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

SDK	Software Development Kit — Kit de desenvolvimento de software. Citado em 2.3
HTML	HyperText Markup Language — Linguagem de Marcação de Hipertexto. Citado em 2.3
CSS	Cascading Style Sheets — Folhas de Estilo em Cascata. Citado em 2.3
API	Application Programming Interface — Interface de progragramação de Aplicação. Citado em 3.1
HTTPS	Hypertext Transfer Protocol — Protocolo seguro de transferência de hypertexto. Citado em 3.1
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 3.1
ORM	Object–relational mapping — Mapeamento objeto-relacional. Citado em 3.2.2.1

1 Introdução

1.1 Contextualização

Comprar é um ato essencial no cotidiano das pessoas. Não apenas por uma questão de sobrevivência, como a compra de alimentos, medicamentos, roupas, imóveis, móveis, entre outros, mas também para lazer. Quando trata-se de compras essencias, as compras de supermercado estão no topo da lista para a maioria das pessoas, pois costumam ser compras muito frequentes (semanalmente, 2 vezes por semana, mensalmente) e com alto volume de produtos, que não envolvem apenas alimentação em si, mas também produtos de higiene da casa e pessoal.

Devido a sua grande importância na vida das pessoas, as compras de supermercados devem ter sua devida atenção e gerenciamento, principalmente por questões financeiras e organização, pois o principal objetivo de muitos consumidores é comprar mais produtos de qualidade gastando menos.

1.2 Problematização

Por ser algo extremamente comum e cotidiano na vida de todas as famílias do país, a tendência é que as pessoas não saibam administrar seus gastos em supermercados, já que as compras ocorrem frequentemente. Atualmente, a forma universal de verificar os gastos de uma compra de um supermercado no Brasil é através do cupom fiscal, um documento frágil que contém todos os produtos comprados e seus respectivos preços. Sua fragilidade é devido a sua composição, o que ocasiona o desaparecimento do texto ao decorrer do tempo. Ao se depender do cupom fiscal para analisar e atualizar os gastos, ocorrem alguns dos seguintes problemas:

- Só é possível fazer uma análise de gastos após a compra, sendo que o consumidor teria grandes benefícios ao analisar os gastos e produtos durante a compra.
- Dificuldade de lembrar e guardar os preços dos alimentos e produtos durante a compra, para futura análise que auxilie o comprador a fazer compras mais econômicas e de qualidade.
- É necessário repassar manualmente todos os dados de cada cupom fiscal para um documento centralizado, se caso o consumidor queira manter um histórico de compra ou fazer uma análise geral.

- Dificuldade em analisar preços de diferentes estabelecimentos de modo a decidir qual o melhor custo-benefício julgado pelo comprador, já que essa informação vai provavelmente constar em algum meio não centralizado, ou seja, o consumidor irá perder a informação rapidamente.
- Dificuldade em analisar e calcular quais as categorias de produtos que possuem maior gasto, menor gasto, ou uma média, baseados em determinado período, que visa facilitar a descoberta de quais produtos são mais comprados, quais são mais caros, a fim de auxiliar o usuário em futuras compras.
- Dificuldade de gerenciar compras colaborativas, cada participante não sabe exatamente o que será comprado ou quais itens serão de sua responsabilidade, a não ser que essas informações sejam compartilhadas através de meios que a ação manual seria necessária, como mandar uma mensagem através de um aplicativo.

1.3 Justificativa

Como é possível perceber com base nos problemas descritos da seção acima, o gerenciamento de compras, quando feito, é uma atividade muito cansativa e manual. Com a frequência de compras somando a dificuldade de administrá-las, as pessoas não analisam e gerenciam suas compras, o que as fazem gastar cada vez mais e não saberem o motivo de tanto gasto, pois não conseguem fazer uma análise de compras detalhada.

Tendo em vista a economia do Brasil, muitas famílias tentam economizar com grande esforço, tanto para guardar o restante do dinheiro no final do mês, como também para sobreviver em casos em que a renda é muito baixa. Então, o gerenciamento e administração de compras é extremamente essencial para o presente e futuro de famílias e pessoas que se encontram com o dinheiro contado ou que precisam investir determinada parte do valor para os filhos, casa, pagamento de dívidas, entre outros.

