UNIVERSIDADE ANHEMBI MORUMBI

CAROLINE BOMFIM DO ESPÍRITO SANTO

KAIO AUGUSTO VITORINO

RALDNEY SAMPAIO ALVES

RAFAEL SOUSA

PIZZALANDIA: APLICATIVO ANDROID PARA COMPRA DE PIZZA

São Paulo

2017

CAROLINE BOMFIM DO ESPÍRITO SANTO

KAIO AUGUSTO VITORINO

RALDNEY SAMPAIO ALVES

RAFAEL SOUSA

PIZZALANDIA: APLICATIVO ANDROID PARA COMPRA DE PIZZA

Trabalho de Computação Móvel apresentado ao Prof Ricardo S Jacomini para obtenção de nota parcial para matéria.

São Paulo

2017

RESUMO

O presente trabalho consiste no desenvolvimento de um aplicativo Android com seu tema principal em delivery de pizza, chamado Pizzalandia. Seu objetivo principal é proporcionar facilidade na realização de pedidos de pizzas através da tecnologia.

PALAVRAS CHAVE: Tecnologia, pizza, Android

SUMARIO

1 INTRODUÇÃO

Com o avanço tecnológico, muitas empresas tradicionais estão se adaptando para atingir públicos adeptos da tecnologia e com isso necessitam de inovação tecnológica nos seus setores.

Para isso o aplicativo Pizzalandia foi desenvolvido, com foco na área de delivery, dessa forma as pessoas poderam realizar seus pedidos com mais facilidade e manter um histórico para futuras análises.

O Pizzalandia foi desenvolvido em android, usando a linguagem de programação Java.

1.1 OBJETIVO

Facilitar a comunicação entre instituição e cliente final através de um aplicativo

2 MODELAGEM DO PIZZALANDIA

2.1 LEVANTAMENTO DE REQUISITOS

Após a coleta das informações referentes ao aplicativo, de acordo com padrões estabelecidos na Engenharia de Software, foram analisados os requisitos funcionais e não funcionais do aplicativo. Nesta fase foram identificadas as necessidades do Pizzalandia e levantadas as principais funcionalidades que o software deveria oferecer.

2.1.1 Requisitos Funcionais

Os seguintes requisitos funcionais foram encontrados após uma análise e levantamento de requisitos inicial da aplicação Pizzalandia:

RF01 – O usuário poderá realizar *login* .

RF02 – Os usuários poderão se cadastrar.

RF03 – Os usuários poderão realizar um pedido.

RF04 – Usuário poderá verificar pedidos realizados.

2.1.2 Requisitos não funcionais

Os seguintes requisitos não funcionais foram encontrados após uma análise e levantamento de requisitos inicial da aplicação Pizzalandia:

RNF01 – O sistema deverá ser escalável.

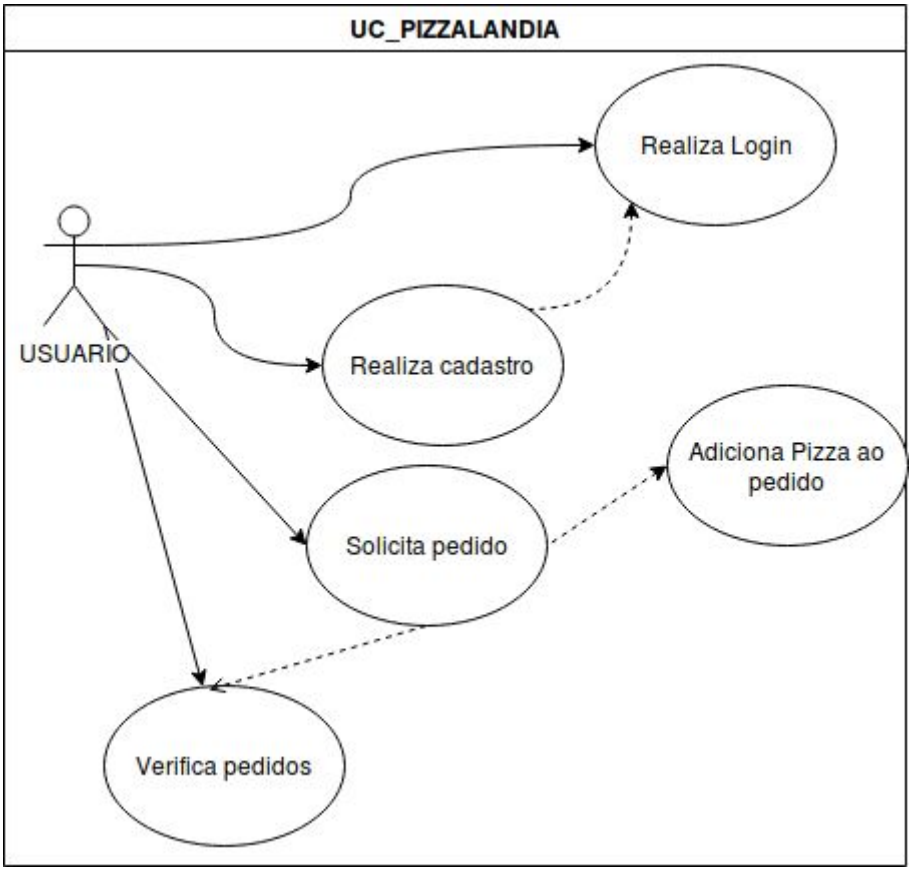
RNF02 – O sistema deverá ser intuitivo.

