UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RODRIGO FELIPE SONNTAG WIEBBELLING

Aplicativo para geração de pedidos em sistema centralizado de padarias e restaurantes

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Leandro Krug Wives

Porto Alegre
2023

RESUMO

Este trabalho tem como objetivo desenvolver um sistema de geração de pedidos digitais através de um aplicativo Android de fácil utilização integrado com o sistema de ERP e PDV da empresa KW Informática, com o objetivo futuro de tornar este aplicativo mais um produto do portfólio da empresa. O aplicativo proposto tem como objetivo permitir que os atendentes de diferentes estabelecimentos comerciais com atendimento ao público em gerar possam efetuar o registro de pedidos de forma digital, tendo os mesmos registrados diretamente na base de dados do estabelecimento para fácil acesso e manutenção dos mesmo ao longo do atendimento de uma mesa ou cliente específicos. Além disso, o sistema se comunica com toda a infraestrutura hoje presente nos diferentes clientes da empresa, permitido que os mesmos possam, através de configurações, ter os pedidos efetuados através do app impressos em diferentes pontos do estabelecimento com seus devidos produtos distribuídos de acordo com sua origem. Para isso, foram escolhidos a linguagem nativa Java Android para permitir o acesso a funcionalidades de mais baixo nível dentro do aplicativo, bem como o banco de dados MySQL para poder manter a compatibilidade com a infraestrutura já existe hoje nos clientes.

Palavras-chave: Pedidos Digitais. Android. Java. MySQL. Mobile. POS.

Mobile aplication for generating digital orders on a centrilized sistem for restaurants and bakaries

ABSTRACT

This work aims to develop a digital order generation system through an easy-to-use Android application integrated with the ERP and POS system of the company KW Informática, with the future objective of making this application another product of the company's portfolio. The proposed application aims to allow the attendants of different commercial establishments with public service to register orders digitally, having them registered directly in the establishment's database for easy access and maintenance of the same throughout serving a specific table or customer. In addition, the system communicates with the entire infrastructure currently present in the different customers of the company, allowing them to be able, through configurations, to have orders placed through the app printed at different points of the establishment with their appropriate products distributed according to your origin. For this, the native Java Android language was chosen to allow access to lower-level functionalities within the application, as well as the MySQL database to be able to maintain compatibility with the infrastructure that already exists today in customers.

Keywords: Digital Order. Android. Java. MySQL. Mobile. POS.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

POS	Pont of Sale
ERP	Enterprise Resource Planning
PDV	Ponto de Venda
IDE	Integrated Development Environment
API	Application Programming Interface
XML	Extensible Markup Language
CUPS	Common Unix Printing Sistem
UFRGS	Universidade Federal do Rio Grande do Sul
SQL	Structured Query Language
EAN	European Article Number
RFID	Radio Frequency Identification
SUS	System Usability Score

SUMÁRIO

1 INTRODUÇÃO

Com a evolução da tecnologia, cada dia que passa, mais e mais ferramentas para facilitar o dia-a-dia de profissionais em diferentes áreas de atuação vem sendo criadas. Tais ferramentas tem como objetivo simplificar o trabalho e maximizar a produtividade dos profissionais, visando um melhor desenvolvimento de suas atividades com menos perda de tempo e recursos desnecessários.

Quando olhamos para o cenário de restaurantes e padarias, muitas vezes, ainda vemos seus atendentes utilizando caneta e papel para cumprir uma atividade crucial dentro destes estabelecimentos, gerar pedidos. Esta atividade, aparentemente simples, pode ser muito mais complexa do que parece, quando analisamos diferentes cenários e necessidades de cada um destes estabelecimentos.

Um pedido gerado dentro destes estanciamentos gera uma série de atividades que, ao serem executadas de maneira correta, resultam em um cliente satisfeito e uma cobrança correta ao seu final. Só que, muitas vezes, não é isso o que acontece.

Ao anotar um pedido em um pedaço de papel, o mesmo precisa ser preservado durante toda a estadia do cliente, para que ao final, o total da sua conta possa ser computado e o pagamento possa ser efetuado de maneira correta e justa. Infelizmente, acidentes acontecem, e com restaurantes e padarias, não é diferente. Um café que derrama em cima da comanda, ou então o vento que sopra a mesma para debaixo de uma bancada sem que ninguém veja são alguns dos exemplos mais comuns que tornam o dia-a-dia de um atendente muito mais difícil.

Visando facilitar este tipo de atendimento, este trabalho visa fornecer uma solução utilizando um Aplicativo Android Nativo que permite ao atendente não apenas gerar um simples pedido, mas ter uma séria de funcionalidades adicionais que o ajudam a ser mais eficiente e assertivo no seu trabalho.

O ambiente proposto consiste de uma infraestrutura já existente composta por um servidor Apache juntamente com um sistema web para configuração dos diferentes sistemas existentes dentro do estabelecimento. Além disso, também temos um servidor desenvolvido em C++ que consiste em fornecer algumas funcionalidade utilizadas pelo aplicativo para que o mesmo consiga acessar a base de dados MySQL existente no cliente através de uma comunicação Soket com protocolo proprietário da empresa.

Por ultimo, temos o aplicativo desenvolvido em Java Android nativo, permitindo assim que funcionalidades de mais baixo nível possam ser utilizadas para permitir uma

compatibilidade entre o aplicativo e o ambiente de pedidos digital já existente.

Este trabalho foi dividido em 6 capítulos, onde o capítulo 2 descreve alguns conceitos e tecnologias utilizadas. O capítulo 3 descreve a estrutura do banco de dados e a arquitetura utilizada, o quarto descreve as funcionalidades do sistema, já o quinto capítulo apresenta as interfaces do sistema, e o capítulo 6 apresenta as conclusões e limitações do sistema, bem como planejamentos futuros do desenvolvimento.

2 CONCEITOS E TECNOLOGIAS

2.1 Conceitos

Esta sessão tem como objetivo apresentar alguns conceitos utilizados durante a apresentação deste trabalho, visando um melhor entendimento do leitor.

