INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

CAMPUS SÃO PAULO

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Alkindar José Ferraz Rodrigues

Carolina de Moraes Josephik

Fabio Mendes Torres

Gabriely de Jesus Santos Bicigo

Leonardo Naoki Narita

Mariana da Silva Zangrossi

Lixt

Desesenho da aplicação

São Paulo

2021

Alkindar José Ferraz Rodrigues
Carolina de Moraes Josephik
Fabio Mendes Torres
Gabriely de Jesus Santos Bicigo
Leonardo Naoki Narita
Mariana da Silva Zangrossi

Lixt

Desesenho da aplicação

Desenho de aplicação para desenvolvimento na disciplina de Projeto Integrado I no 1° semestre de 2021.
Prof. Ivan Francolin Martinez
Prof. José Braz de Araujo

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo

Campus São Paulo

Tecnologia em Análise e Desenvolvimento de Sistemas

São Paulo

2021

Lista de abreviaturas e siglas

API	Application Programming Interface — Interface de progragramação de Aplicação. Citado em 2.1
HTTPS	Hypertext Transfer Protocol — Protocolo seguro de transferência de hypertexto. Citado em 2.1
REST	Representational State Trasfer — Transferência de Representação deEstado: modelo de transferência de dados no qual o estado de um objeto é serializado e transferido entre aplicações. Citado em 2.1
ORM	Object–relational mapping — Mapeamento objeto-relacional. Citado em 2.2.2.1

1 Introdução

2 Desenvolvimento da Aplicação

Neste capítulo descrevemos os conceitos, análises e ferramentas utilizadas pela equipe TGT para o desenvolvimento do porduto Lixt, incluindo os requisitos do projeto, as tecnologias utilizadas, e a arquitetura e a modelagem do produto. Isto é apresentado para que se possa estabelecer parâmetros e métricas que guiarão o desenvolvimento e a entrega final do projeto.

2.1 Arquitetura

Com base na análise do projeto, e nos requisitos que foram levantados como necessários, a arquitetura cliente-servidor é plausível como modelo para o produto que pretendemos entregar. Esta arquitetura é composta por duas aplicações distintas:

- Uma aplicação *front-end*, focada na interação com o usúario e apresentação de dados de uma forma agradável e intuitiva. Esta aplicação será implementada em JavaScript, com o framework React Native, e disponibilizada para as plataformas iOS e Android.
- Uma aplicação *back-end*, que será responsável por tratar os dados coletados no *front-end* e disponibilizar as informações que serão mostradas aos usuários. Como esta aplicação requer uma lógica de servidor, estabilidade e ampla disponiblidade, esta aplicação será implementada em Java, com uso do framework Spring Boot, que abstrai a criação de um servidor.

Podemos ver na Figura 1 uma respresentação desta arquitetura.

A comunicação entre estes serviços será feita com o uso do protocolo HTTPS, que permite a aplicação cliente realizar chamadas ao servidor através de urls, seja para buscar informações para apresentar ao usuário ou postar informações coletadas dele. O framework Spring, além de abstrair a implementação da lógica de um servidor, implementa *listeners* para estas urls, auxiliando a criação de pontos na aplicação do servidor focados na comunicação com a aplicação cliente.

O uso do protocolo HTTPS oferece alguamas vantagens a aplicação *front-end*, que não precisa esperar uma solicitação ao *back-end* ser finalizada antes de realizar outras solicitações, aumentando a repsonsividade da aplicação cliente. Para além disso, quando combinada ao modelo REST na construção da API, o protocolo HTTP oferece meios eficientes para que as aplicações se comuniquem.