1.4 Objetivos

Para solucionar todos os problemas citados e no que eles acarretam, entregando autonomia, controle de compras para os consumidores de maneira intuitiva e fácil, em que todas as informações que eles registrem sejam armazenadas em um local centralizado, criamos o Lixt, uma solução que facilite a vida do consumidor que reside no Brasil, para o gerenciamento e controle de suas compras através de criação e edição de listas de compras compartilhadas ou individuais. A solução será focada em facilitar as compras alimentícias, feitas em supermercados, feiras, lojas de conveniência, e até mesmo em aplicativos de alimentos, como IFood, Uber Eats, Rappi, e entre outros.

Além disso, o Lixt se propõe em oferecer análises de compras por determinado período escolhido pelo usuário, separar produtos em categorias e até mesmo apresentar análises quanto a variação de preço entre estabelecimentos, que poderão ser adicionados de acordo com a preferência do usuário, além de outras funcionalidades que estarão melhores descritas na seção de Escopo.

1.5 Análise da Concorrência

Auditamos soluções que existem atualmente no mercado e, ao verificar as aplicações existentes, conclui-se que há intersecções nas funções dentre os aplicativos analisados. As funções mais básicas, como gerenciamento de itens e gerenciamento de listas, estão presentes em todos, tendo em vista que são essenciais em qualquer aplicativo de lista. Outras funções básicas que deveriam ser incluídas em qualquer aplicação de lista, como gerenciamento de categorias e compartilhamento de listas, não estão presentes em todos os aplicativos analisados.

Contudo, as divergências ficam claras quando analisamos o mecanismo das aplicações, entre elas destacam-se o *Mealime* e o *Cozi Family Organizer* que, apesar de serem voltados para as compras, cumprem também outras funcionalidades. O *Mealime*, cujo foco é o planejamento de refeições, e o *Cozi Family Organizer*, cujo foco é o planejamento familiar, deixam a desejar nas funções relacionadas às compras.

Entre os outros aplicativos analisados, é perceptível que não possuem todas as funcionalidades propostas nesse documento, principalmente quando se trata de compartilhamento de listas, uma vez que cada software lida de modo diferente diante dessa feature. O *SoftList*, por exemplo, permite o compartilhamento de lista, porém não é capaz de ser gerenciada por mais de um usuário, sendo apenas importada para o usuário no qual a lista está sendo compartilhada.

Ao analisar os aplicativos mais populares da categoria, constatamos que o *Out Of Milk*, *Bring!* e o *OurGroceries*, que são destaques na área, não se propõem a exibir análise estatística das compras do usuário e nem manter um histórico do que foi comprado. A tabela 1.5 permite visualizar melhor as diferenças entre os concorrentes.

	Cozi Family Organizer	OurGroceries	Softlist	Out of Milk	Mealime	Bring	Lixt
Login/Cadastro	x	x	x	x	x	x	x
Categorias		x	x	x		x	x
Compartilhamento de listas	x		x	x		x	x
Atribuição de itens							x
Gerenciamento de compras			x				x
Historico de compras			x				x
Análise de compras			x				x
Calculadora			x	x		x	x
Comentários							x

Tabela 1.5: Uma comparação dos aplicativos concorrentes.

2 Revisão Bibliográfica

Nesta seção, os tópicos conceituais e teóricos relacionados ao projeto a ser desenvolvido que irão auxiliar o entendimento do mesmo serão descritos de forma a esclarecer tanto as partes de negócio, como as técnicas. Dessa forma, a compreensão do projeto como todo será mais fácil para o leitor.

2.1 Organização financeira

De acordo com o dicionário Michaelis (**??**), a palavra organização significa preparação de um projeto, com definição de procedimentos e metas. Assim, pode se dizer que organização financeira trata-se de cuidar das finanças (podendo elas ser empresarial, familiar, ou pessoal) afim de atingir um objetivo.