2.1.1 Sistema ERP

ERP significa *Enterprise Resource Planning*, traduzindo do inglês, “Planejamento dos Recursos da Empresa”. Tem como objetivo gerenciar e automatizar a gestão empresarial, permitido ao usuário inserir dados manualmente ou de forma automática, fornecendo uma visão única e eficiente dos diferentes processos de uma empresa, tornado as tomadas de decisão muito mais assertivas e eficientes na gestão da empresa. (MARINS, 2005)

2.1.2 POS e PDV

POS - *Point of Sale*, traduzido do inglês, "Ponto de venda", ou PDV, tem como objetivo automatizar a cobrança de produtos e servidos de uma empresa ou estabelecimento. Sua principal função é contabilizar e totalizar os valores de produtos e serviços fornecidos, bem como efetuar a cobrança dos mesmos, fornecendo um cupom fiscal para o cliente ao final do processo.

2.1.3 Infraestrutura

Infraestrutura é o conjunto de equipamentos e sistemas pertencentes a um mesmo estabelecimento, responsáveis pelo funcionamento do mesmo.

2.1.4 IDE

IDE - *Integrated Development Environment*, traduzido do inglês, Ambiente de Desenvolvimento Integrado, tem como objetivo fornecer um ambiente com todas as fer-

ramentas necessárias para o desenvolvimento em certas plataformas. (LIMA, 2022)

2.2 Tecnologias

O sistema apresentado neste trabalho consiste de um Aplicativo Android integrado com um sistema já existente, sendo assim algumas tecnologias foram utilizadas para o desenvolvimento do mesmo.

2.2.1 Android Studio

O Android Studio consiste de uma IDE para arquitetura ARM utilizada dentro dos dispositivos Android disponíveis no mercado atualmente. Baseado no software IntelliJ IDEA da JetBrains, o Android Studio é distribuído de forma gratuita sob a Licença Apache 2.0. Estando disponível para Download em Windows, Mac OS X e Linux. (GOOGLE, 2023b)

2.2.2 MySQL

MySQL é um sistema de gerenciamento de banco de dados gratuito mais utilizado no mundo, tendo como objetivo fornecer um sistema robusto de gerenciamento de banco de dados relacionais compatível com a linguagem SQL - do inglês, *Structured Query Language*, traduzido para o português, "Linguagem de Consulta Estruturada". O uso do MySQL, juntamente com a linguagem SQL permite que a aplicação grave, manipule e acesse toda e qualquer informação necessária durante a sua execução de forma eficiente e rápida. (GROFF; WEINBERG; OPPEL, 2002)

2.2.3 API

API significa *Application Programming Interface*, traduzido do inglês "Interface de Programação de Aplicação". Ela serve como uma espécie de contrato entre a aplicação e o servidor, permitindo assim que a comunicação entre o aplicativo e outros sistemas seja mais segura e rápida, evitando assim possíveis brechas de segurança durante a execução

da comunicação. (DE, 2017)

2.2.4 CUPS

CUPS (*Common Unix Printing Sistem*), do inglês, "Sistema Comum de Impressão Unix"é um servidor de impressão utilizado para gerenciar diferentes impressoras dentro de uma rede *Ethernet*, permitindo que diferentes padrões de nomes de arquivos sejam impressos em diferentes impressoras, permitindo assim que uma única pasta de impressão seja utilizada por diferentes sistemas para enviar arquivos para a impressão.(INC, 2022)

2.2.5 XML

XML ou *Extensible Markup Language*, traduzido do inglês "Linguagem de Marcação Extensível"uma linguagem de marcação com regras para formatar documentos de forma que eles sejam facilmente lidos tanto por humanos quanto por máquinas. (ALMEIDA, 2002)

3 DESENVOLVIMENTO

3.1 Processo de Desenvolvimento

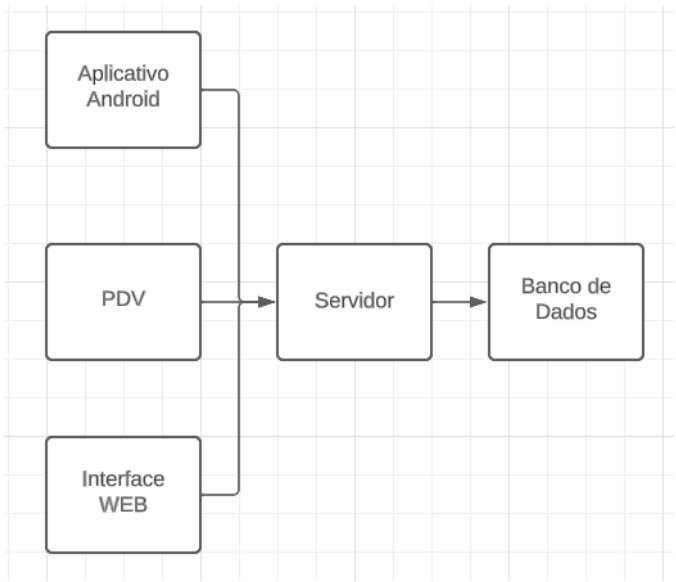
Atualmente, a empresa onde eu trabalho possui uma estrutura onde os clientes são atendidos por filiais ao redor do Brasil, sendo São Paulo, a maior filial existente hoje. Por serem parceiros muito antigos, tenho uma ótima relação com as pessoas que atendem os clientes finais de São Paulo, sendo assim, aproveitei tal relação para trocar diversas ideias e *feedbacks* com eles ao longo do desenvolvimento do projeto.

Para um controle de atividades, funcionalidades e bugs a serem resolvidos dentro do aplicativo, passei a utilizar a ferramenta Asana, que tem por conceito, um quadro Kamban, que também pode ser visualizado em forma de lista de atividades a serem feitas.

A filial de São Paulo acabou por atuar de certa forma, como um *Produc Owner*, utilizando do Asana compartilhado para me passar os *feedbacks* necessários, uma vez que eles possuem uma relação muito próxima ao cliente final, fazendo com que o desenvolvimento como um todo, se aproximasse muito de uma metodologia ágil do tipo SCRUM.

