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 1ř 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	Objectrelational 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.

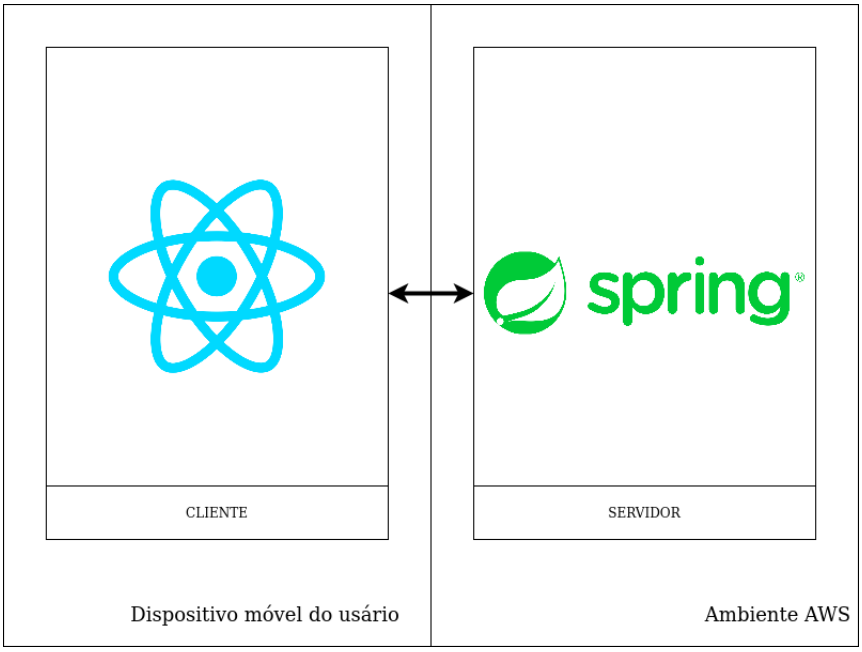

Figura 1 – Arquitetura Lixt

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.

2.2.5 Versionamento

Para o controle de versão do desenvolvimento da aplicação optamos pelo Git com repositório no Github. Esta escolha foi realizada devido à experiência prévia da equipe com a ferramenta e pela quantidade de recursos oferecidos pela plataforma na gestão do desenvolvimento de aplicações. Para o versionamento do projeto utilizamos o Apache Subversion (SVN). Concentramos no repositório fornecido pelos professores todas as entregas previstas (incluindo as versões atualizadas dos códigos do repositório externo do Github).

2.3 Escalabilidade

A escalabilidade da aplicação é sua capacidade de se adequar a um amplo volume de requisições, mantendo a estabilidade do sistema e a velocidade de respostas. Um sistema escalável está apto para responder adequadamente nestes momentos, assim como liberar seus recursos em momentos com poucas requisições.

Por se tratar de algo relativo ao processamento de requisições, a escalabilidade diz respeito ao *back-end*, que centraliza os pedidos dos usúarios. A camada cliente por outro lado, por focar exclusivamente na lógica de visualização, e rodar em dispositivos mobile, não exige tanta capacidade de processamento, e a escalabilidade não é uma preocupação para esta camada.

Como foi mencionado, a aplicação *back-end* ficará armazenada na plataforma Amazon AWS. Existe um processo da ferramenta de integração contínua Jenkins para aumentar o número de instâncias ativas em produção, e este processo será acionado em momentos com intenso volume de requisições.

2.4 Mantenibilidade da aplicação

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

2.4.1 *Logs*

Como forma de monitorar a aplicação em tempo de execução, especialmente na camada de servidor, *logs* serão usados para registrar o estado dos objetos. A ferramenta Log4j será usada, uma vez que os membros do time já tem mais familiaridade com ela. Esta ferramenta perimite o registro em diversos níveis, como

- *info*
- *debug*
- *warn*
- *error*

e pode ser configurada para que apenas os dois últimos sejam registrados no ambiente de produção. Desta forma a cada ponto de falha da aplicação um log de nível apropriado será colocado para que problemas sejam rapidamente identificados, analisados e resolvidos.

2.4.2 Integração Contínua

Visando manter o serviço sempre atualizado para o usuário, a ferramenta de integração contínua Jenkins foi selecionada para a implantação da aplicação *back-end* em produção. Ela permite, a partir do código no repositório git, a execução de testes, o build e o *deploy* para o ambiente de produção, automatizando tarefas complicadas de se executar em máquinas remotas.