Quando trata-se de finanças pessoais ou familiares, ter uma boa organização financeira significa conhecer quais e quanto são suas despesas e receitas mensais (**??**). De acordo com o mesmo artigo, é necessário planejar como, quanto e onde a despesa será feita, além de fazer levantamento de preços.

Pode-se dizer então, que as compras, categorizadas como despesas fixas, devido a sua presença frequênte em ambientes pessoais e familiares, precisam ser planejadas e organizadas para impulsionar uma melhor organização financeira entre os envolvidos da compra.

2.2 Listas

As listas de afazeres, comumente chamadas de *to do list* em inglês, é uma lista de lembretes textuais, como por exemplo, "Ir ao médico", "Comprar caixa de 12 ovos", entre outros (**??**). Essas listas são encontradas em qualquer âmbito, e auxiliam a lembrar de tarefas ou coisas importantes que precisam sobrer uma ação do indívidio, por exemplo, comprar, completar, finalizar.

Seguindo o mesmo príncipio, uma lista de compras se refere à uma lista contendo nomes de produtos nos quais um indivíduo deseja adquirir, podendo conter ou não um limite de gasto total. As listas, além de ser um método eficaz para se organizar e lembrar do que foi escrito, é considerada uma ótima maneira de evitar que alguém ceda à produtos nos quais não quer comprar por serem prejudicias à saúde da pessoa (**??**) ou por qualquer outro motivo.

Existem três tipos de compras (**??**), sendo que dois tipos podem ter a presença de uma lista de compras:

1. Completamente planejada, incluindo na lista de compras os produtos e suas respectivas marcas.
2. Parcialmente planejada, incluindo apenas os produtos a serem comprados mas a marca deles serão decididas no ato da compra.

As compras completamente planejadas são consideras com pouco envolvimento emocional do consumidor, enquanto as parcialmente planejadas, apesar de planejadas, podem ser manipuladas por algum critério exterior, como uma promoção (**??**). No aplicativo a ser desenvolvido, uma lista de compras pode ser tanto completamente planejada ou parcialmente planejada, de acordo com a necessidade do usuário que está utilizando a plataforma. Uma vez que a lista de compras existe, a compra em si se torna planejada.

2.3 Aplicativo *Mobile*

Um aplicativo *mobile* trata-se de uma aplicação de software que é desenvolvida exclusivamente ou acessável por um dispositivo móvel, como celulares e *smartphones*. O uso dos dispositivos móveis cresceu disparadamente nos últimos anos, e é o principal meio de acesso no Brasil desde 2017 (**??**).

Atualmente, existem dois sistemas operacionais que são mais utilizados nas plataformas *mobile*: o *Android* e o *IOS*, tratados como concorrentes das marcas *Google* e *Apple*. Os aplicativos desenvolvidos para os dois sistemas operacionais são divididos em dois tipos:

- Aplicativos nativos.
- Aplicativos híbridos.

Os aplicativos nativos são desenvolvidos utilizando o SDK do sistema operacional em questão, e não pode ser executados em outros ambientes. Por exemplo, um aplicativo desenvolvido com o SDK da empresa *Apple* não pode ser executado em ambiente *Android* e vice-versa (**??**).

Já os aplicativos híbridos são desenvolvidos utilizando tecnologias web, como HTML, CSS e *Javascript*, e por isso, podem ser executados em qualquer sistema operacionais móvel através de um container nativo. Apesar de poder ser executado em diversos dispositivos através de apenas um código, os aplicativos híbridos possuem certa limitações ao acessar recursos nativos do dispositivo, como câmeras, microfones e sistemas internos

de armazenamento e memória, porém, atualmente já existem soluções que facilitam a comunicação entre aplicativo híbrido e o dispositivo, como o *React Native* (**??**).

3 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.

3.1 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.