3.2 Arquitetura do Sistema

Figura 3.1 – Diagrama de Arquitetura



Fonte: O Autor

Para a escolha da arquitetura utilizada, tive grande influência do que já se encontrada disponível nos possíveis futuros clientes (Figura 3.1), visando uma futura implantação mais simples para facilitar a vida do cliente. Sendo assim, a arquitetura utilizada foi a Cliente-Servidor, onde o aplicativo coleta e apresenta as informações para o usuário e o servidor fornece toda a parte de *Backend* para armazenar os dados e processar informações solicitadas pelo aplicativo cliente.

Neste tipo de arquitetura, temos uma instância do servidor rodando em cada loja, onde zero ou mais instancias do aplicativo podem ser executadas ao mesmo tempo, sendo todas elas atendidas pelo mesmo servidor. Tal escolha apresentou alguns desafios ao longo do caminho para evitar que informações conflitantes acabacem por corromper os dados dos pedidos ao longo do dia. Um exemplo deste problema era a aquisição do numero atual do pedido, onde, toda vez que uma instância do app iniciasse um novo pedido, ela deveria atualizar o numero do mesmo na tabela de pedidos para que outro app não tentasse efetuar um pedido utilizando do mesmo numero.

O uso de uma arquitetura cliente-servir nos trás alguns benefícios, como, uma vez que as principais funcionalidades do sistema se encontram implementadas no servidor, uma simples atualização não exige que todos os dispositivos mobiles sejam atualizados, mas sim, apenas o servidor, tornando ajustes e correções de *bugs* muito mais simples de serem implantadas no cliente.

3.2.1 Funcionalidades do Servidor

O servidor fornece ao aplicativo as seguintes funcionalidade durante sua operação.

1. Acesso ao banco de dados
    1. Socket para execução de *querys SQL* diretamente nas tabelas do banco de dados relacional
    2. API de geração de pedidos via XML, permitindo assim que o servidor possa controlar a inserção correta dos mesmos quando mais de uma instância do aplicativo estiver rodando na mesma loja.
3. Impressão de arquivos
    1. *Upload* de arquivos para uma pasta de impressão
    2. Renomeação de arquivos dentro da pasta de impressão

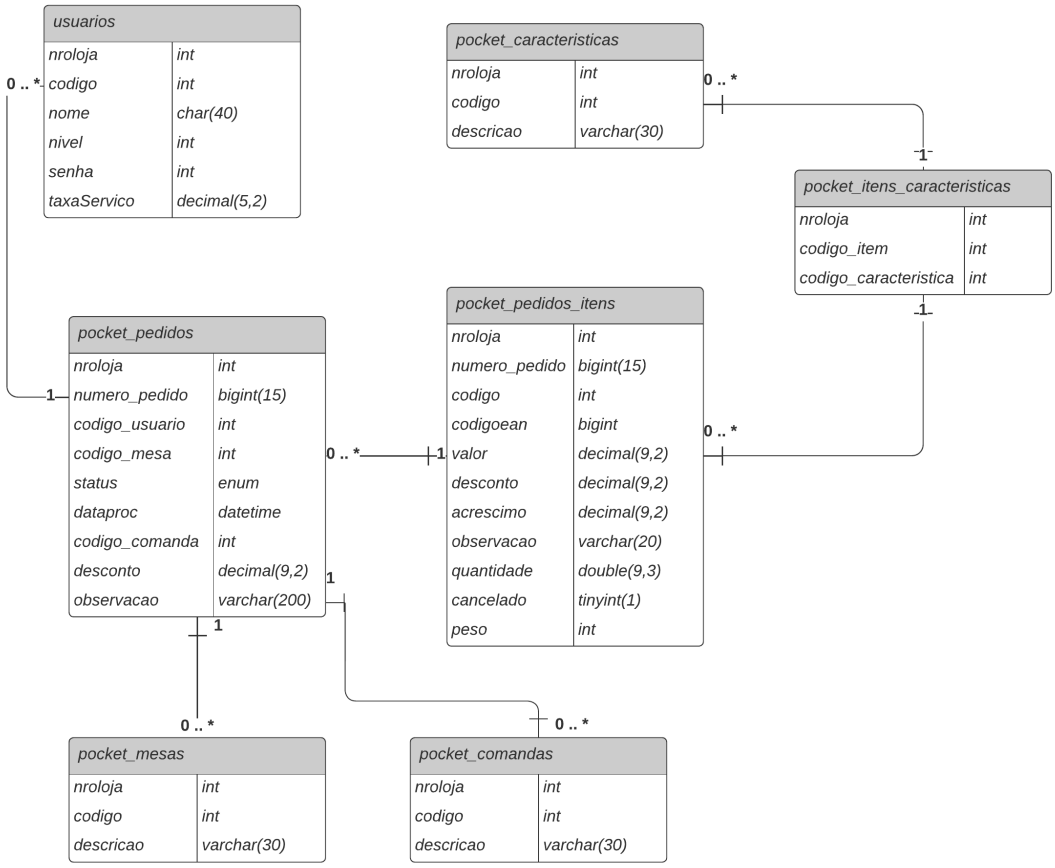
3.3 Banco de Dados

Nesta sessão, apresenta-se a estrutura de banco de dados utilizada para a criação do aplicativo de pedidos, bem como a estrutura de dados já existente que foi utilizada para a integração do aplicativo com o resto do sistema.

3.3.1 Diagrama ER de Pedidos

O Diagrama de entidade-relacionamento representativo utilizado para armazenar todas as informações de um pedido dentro do sistema integrado da empresa. (Figura 3.2)