RNF03 – O sistema deverá disponibilizar uma lista de pizzas.

2.2 DIAGRAMA DE CASOS DE USO

Para a modelagem do Pizzalandia foram utilizados alguns diagramas UML (UML, 2017), com o intuito de possibilitar um entendimento e melhor visualização das atividades envolvidas no uso do aplicativo. Na figura 1, é denotado o Diagrama de Caso de Uso.

Figura 1 – Diagrama de Casos de Uso



Fonte: Elaborado pelos autores (2017).

No caso de uso 'Realiza Login' o usuário se autenticará através do *Firebase* (FIREBASE, 2017).

No caso de uso 'Realizar cadastro' o usuário preencherá as informações básicas de perfil de usuário. Esse caso de uso só será executado se na execução do caso de uso 'Realiza Login' for constatado que o usuário não possui cadastrado no Pizzalandia.

No caso de uso 'Realiza Pedido' o usuário poderá visualizar produtos para poder adicioná-los ao seu pedido.

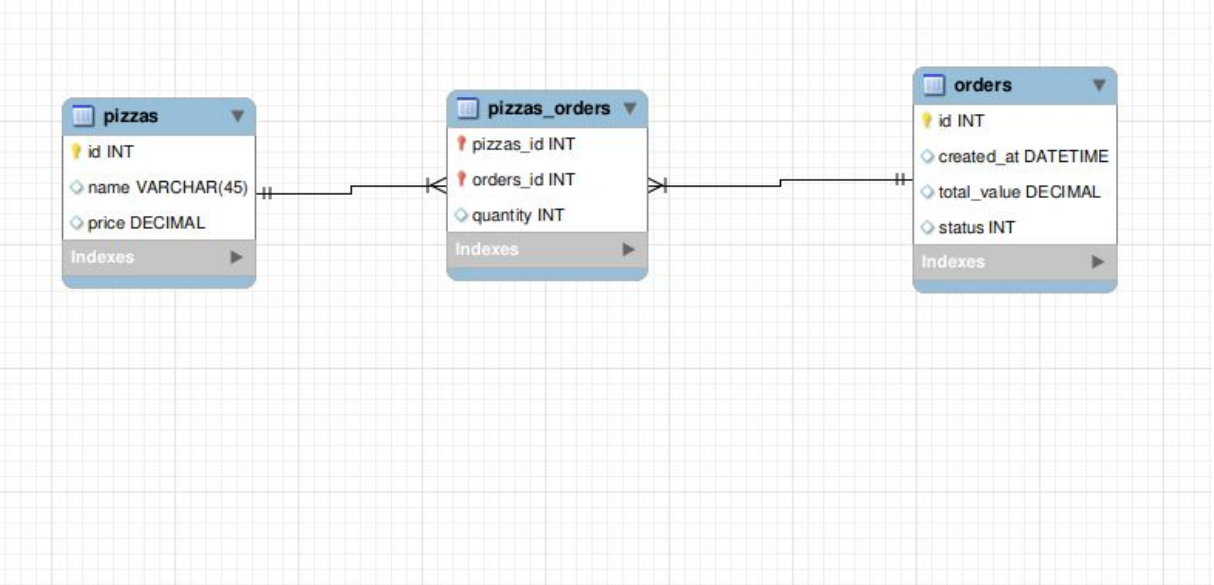
No caso de uso Adicionar pizza ao pedido' o usuário poderá adicionar produtos ao seu pedido.