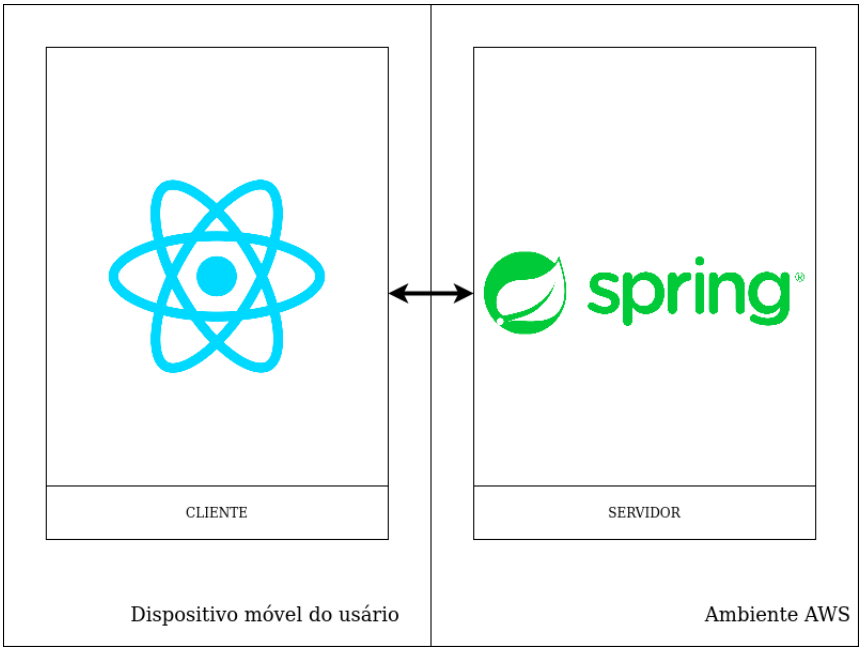

Figura 1 – Arquitetura Lixt

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.

3.2 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:

3.2.1 Linguagens

3.2.1.1 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.

3.2.1.2 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.

3.2.2 Frameworks e ORMs

3.2.2.1 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.

3.2.2.2 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.

3.2.3 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.

3.2.4 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.

3.2.5 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).

3.3 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.

3.4 Manutenibilidade 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

3.4.1 *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.

3.4.2 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.