Figura 3.2 – Diagrama de Banco de Dados para Pedidos

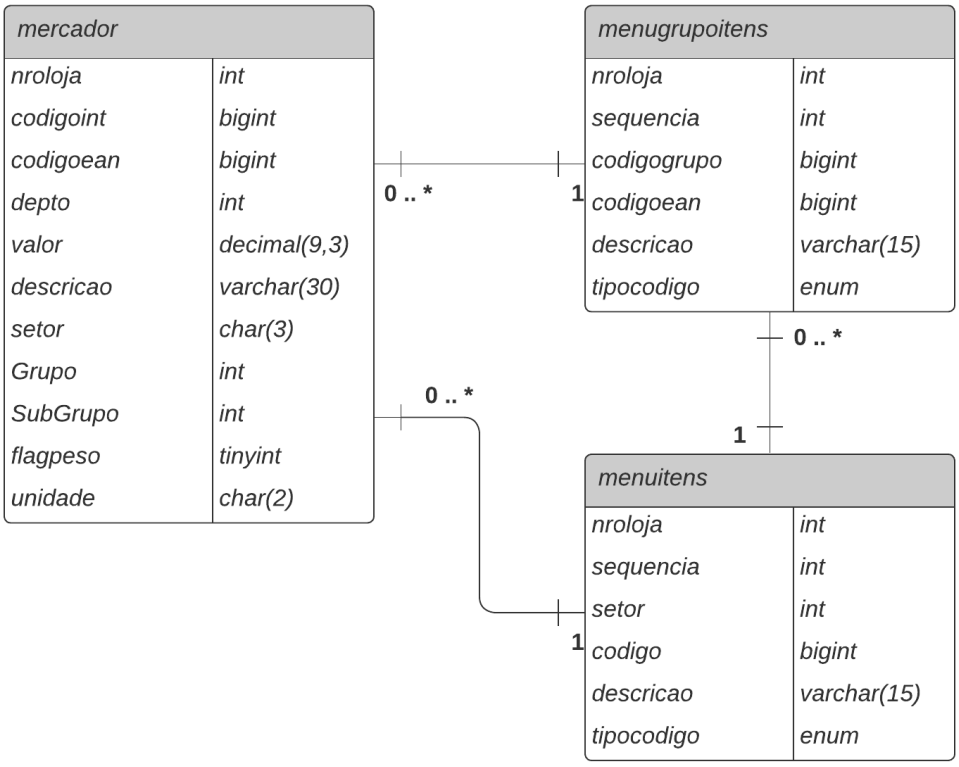


Fonte: O Autor

3.3.2 Diagrama ER de Mercadorias

O Diagrama de entidade-relacionamento representativo utilizado para armazenar todas as informações referentes ao menu principal de venda de mercadorias. (Figura 3.3)

Figura 3.3 – Diagrama de Banco de Dados do Menu Principal



Fonte: O Autor

Para uma melhor visualização, o diagrama do banco de dados foi dividido em 2, para entender a relação entre os 2 diagramas, basta observarmos que o campo codigoean da tabela pocket_pedidos_itens é uma chave estrangeira em relação ao campo codigoean na tabela mercador.

3.3.3 Usuários

A entidade usuários representa os usuários do sistema interligado, esta tabela já era existente no sistema, e, uma vez que procuramos integram o aplicativo com o sistema

já existente, a mesma tabela foi utilizada para armazenar os usuários do aplicativo de pedidos.

- **nroloja**: Identifica o número à qual loja aquele usuário pertence, uma vez que um mesmo servidor pode ser utilizado em mais de 1 loja no modo "multiloja".
- **codigo**: Identificador único utilizado para referenciar o usuário em qualquer contexto do sistema, sendo este único para cada usuário distinto.
- **nome**: Nome do usuário cadastrado.
- **nivel**: Este campo identifica o nível de acesso de cada usuário, utilizado para limitar as funcionalidades do sistema de acordo com cada nível existente.
- **senha**: Armazena a senha criptografada do usuário que é utilizada para acessar os diferentes pontos do sistema.
- **taxaServiço**: Valor de taxa que o usuário recebe por cada venda, utilizado para totalizar os valores de cada pedido.

3.3.4 pocket_pedidos

Esta entidade armazena os pedidos efetuados pelo aplicativo e suas características principais, dados secundários como os itens do pedido serão armazenados na tabela pocket_pedidos_itens utilizando nroloja e numero_pedido como chave estrangeira desta tabela.

- **nroloja**: Identifica o número à qual loja aquele pedido pertence, uma vez que um mesmo servidor pode ser utilizado em mais de 1 loja no modo "multiloja".
- **numero_pedido**: Código único de identificação do pedido que será utilizado para saber o que o cliente consumiu no estabelecimento quando o mesmo passar pelo PDV para efetuar o pagamento do seu consumo.
- **codigo_usuario**: É o código do usuário que efetuou o pedido.
- **codigo_mesa**: É o código da mesa onde o cliente se encontra.
- **status**: É o status do pedido, que pode estar "Em aberto", "Fechado", "Pago", "Em Fechamento"ou "Cancelado"
- **dataproc**: Armazenam a data e a hora que o pedido foi criado
- **codigo_comanda**: Armazena o código da comanda quando temos mais de 1 cliente sentado na mesma mesa com comandas distintas.

- **desconto**: É o valor de desconto aplicado ao pedido, que pode ocorrer por diferentes motivos.
- **observacao**: Este campo pode armazenar alguma observação do pedido, que será impressa juntamente com os itens a serem preparados. Por exemplo, "O é para levar.".

3.3.5 pocket_mesas

Nesta entidade, são armazenados os cadastros de todas as mesas da loja ou lojas, para futuramente o aplicativo poder mostrar um mapa de mesas ao usuário.

- **nroloja**: Identifica o número da loja a qual aquela mesa pertence.
- **codigo**: É o código único da mesa.
- **descricao**: É o nome dado para cada mesa cadastrada no sistema.

3.3.6 pocket_comandas

Nesta entidade, são armazenados os cadastros de todas as comandas da loja ou lojas, para futuramente o aplicativo poder mostrar um mapa de comandas ao usuário.

- **nroloja**: Identifica o número da loja a qual aquela comanda pertence.
- **codigo**: É o código único da comanda.
- **descricao**: É o nome dado para cada comanda cadastrada no sistema.

3.3.7 pocket_caracteristicas

Nesta entidade, são armazenados todas as características de diferentes itens, que depois serão associadas a cada item através da tabela pocket_itens_caracteristicas, permitindo assim que características em comum, como "Sem gelo", possam ser utilizadas por vários itens ao mesmo tempo.

- **nroloja**: Identifica o número da loja a qual aquela característica pertence.
- **codigo**: É o código único da característica.
- **descricao**: É o nome dado para cada característica cadastrada no sistema.

3.3.8 pocket_itens_caracteristicas

Esta entidade armazena a relação itens-características, indicado quais características pertencem a cada item.