No caso de uso 'Verifica Pedido'', após o caso de uso 'Realiza Pedido', o usuário terá a opção de visualizar seu pedido.

2.3 DIAGRAMA DE ENTIDADE E RELACIONAMENTO

A figura 3 apresenta o diagrama de entidades e relacionamentos do Pizzalandia, descrito a seguir.

Figura 2 – Diagrama Entidade e Relacionamento (DER)



Fonte: Elaborado pelos autores (2017).

A entidade *pizza*, representa os produtos que o usuário poderá adicionar ao seu pedido, essa entidade e populada através do firebase.

A entidade *order*, representa o pedido do cliente.

A entidade *pizzas_orders*, representa as pizzas que fazem parte do pedido do cliente.

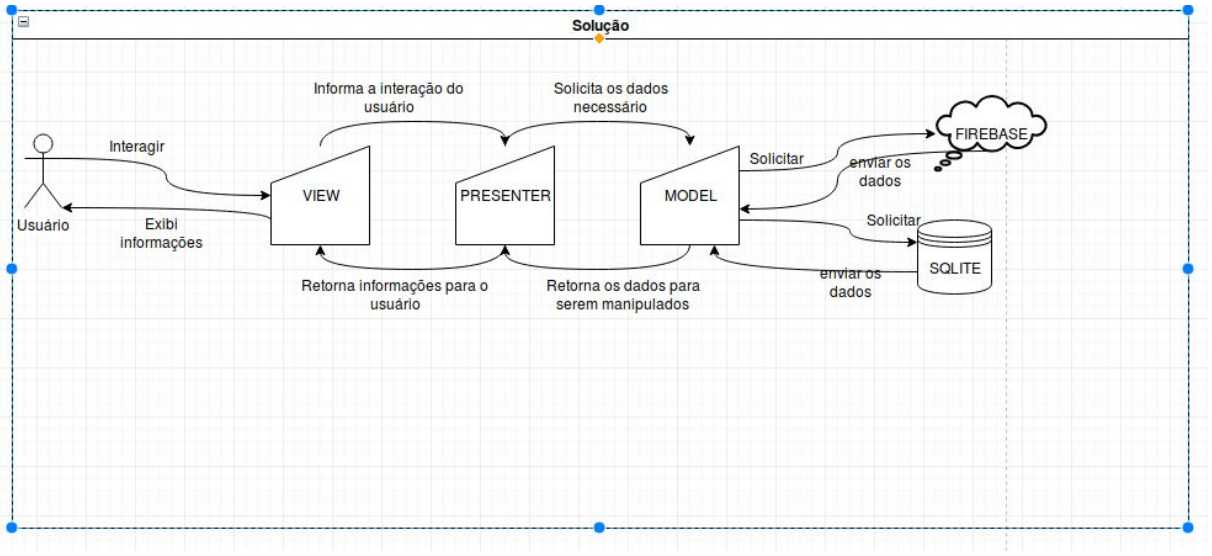
3 IMPLEMENTAÇÃO DO PIZZALANDIA

Na implementação da aplicação *Android* do Pizzalandia foi utilizado a linguagem de programação *Java*, para a *API* 25 ou superior.

3.1 ARQUITETURA

A aplicação foi desenvolvida seguindo a arquitetura cliente-servidor, onde ambos os lados, possuem suas subdivisões de camadas. A Figura 27 mostra a arquitetura geral do sistema que será detalhada a seguir.

Figura 3 – Arquitetura da Solução (Aplicação).



Fonte: Elaborado pelos autores (2017).

