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 tabelas

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
MVP	*Minimum Viable Product* — Mínimo Produto Viável. Citado em 2.2
LGPD	Lei Geral de Proteção de Dados. — Citado em 2.4

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.

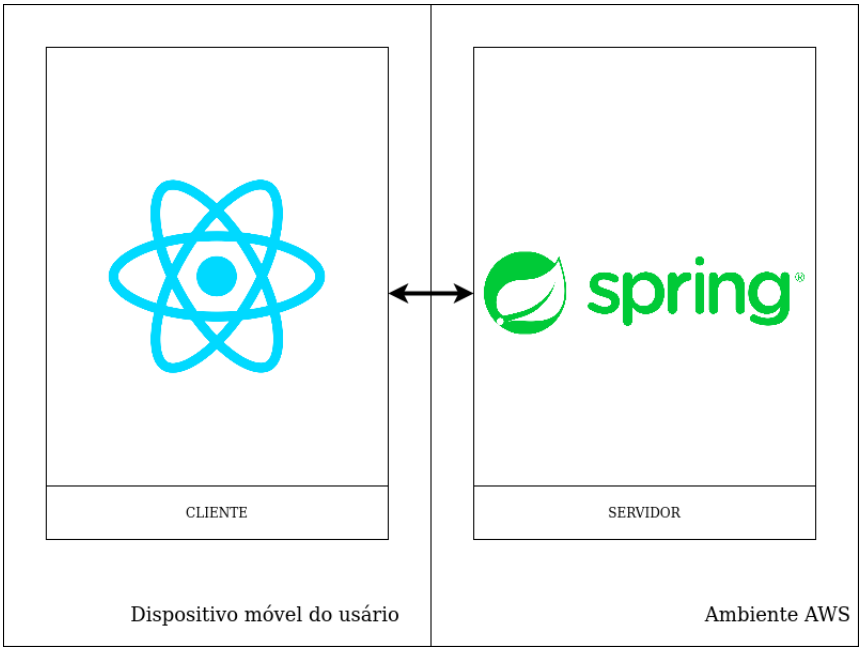

Figura 1 – Arquitetura Lixt

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 Escopo do Projeto

Lixt é um aplicativo para gerenciamento de listas de compras compartilhadas ou não.

O aplicativo vai seguir a dinâmica de uso abaixo:

1. O usuário cria uma lista de compras e insere todos os itens antes da compra;
2. O usuário inicia um carrinho de compras quando chega ao mercado, nesse momento ele tem a opção de importar para aquele carrinho os itens das listas que ele possui em aberto (que ainda não foram finalizadas), marcando quais listas ele deseja que sejam incluídas. Com o carrinho de compras ele poderá anotar os preços dos itens, riscá-los e ver o total gasto. Ao finalizar a compra as listas são atualizadas, e já aparecem riscados os itens que já foram comprados;
3. Quando o usuário definir que uma lista não é mais relevante ele poderá deletar a lista ou desmarcar todos os itens, para reutilizar a lista.

A seguir listamos as principais funcionalidades como uma lista de tópicos para facilitar a visualização de quais funcionalidades dependem de outras de forma hierárquica:

- *Login*:
  - Criar conta;
  - Redefinir senha;
- Editar uma lista:
  - Adicionar item:
    - Definir nome;
    - Definir quantidade;
    - Definir unidade de medida (un. ml, L etc.);
    - Definir medida;
    - Adicionar uma categoria:
      - Criar nova categoria;
    - Adicionar um comentário;
    - Atribuir a um usuário (caso a lista tenha sido compartilhada e pelo menos um convite já tenha sido aceito);
  - Remover itens;
  - Convidar pessoas para a lista (apenas quem criou a lista):
    - Enviar convite:
      - Acompanhar *status* (aceito ou pendente);
      - Remover convite;
  - Deletar uma lista;
  - Limpar uma lista;
- Iniciar um carrinho de compras:
  - Selecionar listas para compor o carrinho;
  - Informar o mercado onde a compra será realizada (automaticamente através da localização, se não estiver habilitada será solicitado que o usuário insira o nome do mercado);
  - Exibir o valor total do carrinho;
  - Informar a quantidade que será efetivamente comprada naquele momento (o usuário pode ter planejado 10 unidades e apenas comprar 5 naquele momento);
  - Riscar itens;