- **nroloja**: Identifica o número da loja a qual a relação item-característica pertence.
- **codigo_item**: É o código único do item.
- **codigo_caracteristica**: É o código único da característica.

3.3.9 pocket_pedidos_itens

Aqui, nesta entidade, temos o armazenamento de todos os itens de um pedido, mesmo os cancelados, indicados pelo campo "cancelado", permitindo assim que todo o pedido seja carregado para edição via aplicativo ou então para pagamento no PDV da loja.

- **nroloja**: Identifica o número da loja a qual aquele item pertence.
- **numero_pedido**: É o número do pedido que aquele item pertence
- **codigo**: Representa o código único daquele item.
- **codigoean**: É o código EAN do item.
- **valor**: Armazena o valor do item vendido.
- **desconto**: É o desconto a ser aplicado no valor.
- **acrescimo**: É o valor a ser acrescido no valor.
- **observação**: É uma observação especifica do item, ex: "Sem gelo".
- **quatidade**: É a quantidade vendida
- **cancelado**: Indica se o item foi cancelado, para termos um rastreio de todos os itens lançados em um pedido.
- **peso**: Indica se a quantidade do item é fracionária.

4 FUNCIONALIDADES DO SISTEMA

Neste capítulo irei apresentar as funcionalidades dos diferentes componentes do sistema. Sendo elas as novas funcionalidade do servidor e o cliente móvel por completo.

4.1 Servidor

Como já comentado anteriormente, o servidor já era um sistema existente ao inicio do projeto, porém, para que o mesmo pudesse ser utilizado para o desenvolvimento do aplicativo cliente, algumas funcionalidades tiveram de ser implementadas e adicionadas ao servidor para que ele pudesse atender todas as necessidades do aplicativo.

Alguns do requisitos funcionais de um servidor, como escalabilidade, segurança e disponibilidade já existiam uma vez que o servidor não foi implementado do zero, e sim, apenas adaptado para o projeto. Como o foco do projeto não esta no servidor, mas sim na aplicação mobile, tais requisitos foram reaproveitados do sistema já em funcionamento, poupando tempo de desenvolvimento do lado do servidor.

4.1.1 Protocolo de comunicação

Antes de implementar qualquer funcionalidade no sistema, foi necessário a criação de um protocolo de comunicação, para permitir que aplicativo e servidor pudessem trocar mensagens. Tal protocolo foi definido como PAAA...*; onde P representa a funcionalidade a ser utilizada e AAA...* é um texto alfa-numérico de tamanho livre e ; representa o fim da mensagem. Várias estruturas AAA...*; podem ser concatenadas após o carácter P, permitindo assim que mais de uma mensagem seja enviada ao servidor ao mesmo tempo para uma mesma funcionalidade.

4.1.2 Comunicação Socket

A primeira funcionalidade a ser implementada no servidor foi uma comunicação do tipo socket tcp-ip. Esta comunicação é a base para a troca de mensagem entre o aplicativo e o servidor durante todo o processo de funcionamento do aplicativo cliente. Juntamente com o protocolo descrito acima, a comunicação socket é o que viabilizada o

funcionamento do aplicativo dentro da arquitetura já existente, uma vez que implementar uma REST API não era possível na estrutura atual por questões de segurança.

4.1.3 Acesso ao banco de dados

Utilizando da comunicação socket e seu protocolo, o acesso ao banco de dados para a busca de informações podia ser feito seguindo o protocolo definido, ex: Sretag;SELECT * FROM usuarios; onde S representa a funcionalidade "Executar query SQL", retag representa o nome do banco a ser acessado e "SELECT * FROM usuarios"representa a consulta SQL a ser executada no banco selecionado. Uma vez executada, o resultado da consulta era retornado ao aplicativo seguindo o mesmo protocolo, onde cada campo da consulta era concatenado na mensagem com ; entre eles para efetuar a separação das informações.

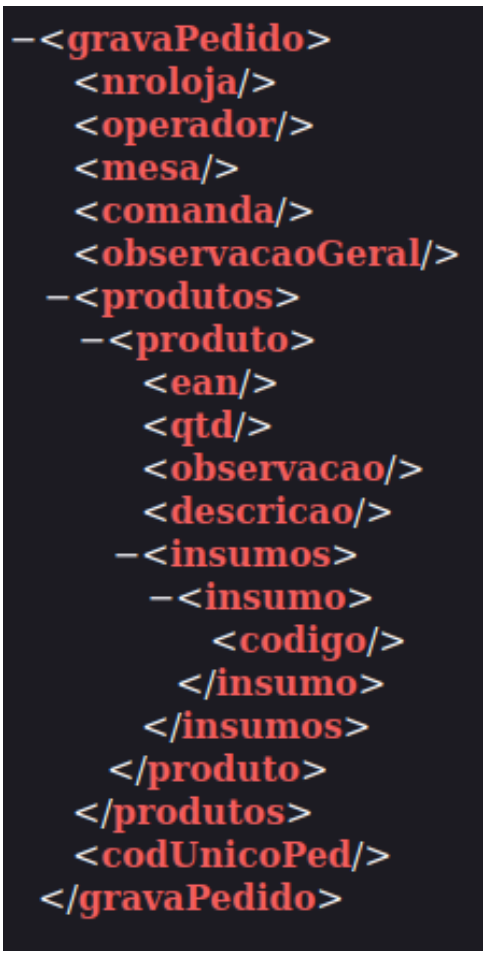
4.1.4 Gravação/Edição de Pedidos

Com a evolução do aplicativo, o protocolo começou a se tornar um pouco fraco para o envio de algumas mensagens, uma delas sendo a gravação de pedidos, que foi ficando cada vez mais complexa ao longo do desenvolvimento. Sendo assim, foi decido por se manter o protocolo inicialmente proposto, mas, dentro da mensagem para a gravação de pedidos, passamos a enviar um texto XML, permitindo assim que estruturas mais complexas fossem enviadas para o servidor na hora de gravar um pedido no banco de dados.

Para a gravação de um novo pedido, o seguinte protocolo foi implementado, #001; XML; onde # representa a funcionalidade gravação/edição de pedido, "001"representa que queremos gravar um novo pedido, e "XML"representa o texto em formato XML enviado ao servidor (Figura 4.1)

Para a edição de pedidos, o componente "001"da mensagem é trocado por "002", indicando assim ao servidor que o XML contem um pedido já existente, e que o mesmo deve apenas ser atualizado de acordo com as informações existentes no XML enviado.