O cliente é responsável pela interação com os usuários, apresentação e coleta de informações. O servidor é onde acontece o processamento, armazenamento e as regras de negócios da aplicação. Veja nos tópicos abaixo uma explicação mais detalhada sobre o assunto.

3.2 APLICAÇÃO

Conforme observado na figura 3, aplicação cliente é dividida em três camadas, Modelo, Apresentação e Visualização.

Essa separação facilita nas distribuições e desacoplamento das responsabilidades, possibilitando refatorações menos traumáticas, pois alterações em uma das camadas não afetaria as outras, dado que a comunicação entre as elas é feita através de contratos.

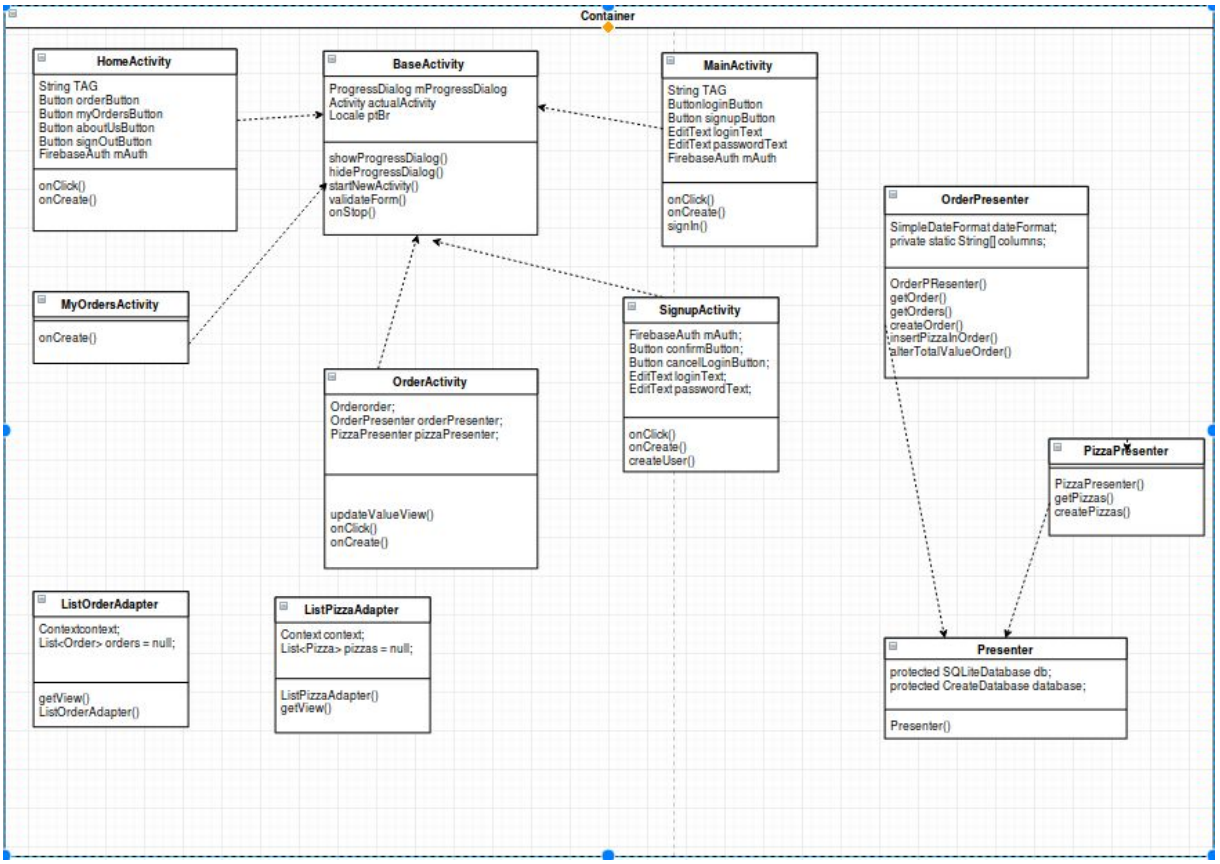
A camada Visualização é responsável por interagir com o usuário, mostrando informações e percebendo as interações que o usuário faz na tela, ou seja, é ela que percebe os cliques dos usuários, e também é ela que exibe as informações na tela. Ao receber interações do usuário a camada Visualização informa para a camada Apresentação o que aconteceu, e através de um contrato, ela a camada Apresentação diz para a camada Visualização o que mostrar na tela, como por exemplo mostrar *loading*, mostrar lista de destinos e etc.