Para o *front-end*, no entanto, não existem ferramentas de integração contínua, uma vez que as imagens do aplicativo mobile precisam ser aprovadas pelas lojas de aplicativo. Contudo, ainda é possível usar ferramentas de CI para a execução de testes e verificação da integridade do código a cada nova versão.

2.4.3 *Code Conventions*

As convenções de código são acordos internos ao time que visam estandartizar a forma como os diversos integrantes do time produzem seus códigos. Elas visam facilitar o entendimento mútuo entre os integrantes do time, de modo o estilo de progamação seja indistiguível e independente de seus autores. Geralmente, as convenções de código estabelecem estilos para se organizar o código textualmente, isto é, dizem respeito a forma como nomes de variáveis são escolhidas e comentários são posicionados, por exemplo.

As convenções adotadas são baseadas na especificação da SUN MICROSYSTEMS, de 1997. Esta é comumente usadas para o desenvolvimento na linguagem Java, e muito próxima do padão adotado em JavaScript, e vale destacar o seguintes pontos:

- Minimizar o uso de variáveis, funções e objetos globais.
- As declarações globais estarão preferencialmente no início do arquivo.
- Declarar as variáveis próximo do ponto onde elas serão inicializadas.
- A indentação é de 4 espaços.
- Linhas mais longas que 80 caracteres serão quebradas e indentadas a 8 espaços.
- Pacotes e variáveis com nomes curtos, em `camelCase` e substantivos.
- Classes e interfaces em `CamelCase` e substantivos.
- Métodos em `camelCase` e verbos.
- Constantes em `UPPER_CASE`.

No entanto, especificamente para a linguagem Java,

- Classes e métodos devem ser documentados com um comentário na seguinte forma, uma vez que as IDEs reconhecem este formato e formatam o text na forma de *pop-ups* quando o cursor está sobre uma referência a esta classe.

```
/**
 * Class ListService
 *
 * Implementar endpoints para as funcionalidades de lista.
 */
```

2.4.4 Testes

Testes são uma ferramenta fundamental para o desenvolvimento da aplicação, uma vez que garenatem, em tempo de compilação, o comportamento correto do aplicativo. Para além disso, testes tem um papel de documentação, uma vez que encapsulam de forma breve o comportamento esperado das classes e métodos produzidos, e podem ser consultados em caso de dúvida quando ao uso destes. Este tipo de teste é chamado de teste unitário, em oposição aos testes de integração, que verificam o funcionamento da aplicação de ponta-a-ponta, isto é, a partir de uma chamada a um endpoint, apenas os serviços externos são mimetizados, garantindo o funcionamento correto de toda a aplicação.

Desta forma, a construção de testes, de ambos os tipos é de extrema importância para a elaboração do projeto visando a sua manutenibilidade, e será o primeiro passo de uma sprint, após o planejamento, a construção de testes relevantes para a tarefa, seguindo

os princípios do TDD. Como as ferramentas de teste são específicas de cada linguagem, cada camada da aplicação fará uso de frameworks distintos.

O *back-end* será testado com o framework JUnit, fazendo uso da biblioteca Mockito quando necessário simular comportamentos de objetos que não são o alvo da suite de teste. Este framework ainda oferece ferramentas para testar o banco de dados, ou melhor, testar a conexão com o banco e verificar o comportamento das classes de acesso a ele. Já os teste de *front-end* serão feitos com a biblioteca Jest, que auxilia a construção de testes unitários, e por se tratar de uma GUI, as interções de usuário devem ser simuladas também. Para tanto, a blioteca React Testing Library será usada.

2.5 Viabilidade Financeira

O projeto de análise de viabilidade financeira consiste em averiguar a garantia de lucro sobre as despesas do projeto. Portanto, nesse projeto será descrito cada processo a fim de fazer essa verificação.

2.5.1 Gerenciamento de Custos

Nesse tópico, serão abordados temas de investimento inicial e de desenvolvimento do projeto, incluindo tópicos de análise de requisitos, desenvolvimento, manutenções e imprevistos.

2.5.1.1 Análise de Requisitos e Desenvolvimento

Para iniciar o projeto, é necessário fazer os primeiros planejamentos, elicitação de requisitos, abstrair e concretizar as primeiras ideias e fazer os primeiros planejamentos (diagramas, cronogramas e documentação). Logo após, o projeto chega na fase de desenvolvimento, onde é começado a se tornar real.

Contudo, o projeto não vai possuir nenhum custo de análise e implementação do sistema, devido ao fato de ser um projeto educacional.