3.4.3 *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.
 */
```

3.4.4 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.

3.5 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.

3.5.1 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.

3.5.1.1 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.

3.5.1.2 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.

3.5.2 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:

3.5.2.1 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.

3.5.2.2 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
(AMAZON, 2021b)

3.5.2.3 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
(AMAZON, 2021a)

3.5.3 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. (GOOGLE, 2021)

4 Planejamento e Gerenciamento do Projeto

As próximas seções possuem o objetivo de descrever mais sobre como a equipe e o projeto estão organizados quanto a metodologia, gestão de tempo e papéis de atuação.

4.1 Metodologia de Gestão e Desenvolvimento de Projeto

O grupo optou pelo uso de uma metodologia ágil para o projeto e, após algumas reuniões, foi decidido que o que guiaria o processo de desenvolvimento seria uma junção de frameworks extremamente úteis para a melhor performance do time, o Scrum e o Kanban. O Scrum se baseia na divisão do projeto em vários ciclos de atividades - conhecidos como Sprints - com diversas cerimônias e interações frequentes da equipe a fim de alinhar o que tem sido feito, tratar impedimentos e pensar em maneiras de otimizar o processo de trabalho, aumentando a agilidade no desenvolvimento. Já o Kanban é um método de controle e gestão de forma visual, geralmente feito com o uso de post-its coloridos que reforçam a simbologia das tarefas e ações que precisam ser feitas, estão sendo feitas, ou foram concluídas.

Dentro das metodologias ágeis é muito comum - e essencial - a definição do Product Backlog contendo todas as funcionalidades desejadas na aplicação pelo cliente e elencadas por prioridade. Esse backlog é, posteriormente, dividido em tarefas que serão distribuídas em cada Sprint, gerando o Sprint Backlog. É nesse momento em que, no projeto, o Kanban será aplicado. Dessa forma, o Scrum será utilizado para a determinação do método de desenvolvimento iterativo e incremental e o Kanban será utilizado para métricas e fluxos de produção da equipe.

4.2 Organização da Equipe

As primeiras reuniões de alinhamento entre os membros da equipe foram essenciais para que os integrantes pudessem interagir e compartilhar mais sobre quais conhecimentos prévios possuem, quais são suas áreas de maior facilidade e interesse e dessa forma planejar uma estratégia de desenvolvimento que, apesar de desafiadora, também seja inclusiva a todos do grupo.

No Scrum, metodologia escolhida para o projeto, a definição de alguns papéis é necessária. São eles:

- **Product Owner**: É o responsável pelos interesses do cliente no projeto, captando, interpretando e repassando ao time de desenvolvimento suas necessidades. Quem

assumiu este papel foi o Fabio Mendes;

- **Scrum Master**: É o facilitador no desenvolvimento do projeto, cuidando para que as cerimônias do Scrum sejam devidamente cumpridas e ajudando a resolver os impedimentos que possam atrapalhar a equipe de desenvolvimento. Quem assumiu este papel foi a Carolina de Moraes;
- **Development Team**: Time que cuida do desenvolvimento técnico do projeto. Quem assumiu este papel foram Alkindar Rodrigues, Gabriely Bicigo, Leonardo Naoki e Mariana Zangrossi.

É importante salientar que, apesar da divisão de papéis, todos os participantes estão contribuindo no desenvolvimento com o objetivo de entregar o projeto na data estimada.

Para uma divisão mais organizada e formal de tarefas, foi montada uma tabela de responsabilidades dos integrantes referentes aos tópicos de desenvolvimento do projeto.

Atividade	**Alkindar**	**Carolina**	**Fabio**	**Gabriely**	**Leonardo**	**Mariana**
Back-End	x		x		x	
Banco de dados	x		x		x	
Blog	x	x	x	x	x	x
Documentação	x	x	x	x	x	x
Front-End		x		x		x
Vídeos		x		x		x

Tabela de responsabilidades

4.3 Gestão de Tempo

Cada Sprint do projeto terá a duração de 2 semanas e respeitará todas as cerimônias do Scrum. Antes de dar início a uma nova Sprint ocorrerá a realização da Sprint Planning, reunião de planejamento onde serão definidos os entregáveis da Sprint a ser iniciada. Após o início da iteração, as dailies serão feitas através do grupo de mensagens instantâneas do time. Ao fim do período de 2 semanas, ocorrerá a Sprint Review, reunião com o objetivo de avaliar o que foi entregue com sucesso naquela Sprint, e também a Sprint Retrospective, reunião onde a equipe conversa sobre os pontos positivos e negativos da última iteração e o que pode ser melhorado para a próxima Sprint. Todas essas cerimônias foram incluídas no planejamento de gestão de tempo para o cálculo da quantidade de Sprints necessárias até a entrega final do projeto.

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. 10

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. 10, 11, 12, 13, 14, 16

deploy Processo pelo qual a aplicação é implantada em ambiente de prpdução, e está disponível para os usuários finais. 14

framework Um conjunto de pacotes e bibliotecas que abstarai alguma função complexa, geralmente de nível mais baixo, e sobre o qual uma aplicação pdoe ser contruída. Ex: o framework web Spring asbtrai a lógica de implementação de um servidor, auxiliando a criação de endpoints, rotas e processamento de HTTP. 10, 12

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*. 10, 14, 16

GUI Interface gráfica de usuário é uma forma visual de se apresentar dados e coletar interações com o usuário, em oposição a linha de comando, que funciona por texto apenas. 16

REST A Tranfêrencia por Representação de Estado é uma forma de se transferir dados na qual os atirbutos de um objeto (seu estado) são serializados em um arquivo de formato específico, e o objeto pode ser reconstituído na aplicação que recebe o arquivo. 10

Referências

AMAZON. Amazon rds on vmware pricing. 2021. Citado na página 18.

AMAZON. Preço sob demanda do amazon ec2. 2021. Citado na página 17.

GOOGLE. Quanto é possível lucrar com a admob. 2021. Citado na página 18.

PADMANABHAN, A. Convention over configuration. *Devopedia*, 2020. Citado na página 12.

SUN MICROSYSTEMS, I. Java coding conventions. 1997. Citado na página 14.