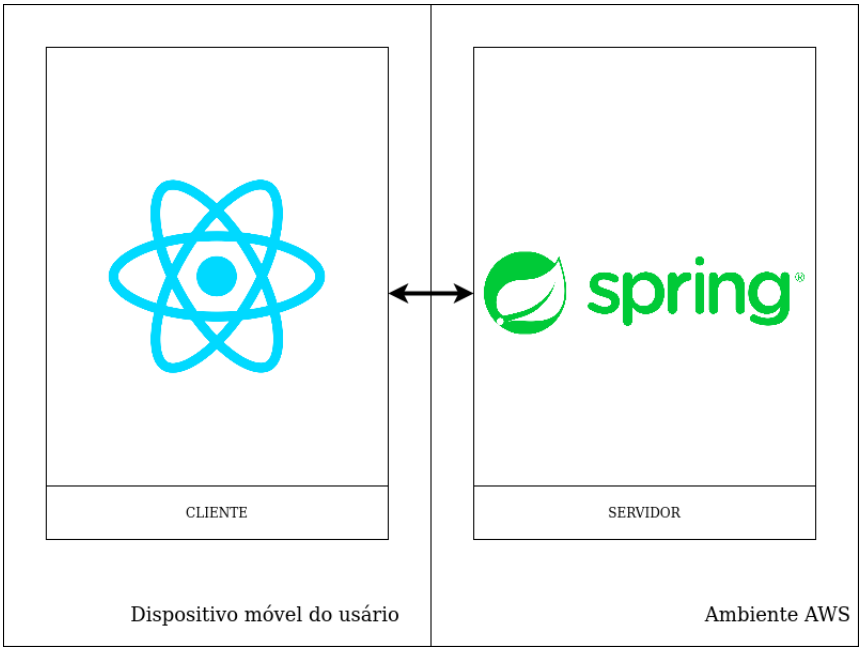

Figura 1 – Arquitetura Lixt

Como plataforma de servidor, o serviço AWS será utilizado, uma vez que é oferecida de maneira gratuita para a realização deste projeto, e nele serão armazenadas algumas instâncias da aplicação *back-end*. Esta redundância é necessária como forma de garantir a estabilidade do sistema, de forma que sempre haja alguma disponível para o processamento de novas requisições, e, em caso de falha numa delas, o serviço não seja interrompido aos usuários.

2.2 Tecnologias Utilizadas

As tecnologias que decidimos utilizar foram escolhidas a partir do conhecimento prévio da equipe, da curva de aprendizado e levando em consideração também o tamanho das comunidades que já a utilizam, visando um maior apoio e material de pesquisa. Dito isso, escolhemos as seguintes tecnologias:

2.2.1 Linguagens

2.2.1.1 Back-end

Decidimos que a linguagem para o back-end seria o Java. A linguagem se adequa à nossa proposta e atende o paradigma de linguagem orientada a objetos do qual nos foi orientado a utilizar. A comunidade de Java é extensa e ativa, contribuindo com muitos materiais e recursos. Ainda podemos destacar que a utilização da linguagem previamente pelos integrantes da equipe também foi impactante na consolidação dessa decisão.

2.2.1.2 Mobile

Para o desenvolvimento da plataforma mobile decidimos utilizar o Javascript. A linguagem possui também uma comunidade ativa e uma variedade de materiais disponíveis e atualizados. Apesar de nem todos os integrantes terem tido contato prévio, por conta da facilidade de assimilação e necessidade de poucos recursos para a configuração do ambiente de desenvolvimento, optamos pelo Javascript.

2.2.2 Frameworks e ORMs

2.2.2.1 Back-end

Para o back-end decidimos utilizar o framework Spring, usando a ferramenta Spring Boot que proporciona agilidade na criação das aplicações pois segue a filosofia de Convention over Configuration(PADMANABHAN, 2020), nos poupando de depreender muito tempo nas configurações. Não obstante, a framework facilita o desenvolvimento pois nos propicia a utilização de módulos que julgarmos necessários (como Spring MVC e Spring Data JDBC). Além disso, há uma gama vasta de materiais para consultarmos. Como ferramenta ORM decidimos usar o Hibernate pela consolidação dele no mercado e o uso amplo em aplicações Java que necessitam de mapeamento relacional dos dados. Por conta da quantidade de modelos da aplicação, julgamos necessário utilizar uma ferramenta que facilitasse esse processo.

2.2.2.2 Mobile

Na aplicação mobile decidimos utilizar o framework React-Native. Esta framework gera aplicativos nativos, não necessita de muitos recursos e configurações para montar o ambiente de desenvolvimento e possibilita um conforto maior no desenvolvimento do código por ser uma framework Javascript. Não obstante, também é uma framework com larga quantidade de recursos para consulta além de uma comunidade muito ativa.

2.2.3 Banco de dados

O banco de dados que escolhemos foi o MySQL pois precisávamos para a nossa proposta de um banco de dados relacional e que fosse possível de ser hospedado na AWS. Verificamos que o MySQL cobria não apenas esses critérios mas também possui uma ferramenta gráfica (MySQL Workbench) que facilita a visualização e a operação do banco. Além disso os integrantes da equipe já tiveram experiências com a ferramenta anteriormente.

2.2.4 Gerenciamento de tarefas

Para o gerenciamento das tarefas optamos pela ferramenta Trello por ser gratuita, de fácil manuseio e visualização. Além disso, o Trello figura entre as ferramentas que foram utilizadas com sucesso nos semestres anteriores durante o desenvolvimento de projetos.

2.3 Mantenibilidade da aplicação

É fundamental para o desenvolvimento do projeto, tanto o previsto quanto em avanços posteriores, que certos requisitos de manutenibilidade sejam estabelecidos. Estes requisitos são indispensáveis para que que a aplicação atinja um nível adequado de qualidade,

2.3.1 Code Conventions

Referências

PADMANABHAN, A. Convention over configuration. *Devopedia*, 2020. Citado na página 6.