A camada Apresentação irá intermediar a comunicação entre a camada Visualização e a camada Modelo. Ela orquestra o que aparece na tela e quais processos a aplicação deve realizar.

A camada Modelo é responsável pela comunicação com componentes externos a aplicação, como por exemplo: banco de dados loca.

Para acesso ao banco de dados local, no caso o *SqLite* (SQLITE, 2017), é utilizado a *API* nativa do *android* (DEVELOPER ANDROID, 2017) que permite fazer criação de tabelas, atualizações, consultas, inserções e etc.

3.3 CLASSES



3.4 PRINCIPAIS CLASSES

A aplicação possui algumas classes que são essenciais para o seu funcionamento e entendimento das regras de negócio.

3.4.1 SignupActivity

A classe *SignupActivity* responsável pela criação do usuário através da função *createUser*(), onde ela recebe um email e uma senha no formato string, e realiza a comunicação com o *Firebase,* após o cadastro o usuário é redirecionado para a aplicação.

3.4.2 MainActivity

A classe *MainActivity* responsável pelo login do usuário através da função *signIn*() que realiza autenticação do usuário junto ao *Firebase,* após a validação do login é acionado o método de criação de Pizzas, com isso o sistema acessa o

Firebase, busca a entidade Pizzas e realiza a inserção dos seus valores no SQLite do smartphone.

3.4.3 OrderPresenter

A classe *OrderPresenter* é responsável pelo gerenciamento dos pedidos com isso ela recebe um *Context* e realiza a manipulação através das funções, *getOrder()* que busca o último pedido em aberto do usuário, a *getOrders*() busca todos os pedidos do usuário, a *createOrder*() que recebe uma pizza para criação do pedido e após isso chama o método *insertPizzainOrder*() que insere a pizza dentro do pedido, recebendo um pedido, uma pizza e uma quantidade, com isso ela chama a função *alterTotalValueOrder()* que recebe um pedido e o valor da pizza para ser inserido no pedido.

3.4.4 PizzaPresenter

A classe *PizzaPresenter* é responsável pelo gerenciamento das pizzas através dos métodos *getPizzas()* que consulta as pizzas armazenadas no *SQLite*, e através do *createPizzas*(), que busca as pizzas do *Firebase* e insere no *SQLite*

3.4.5 Adapter

A aplicação utiliza dois *Adapters* para exibição dos dados um para os pedidos e outro para as pizzas, ele recebe um *Context* e com isso busca o layout que irá receber os dados e realizar a interação preenchendo um *ListView*.

3.4.6 CreateDatabase

A classe createDatabase é responsável pela criação do banco de dados local, SQLite, e possui os parâmetros para geração das tabelas e controle de versão

4 CONCLUSÃO

Através de pesquisas realizadas, foi identificado que a tecnologia pode auxiliar desde grandes empresas até aquelas familiares e pequenas, que são o público alvo do Pizzalandia.

Atualmente realizar um pedido de delivery é simples, porém acompanhar o pedido que é o grande enfoque do Pizzalandia, pois auxilia o consumidor a listar as pizzarias mais próximas a sua região, acompanhar horário de funcionamento das mesmas, cardápio, realizar pedido e acompanhar status da entrega.

REFERÊNCIAS

DEVELOPER ANDROID. **Sqlitedatabase**. Disponível em: <https://developer.android.com/reference/android/database/sqlite/sqlitedatabase.html>. Acesso em: 3 dez. 2017.

SQLITE. **Sqlite**. Disponível em: <https://www.sqlite.org/>. Acesso em: 3 dez. 2017.

TECHTUDO. **Afinal, o que é android?**. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2011/01/afinal-o-que-e-android.html>. Acesso em: 3 dez. 2017.

FIREBASE. **Firebase.** Disponível em: <https://firebase.google.com/?hl=pt-br> acesso em: 4 dez 2017.

UML. **Uml.** Disponível em: <http://www.uml.org/> acesso em: 5 dez 2017.