 Introdução

O presente relatório visa explicar de forma detalhada o procedimento de idealização e tomadas de decisões sob o contexto da criação da plataforma proposta.

A plataforma SIGADica é num aplicativo web livre cujo objetivo é auxiliar os alunos da UFRJ no planejamento de suas grades de horário. Para tanto, a ferramenta irá expor funcionalidades de busca de diciplinas capazes de atender os anseios dos usuários.

Além de servir como uma ferramenta de busca de informações de diciplinas, a plataforma também servirá como centro de compartilhamento de opiniões de forma anonimizada. As opiniões serão notas atribuidas à determinadas caracteristicas das disciplinas as quais serão mapeadas pelo sistema. Essas notas não apenas estarão presentes nos resultados de busca dos usuários, como também servirão de parâmetros de filtro.

Uma outra funcionalidade que o sistema poderá expor, além das apresentadas acima, é uma engrenagem de recomendação de disciplinas baseado num formulário de perguntas. Dado que as informações das disciplinas disponibilizadas pelo SIGA não possuem metadados de horário, não é possivel identificar sobreposição de horários. Por isso, esta funcionalidade apenas recomendará disciplinas.

Nas seções a seguir serão discutidas a arquitetura proposta para a plataforma, assim como as decisões tomadas pela equipe em relação à comunicação, separação de tarefas e escolhas de tecnologias.

2 Arquitetura proposta

A arquitetura proposta foi pensada sob a perspectiva de micro-serviços. Cada aplicação é independente da outra. Cada um com sua regra de negócio associada. Entretanto, para o perfeito funcionamento da plataforma, cada componente deve funcionar e se comunicar perfeitamente.

O desacoplamento dos componentes é essencial num cenário de produção. Pensando num futuro onde o número de usuários da plataforma cresça, será necessario o escalamento do backend e possivelmente do frontend. Entretanto, não necessariamente o banco de dados ou o webscrapper precisem escalar. Isso permite atender as demandas sem gasto desnecessário de recursos.

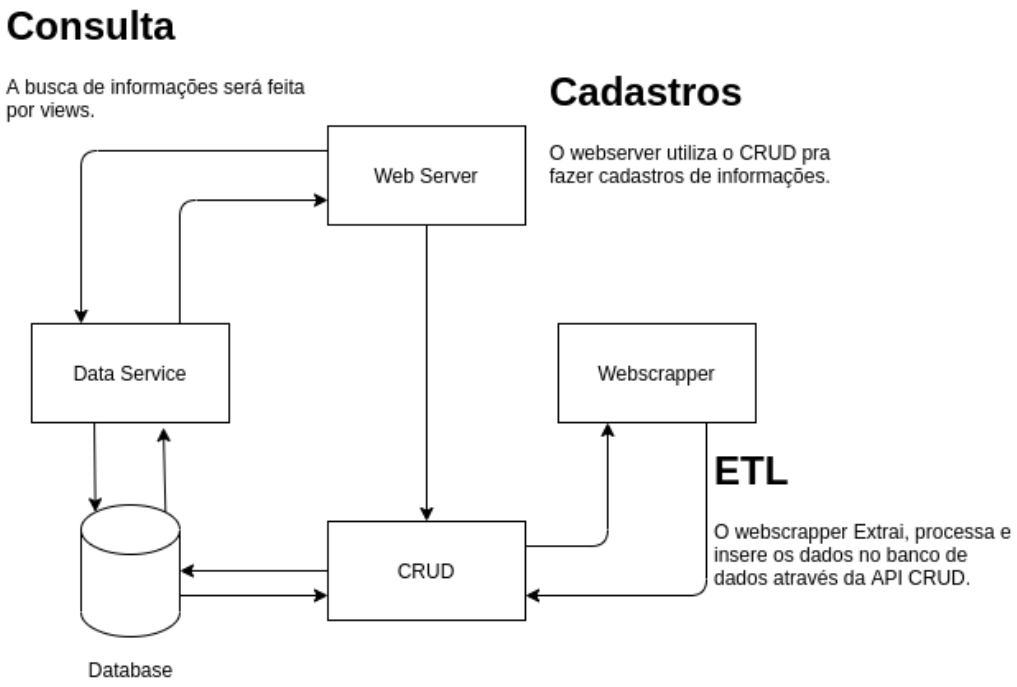

Figura 1: Arquitetura proposta para o trabalho

3 Separação de tarefas

A separação das atividades foi baseada no conhecimento prévio de cada integrante. Como foi pensando numa arquitetura de micro-serviços, os integrantes podem trabalhar de forma independente, entretanto visando a futura comunicação que irá ocorrer entre os componentes. Neste sentido, um integrante ficou responsável pelo servidor web, outro ficou responsável pelo desenvolvimento do backend e por fim, o último integrante ficou responsável por desenvolver as ferramentas necessárias para alimentar o banco de dados que servirá o webserver.

4 Comunicação entre a equipe

Para manter um bom nível de alinhamento entre os integrantes, decidiu-se utilizar um grupo no telegram para comunicações mais instantâneas e de urgência. Além disso, quando é possível, o grupo se reúne nas quarta-feiras pós aula para realizar alguns alinhamentos e tomadas de decisão. Por fim, para tornar melhor o acompanhamento do desenvolvimento, todo o fluxo de commits é feito em "draft pull request"para que o time possa ver o que o

companheiro esta desenvolvendo e opinar sobre qualquer decisão de código.