Figura 4.1 – XML de geração de pedidos.



Fonte: O Autor

4.1.5 Impressão de arquivos

Como explicado anteriormente, a arquitetura possui um servidor CUPS, utilizado para o gerenciamento de impressoras em rede. Este servidor foi configurado para ficar "olhando"para uma pasta dentro do servidor da empresa, onde, toda vez que um arquivo, possuindo algum padrão específico de nome seja criado, o mesmo deve ser pego e impresso em alguma impressora configurada para aquele padrão.

EX: Temos 2 impressoras no restaurante, uma na cozinha e outra na copa. No servidor CUPS podemos configurar para que, todos os arquivos que comecem com "cozinha", sejam impressos na impressora A (Localizada na cozinha) e, todos os arquivos que comecem com o nome "bebidas"sejam impressos na impressora B (Localizada na copa).

Para utilizar de tal funcionalidade, foi implementado no protocolo, uma chamada para geração de arquivos no servidor, onde o aplicativo envia a seguinte mensagem: Far-

quivo.txt;dadosArquivo; onde F indica a funcionalidade de criação de arquivo, arquivo.txt indica o nome com o qual o arquivo deve ser criado e dadosArquivo indica o que deve ser gravado dentro do arquivo. Desta maneira, o aplicativo consegue separar os itens de acordo com sua classificação e gerar diferentes arquivos de impressão para cada ambiente do restaurante.

4.1.6 Renomeação de arquivos

Com a evolução do desenvolvimento, um problema que encontramos na hora de criar arquivos foi que, o tempo necessário para gravar o arquivo por completo no servidor era maior que o tempo necessário para o servidor CUPS detectar o arquivo e imprimir o mesmo, levando assim à impressão de arquivos pela metade. Para mitigar este problema, criamos a funcionalidade de renomeação de arquivo, permitindo assim com que o aplicativo primeiro criasse o arquivo com um nome temporário, que não seria consumido pelo servidor de impressão, e, após ter o arquivo gravado por completo, o aplicativo poderia então apenas renomear o arquivo já existente, evitando assim que arquivos pela metade fossem consumidos.

Para isso foi implementado a seguinte mensagem: R;arquivoTemporario.txt;arquivoFinal.txt; onde R indica que é uma renomeação de arquivo, arquivoTemporario.txt é o nome do arquivo já existente no servidor e arquivoFinal.txt é nome que queremos dar para o arquivo novo.

4.2 Aplicativo Mobile - Decisões do Desenvolvimento

Nesta seção vamos falar sobre o processo de criação do aplicativo. Falar de algumas tecnologias e o motivo da escolha de cada uma delas.

4.2.1 Nativo x Multiplataforma

A primeira escolha que devemos fazer é se o aplicativo a ser desenvolvido será feito em linguagem nativa de cada sistema operacional ou se usaremos um *f*ramework multiplataforma para desenvolver o mesmo.

Ao escolhermos o uso de um *f*ramework multiplataforma, o *f*ramwork utilizado

irá mapear os recursos do sistema para funcionalidades genéricas, fazendo com que o aplicativo fique limitado em relação a certas funcionalidades, uma vez que ambos os sistemas operacionais precisam possuir tal funcionalidade para que ela possa ser mapeada. No caso de funcionalidades únicas de um sistema ou outro, APIs nativas ainda são necessárias para que o recurso escolhido possa estar disponível na plataforma específica.

Ao escolhermos o uso de uma linguagem nativa, seja ela IOs ou Android, temos o acesso completo para à todas as funcionalidades que aquela plataforma pode nos oferecer, porém, se quisermos que nosso aplicativo exista nas duas plataformas, teremos o trabalho dobrado de desenvolvimento, uma vez que precisaremos fazer efetivamente 2 aplicativos, um para cada plataforma. Isso acaba tornado a manutenção dos mesmos mais complexa e muitas vezes custosa ao longo do tempo.

Neste projeto estaremos escolhendo o uso do desenvolvimento nativo, uma vez que funcionalidades de mais baixo nível serão necessárias para o desenvolvimento da comunicação entre cliente e servidor, funcionalidades essas que muitas vezes não se encontram disponíveis dentro dos *f*ramworks de desenvolvimento multiplataformas.

4.2.2 Android x IOS

Uma vez decido que o desenvolvimento será nativo, precisamos agora escolher a plataforma a ser utilizada. Aqui cabe ressaltar que a escolha de uma das plataformas está muito vinculada ao seu público alvo, uma vez que dispositivos IOS são normalmente encontrados em um público alvo com maior poder aquisitivo, já os dispositivos Android podem ser encontrados em todas as classes sociais.

Outro fator importante é considerar se o cliente irá adquirir novos dispositivos para rodar o sistema, neste caso, se olharmos para o mercado de celulares e tablets do Brasil, iremos perceber que a grande maioria de dispositivos com valores mais acessíveis serão Android, tornando assim muito mais provável o uso destes dispositivos em um ambiente de trabalho.

Por este motivo que a escolha para o desenvolvimento do aplicativo foi o Android, garantindo assim que mais clientes consigam utilizar o aplicativo com um investimento muito mais plausível em seus estabelecimentos.

Outro fator que ajudou na hora de escolher a plataforma em que o aplicativo seria desenvolvido foi a documentação para desenvolvimento de cada uma das plataformas, onde, do meu ponto de vista, a Google se mostrou muito mais completa e de fácil enten-

dimento para um desenvolvedor de primeira viagem. (GOOGLE, 2023a)

4.2.3 Java x Kotlin

Ao escolhermos a plataforma Android, temos duas opções de linguagens para o desenvolvimento do aplicativo, para o nosso projeto, vamos utilizar a linguagem Java por ser uma linguagem mais conhecida e que não exige nenhuma curva de aprendizado para o desenvolvimento, uma vez que já tenho domínio da mesma.

Outro fator importante para escolhermos a linguagem Java é que, por estar a mais tempo no mercado, temos mais informações disponíveis sobre ela na Internet, facilitando assim a busca por soluções quando necessário.