  - Finalizar um carrinho de compras;

- Ver estatísticas:
  - Selecionar uma lista e ver o total gasto naquela lista ao longo do tempo em um gráfico de linha:
    - Selecionar um dos pontos do gráfico e ver detalhes daquela lista;
  - Selecionar uma lista para ver um gráfico de pizza com os valores médios gastos por categorias naquela lista;
  - Verificar histórico de preços de um item (tabela com nome do produto, quantidade, marca, preço, mercado e data da compra):
    - Selecionar uma lista, dentro da lista selecionada selecionar o produto para ver o histórico;

Para o *Minimum Viable Product* (MVP) vamos implementar as seguintes funcionalidades, as demais ficarão para o próximo semestre:

- *Login*:
  - Criar conta;
  - Redefinir senha;
- Criar lista de compra:
  - Atribuir um nome;
  - Atribuir uma descrição;
  - Importar uma lista anterior;
- Editar uma lista:
  - Adicionar item:
    - Definir nome;
    - Definir quantidade;
    - Definir unidade de medida (un., ml., L etc);
    - Definir medida;
    - Adicionar a uma categoria:
      - Criar uma nova categoria;
    - Adicionar um comentário;
    - Atribuir a um usuário (caso a lista tenha sido compartilhada e pelo menos um convite já tenha sido aceito);

  - Remover itens;
  - Convidar pessoas para a lista (apenas quem criou a lista):
    - Enviar convite:
      - Acompanhar status (aceito ou pendente);
      - Remover convite;
  - Deletar uma lista;
  - Limpar uma lista;

- Iniciar um carrinho de compras:
  - Selecionar listas para compor o carrinho;
  - Informar o mercado onde a compra será realizada (será solicitado que o usuário insira o nome do mercado manualmente);
  - Exibir o valor total do carrinho;
  - Informar a quantidade que será efetivamente comprada naquele momento (o usuário pode ter planejado 10 unidades e apenas comprar 5 naquele momento);
  - Riscar itens;
  - Finalizar o carrinho de compras.

2.2.0.1 Requisitos Funcionais

Os requisitos funcionais dizem respeito às funcionalidade que o sistema deve ter. A Tabela 1 lista os requisitos funcionais, suas dependências, a sigla e a prioridade de implementação.

Tabela 1 – Requisitos funcionais

Sigla	**Descrição**	**Prioridade**	**Dependências**
RF01	*Login*: o usuário deve ser capaz de criar sua conta no aplicativo, definir sua senha e realizar o *login* no sistema.	Alta	
RF02	O sistema deve possibilitar que o usuário crie suas listas de compras e possa atribuir um nome, uma descrição e ter a opção de importar uma lista existente.	Alta	RF01
RF03	Editar uma lista: Possibilita ao usuário o gerenciamento dos itens da lista, como adicionar itens, remover e enviar convites para a lista.	Alta	RF02
RF04	Iniciar um carrinho de compras: permitir que o usuário importe várias listas de compras, informe o local da compra, o total gasto, quantidade de itens a ser comprados, riscar itens e finalizar o carrinho.	Alta	RF03
RF05	Ver estatísticas: ver o histórico de valores pagos em uma lista ao longo do tempo, ver os valores gastos por categorias em uma lista, ver o histórico de preços de um determinado item ao longo do tempo.	Média	RF04

Fonte: Os Autores

2.2.0.2 Requisitos Não Funcionais

De maneira simplificada, os requisitos não funcionais não estão relacionados diretamente às funcionalidades do sistema, mas ao seu funcionamento de um modo geral, ou seja, como ele as funcionalidade serão executadas.