5 Ferramentas utilizadas

A ferramenta CRUD da plataforma foi implementada utilizando a linguagem de programação Python. Esta linguagem foi escolhida devido à experiência que o integrante já possuía com a linguagem. Nesse âmbito, foi decidido utilizar duas ferramentas para a implementação do CRUD: a biblioteca SQLAlchemy com o objetivo de implementar a modelagem virtual do banco de dados e aproveitar os benefícios da utilização de ORM (Object–relational mapping) e o framework FastAPI como ferramental para a API REST. Por fim, criou-se um Dockerfile com as definições necessárias para que no docker-compose a API e o banco de dados trabalhassem em conjunto.

O webscrapper, assim como a API CRUD, foi implementado utilizando a linguagem de programação Python, escolhida para esta tarefa devido à grande presença em fóruns e quantidade expressiva de tutoriais on-line. Os dados são extraidos da lista de cursos disponibilizados pelo SIGA via requisições http atraves do uso da biblioteca requests. As informações são processadas com a biblioteca lxml e enviadas ao banco utilizando a ferramenta CRUD implementada.

Por fim, a interface de cadastro foi implementada utilizando o framework flutter. Este framework foi escolhido devido à experiências previas na criação de aplicações mobile. Para o cadastro foi utilizado a ferramenta do google Firebase, que possui excelente integração com o framework do flutter, tornando o desenvolvimento mais prático e eficiente.

6 Modelagem do banco de dados

O banco de dados é uma instância MySQL cuja modelagem das entidades será mostrada na figura a seguir. Para a execução local dos testes, foi utilizada uma imagem oficial com o objetivo de hospedar localmente o banco de dados.

Uma vez que o banco de dados esta ativo, um script em python lê um arquivo SQL e executa todos os DDLs necessários para configurar o banco de dados.

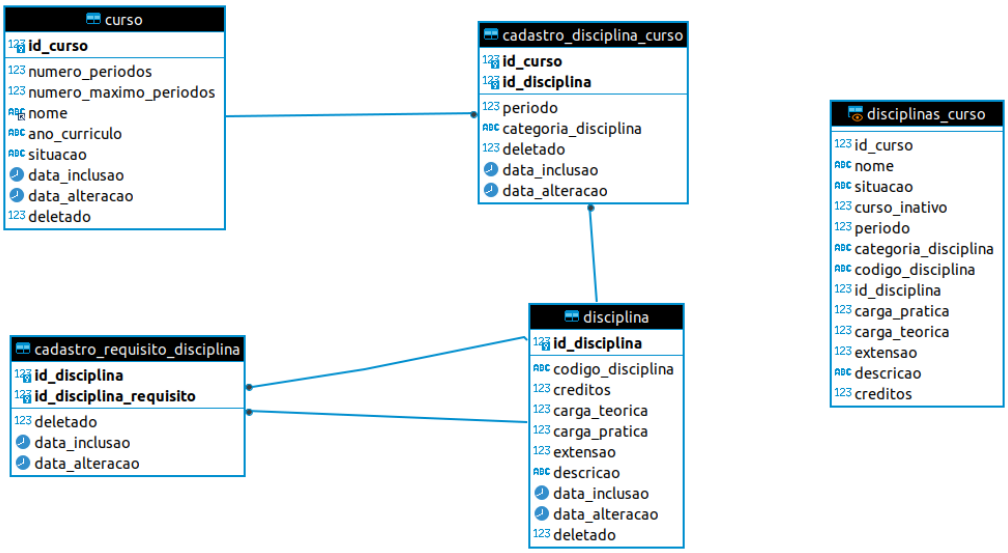

Figura 2: Modelagem do banco de dados

Entretanto, vale ressaltar que no momento em que este relatório esta sendo escrito, a modelagem utilizada é a apresentada pela figura acima. Entretanto, à medida que o desenvolvimento da plataforma avance, alteraços na modelagem inicial podem surgir.

7 Conclusão

Neste trabalho o grupo buscou criar aplicações nas quais se torna possível botar em prática alguns dos conceitos mostrados durante as aulas. Principalmente o conceito de micro-serviços. O desacolpalmento dos componentes permitiu independência de implementação, a qual proporcionou mais conforto e liberdade para que os integrantes desenvolverem suas aplicações. Entretanto, este processo tornou claro o desafio de manter a comunicação ativa entre as partes. Por fim, vale ressaltar que tudo o que foi apresentado neste relatório esta passivo de modificações à medida que o grupo encontre melhores maneiras e diferentes ferramentas para tornar realidade a proposta da plataforma SIGADica.

8 Referências

- https://www.sqlalchemy.org/
- https://fastapi.tiangolo.com/
- https://hub.docker.com/_/mysql
- https://www.python.org/
- https://siga.ufrj.br/sira/repositorio-curriculo/ListaCursos.html
- https://lxml.de/
- https://flutter.dev/
- https://firebase.google.com/
- https://github.com/PatrickfBraz/SIGADica