4.3 Aplicativo Mobile - Funcionalidades

Nesta seção, serão apresentadas as principais funcionalidades que compõem o aplicativo, funcionalidades estas que foram mapeadas em conjunto com os usuários finais, visando entender as suas necessidades e encontrar as melhores soluções para cada um dos problemas apresentados por eles.

4.3.1 Criação de um Pedido

A principal função e motivação para a criação do Aplicativo é a criação de pedidos de forma digital de maneira integrada com o sistema de PDV do cliente. No aplicativo, o usuário é capaz de criar pedidos complexos com diferentes itens, tendo uma visualização do tipo "Carrinho de Compras", que torna a visualização do pedido como um todo simples e intuitiva.

4.3.1.1 Adicionando um item ao pedido

Primeiramente, para criar um pedido, é necessário que itens sejam adicionados ao pedido. Para isso, foi criado um menu de acesso rápido totalmente customizável pelo cliente. Este menu permite que o usuário acesse vários itens de maneira rápida e fácil, tornando o uso do aplicativo algo vantajoso em relação a ter que anotar o pedido em um

pedaço de papel.

Outra maneira que o usuário pode adicionar itens ao pedido é através da busca por descrição ou código do produto, para aqueles itens de menos fluxo, ou até mesmo para um item que não foi possível de ser encontrado dentro do menu de acesso rápido.

4.3.1.2 Deletando um item do pedido

Ao criarmos pedidos, muitas vezes o usuário pode cometer um erro ao registrar um item, sendo necessário assim que ele possa deletar este item do pedido. Para isso, basta que o usuário arraste o item que se encontra na listagem para a direita, tendo um feedback em vermelho, indicando que o mesmo irá deletar o item do pedido.

4.3.1.3 Editando um item do pedido

Outra funcionalidade muito necessária no aplicativo é a edição de um item, pois muita vezes temos mais de uma pessoa pedindo a mesma coisa, sendo assim, o usuário pode editar o pedido, alterando a quantidade desejada de cada item, sem a necessidade de adicionar várias vezes o mesmo item ao pedido, para isso, ao contrário de deletar o item, basta arrastar o mesmo para a esquerda, com um feedback azul indicando que o item será editado.

4.3.2 Edição de um pedido

Uma vez feito, um pedido muitas vezes precisa ser alterado, seja por erro do usuário, seja por problemas da cozinha, ou até mesmo, pelo cliente que deseja adicionar ou cancelar algum item. Para isso, o aplicativo permite que o usuário busque por uma comanda ou mesa e carregue todos os itens de todos os pedidos com status "em aberto"daquela comanda ou mesa.

Uma vez carregado, ele consegue editar o pedido da mesma maneira que quando ele está criando um novo pedido com apenas um porém. Para evitar que pedidos sejam gerados e posteriormente editados para excluir itens já consumidos para evitar cobranças, o usuário só irá conseguir deletar ou diminuir a quantidade de um item sob autorização de um usuário com permissão para editar pedidos já salvos. Sem esta permissão, o usuário poderá apenas adicionar novos itens ao pedido, ou aumentar a quantidade de itens já presentes no pedido.

Estes novos itens e quantidades podem ser deletados ou subtraídos pelo usuário sem nenhuma autorização prévia, desde que as quantidades originais antes da edição do pedido sejam respeitadas.

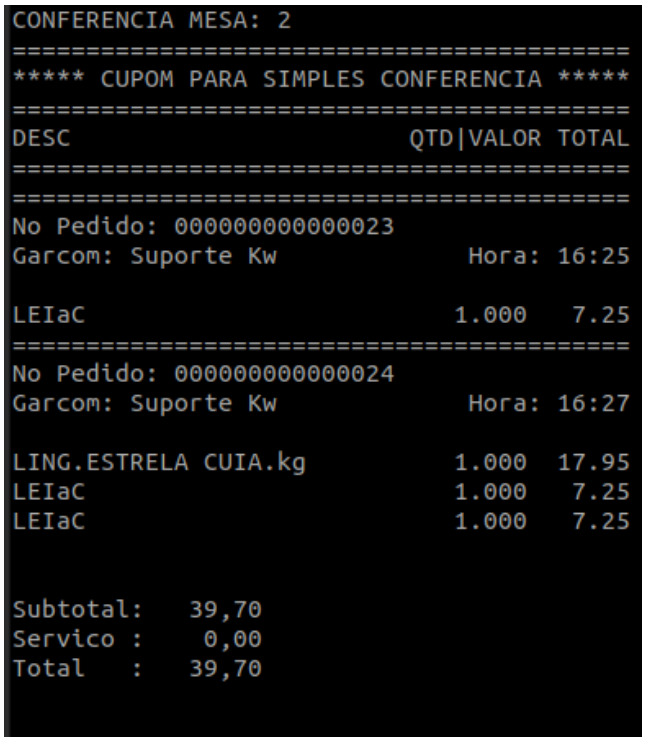
4.3.3 Movendo um pedido

Muitas vezes, quando estamos em um restaurante ou bar, por diferentes razões, acabamos tendo de mudar de lugar, e, como visto anteriormente, nossos pedidos estão associados a mesa que estamos sentados. Sendo assim, foi necessário disponibilizar uma funcionalidade que permitisse aos usuários transportar o pedido de uma mesa ou comanda para outra mesa ou comanda. Esta funcionalidade, assim como a edição de itens já cadastrados anteriormente, exige que um usuário com autorização aprove a transferência caso o usuário atual não tenha tal aleatorização.

4.3.4 Impressão de pedido

Ao final do consumo, é necessário cobrar o cliente, e para isso, é de comum prática, apresentar um cupom de conferencia do que foi consumido pelo usuário. Para isso, foi implementado a funcionalidade de impressão do cupom de conferencia do pedido, que permite ao usuário imprimir um cupom com tudo o que foi consumido em uma mesa ou comanda, com uma totalização parcial e uma totalização com a taxa de serviço aplicada para que o usuário possa verificar os dois valores (Figura 4.2).