2.5.1.2 Manutenções

Inevitavelmente, manutenções do sistema ocorrerão pós finalização do projeto e estar devidamente funcional em produção. Contudo, os custos de manutenções também não serão cobrados, devido a ser um projeto educacional.

2.5.2 Custos de deploy e de Ambiente de Produção

Nesse tópico, são apresentados os custos de manter o sistema funcional e disponível para os usuários. Desse modo, será feito uma previsão anual de cada plataforma utilizada:

2.5.2.1 Frontend

Tendo em vista que o projeto é *mobile* voltado para dispositivos Android, será publicado na PlayStore, estimando um valor de 25.00 USD anual.

2.5.2.2 Backend

Inicialmente gratuíto no Amazon EC2, sendo permitido 750h de instâncias por mês durante o período de 12 meses.

A partir do momento que for necessário grande porte, será indicado o plano Sob Demanda do Amazon EC2, que garante viabilidade econômica e estratégica (visto que o preço é calculado a partir do uso).

Nesse plano, o custo de transferência de dados (ou seja, entrada e saída de dados) geram o valor de, na região de São Paulo, no máximo 0,15 USD por GB.

Na Figura 2, seguem os preços dos planos Sob Demanda na região de São Paulo para Linux usando tipo de instância geral com 1 vCPU.

Visualizando 4 de 231 instâncias disponíveis

1

Nome da instância	Taxa horária sob demanda	vCPU	Memória	Armazenamento	Performance das redes
t2.nano	0,0093 USD	1	0.5 GiB	Somente EBS	Baixo
t2.micro	0,0186 USD	1	1 GiB	Somente EBS	Baixo a moderado
t2.small	0,0372 USD	1	2 GiB	Somente EBS	Baixo a moderado
m6g.medium	0,0612 USD	1	4 GiB	Somente EBS	Até 10 gigabits

Figura 2 – Preços do Amazon EC2 - Sob Demanda

2.5.2.3 Banco de Dados

Inicialmente gratuíto no Amazon RDS, sendo permitido 750h de instâncias durante o período de 12 meses. O Amazon RDS possui suporte a vários Sistemas Gerenciadores de Banco de Dados (SGBD), incluindo o MySQL, que foi o SBGD optado para desenvolver a aplicação Lixt.

A partir do momento que for necessário grande porte, será indicado o plano Sob Demanda do Amazon RDS, que garante viabilidade econômica e estratégica (visto que o preço é calculado a partir do uso).

Na Figura 3, seguem os preços dos planos Sob Demanda na região do Leste dos EUA (única opção disponível em 07/06/2021).

Instâncias de uso geral - Geração atual	vCPU	Memória	Preço por hora
db.mv11.medium	1	4 GiB	0,084 USD
db.mv11.large	2	8 GiB	0,168 USD
db.mv11.xlarge	4	16 GiB	0,336 USD
db.mv11.2xlarge	8	32 GiB	0,672 USD
db.mv11.4xlarge	16	64 GiB	1,344 USD
db.mv11.12xlarge	48	192 GiB	4,032 USD
db.mv11.24xlarge	96	384 GiB	8,064 USD

Figura 3 – Preços do Amazon RDS - Sob Demanda

2.5.3 Medidas de Obtenção de Retorno Financeiro

Para gerar uma receita positiva a fim de obter lucro, haverá duas formas principais de retorno financeiro:

- Cobrança do aplicativo: O aplicativo estará disponível gratuitamente na PlayStore, não gerando, portanto, retorno financeiro.
- Propaganda/Recomendação: Será utilizado mediador de anuncio AdMob (responsável por conectar aplicações e anunciantes), onde o valor varia por visualizações de anúncios e cliques neles. Contudo, no próprio site do admob, é citado um caso no qual houve 300.000 downloads e recebia, através do AdMob, 100 USD por dia.

Glossário

API Uma Interface de Progamação de Aplicação (*Application Programming Interface*), são pontos que a aplicação expoõe para permitir que usuários ou serviços externos executem tarefas dentro da aplicação. 4

back-end Um sistema *back-end* é aquele que encontra na camada de servidor, em uma aplicação de duas camadas. Sua principal função é fornecer informações e capacidade de processamento a aplicação cliente. 4, 5

front-end Um sistema *front-end* é aquele que encontra na camada cliente, em uma aplicação de duas camadas. Sua principal função, no escopo deste projeto, é atuar como interface gráfica para o usário, coletar dados e enviá-los par o *back-end*. 4

Referências

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

SUN MICROSYSTEMS, I. Java coding conventions. 1997. Citado na página 8.