Na Tabela 2 estão elencados os requisitos não funcionais, cada um com sua nomenclatura, categoria e descrição.

Tabela 2 – Requisitos Não funcionais

Nomenclatura	**Descrição**	**Categoria**
RNF01	Criptografia das senhas: Por uma questão de segurança as senhas não serão armazenadas diretamente no banco, serão criptografadas antes de serem armazenadas como um *hash*.	Segurança
RNF02	Comunicação: A comunicação entre as camadas da aplicação deverá ser feita utilizando o protocolo HTTPS, para garantir a segurança no envio dos dados através da rede.	Segurança
RNF03	Responsividade: O sistema deve exibir corretamente os elementos da interface gráfica nos mais variados tamanhos de celulares.	Usabilidade
RNF04	Internacionalização: O sistema deverá suportar dois idiomas (inglês e português) e suportar que futuramente seja possível adicionar outros idiomas.	Usabilidade
RNF05	Escalabilidade: O sistema deverá ser projetado para garantir que futuras melhoras e expansões sejam possíveis.	Desempenho
RNF06	Disponibilidade: O sistema deverá estar disponível aos usuários ininterruptamente	Disponibilidade

Fonte: Os Autores

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

2.4 Segurança, Privacidade e Legislação

A principal lei brasileira que trata de tratamento de dados nos meios digitais é a lei N° 13.709, sancionada em 14 de Agosto de 2018 (BRASIL, 2018) e que entrou em vigor no ano de 2020, conhecida como Lei Geral de Proteção de Dados (LGPD).

Seguindo o disposto no Artigo 6°, inciso terceiro da LGPD, a aplicação vai coletar o mínimo de dados do usuário necessários para uso da aplicação, os dados serão: localização,

endereço de email e nome do usuário. Sendo facultativo ao usuário ativar ou não a sua localização.

O design da aplicação vai seguir o princípio da transparência. O usuário do aplicativo será informado de quais as informações que serão coletadas. O próprio sistema Android, por padrão, solicita ao usuário permissão para uso da localização, um dos dados que será necessário coletar, caso o usuário opte pela detecção do local da compra automaticamente.

Como a maioria dos sistemas atuais, utilizaremos uma API (Application Program Interface) para realizar a comunicação e transferência de dados entre a interface de usuário e o servidor. Essa arquitetura possui intrinsecamente vulnerabilidades conhecidas e quando não são bem projetadas as API's podem ser um dos pontos fracos do sistema quando se trata de segurança. A empresa de consultoria Gartner (LAMBA, 2019) prevê que a tendência é que em 2022 as API's se tornem o principal foco de ataques cibernéticos.

Cientes desse cenário, foi definido que o sistema deverá seguir algumas boas práticas de desenvolvimento, citadas brevemente a seguir:

- Autenticação: As requisições apenas serão aceitas se o usuário estiver logado no sistema;
- Criptografia: Para evitar ataques do tipo man-in-the-middle as mensagens entre cliente e servidor serão criptografadas, seguindo o protocolo HTTPS;
- Documentação dos *endpoints* sensíveis: Será feito um levantamento de todos os *endpoints* que acessam informações sensíveis, para garantir que apenas usuários autenticados tenham acesso;

Como medida de segurança, o banco de dados não irá armazenar as senhas das contas dos usuários e sim um *hash* da senha, e este será comparado com o *hash* da senha informado no momento do login. Ainda como uma forma de segurança, ao cadastrar a senha ou mudar a senha atual o usuário não poderá cadastrar senha que não possua menos de seis dígitos, e que não tenham letras maiúsculas e minúsculas e números, e deverá ter no mínimo um caractere especial.

Referências

BRASIL. *Lei Geral de Proteção de Dados*. 2018. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em: 7 jun. 2021. Citado na página 12.

LAMBA, A. Api design principles and security best practices – accelerate your business without compromising security. *SSRN Electronic Journal*, 01 2019. Citado na página 13.