Figura 4.2 – Arquivo de conferencia de uma mesa.



```
CONFERENCIA MESA: 2
========================================
***** CUPOM PARA SIMPLES CONFERENCIA *****
========================================
DESC                        QTD|VALOR TOTAL
========================================
========================================
No Pedido: 000000000000023
Garcom: Suporte Kw              Hora: 16:25

LEIaC                          1.000   7.25
========================================
No Pedido: 000000000000024
Garcom: Suporte Kw              Hora: 16:27

LING.ESTRELA CUIA.kg           1.000  17.95
LEIaC                          1.000   7.25
LEIaC                          1.000   7.25


Subtotal:    39,70
Servico :     0,00
Total   :    39,70
```

Fonte: O Autor

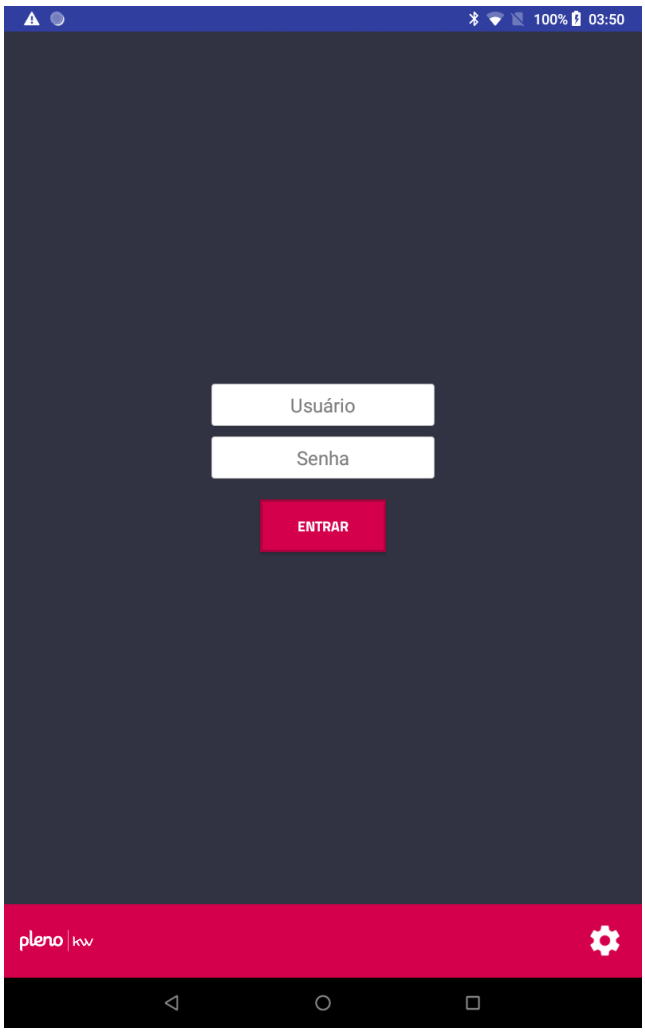
5 INTERFACES DO APLICATIVO

5.1 Tela Inicial

Para inciar o uso do aplicativo, o usuário primeiramente deve efetuar o login, o que ocorre na tela inicial (Figura 5.1) do aplicativo. Nesta tela, temos o campo de usuário e senha, juntamente com um botão de entrar. Esta tela também apresenta uma engrenagem no canto inferior direito, que é um ícone clicável que permite ao usuário acessar a tela de configurações (Figura 5.2) sem a necessidade de efetuar o login.

Aqui, o usuário e senha utilizados são os mesmos já existentes no sistema da empresa, sendo assim, o usuário não irá precisar efetuar um novo cadastro dentro do aplicativo para utilizar o mesmo.

Figura 5.1 – Tela de login do aplicativo.

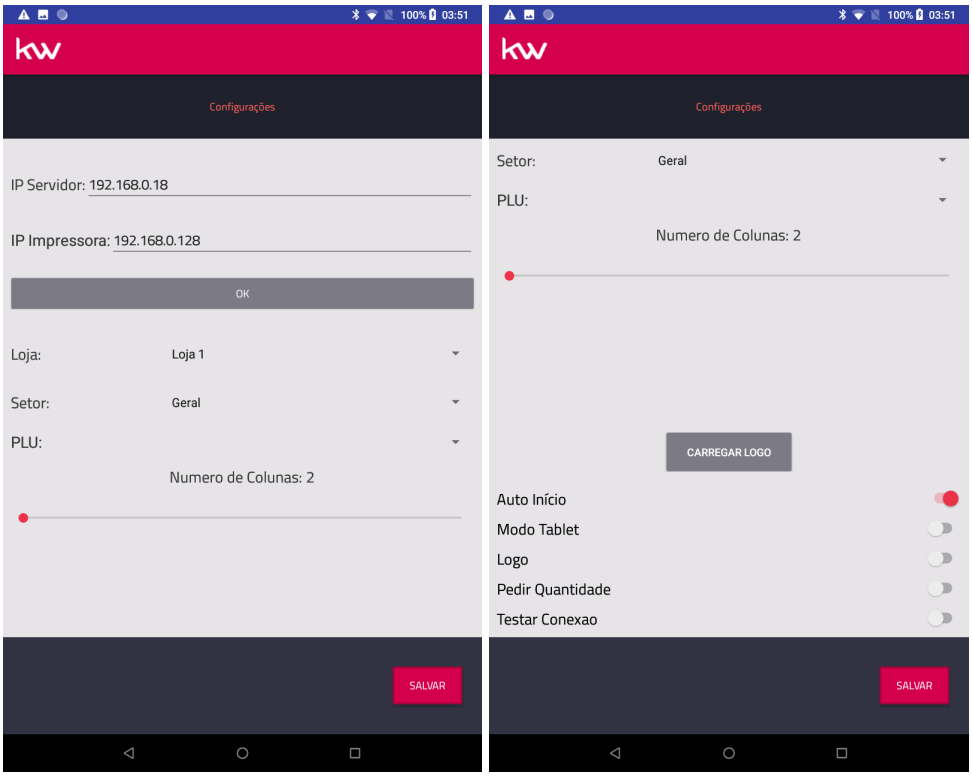


Fonte: O Autor

5.2 Tela de Configurações

Ao clicar na engrenagem, o usuário será redirecionado para a tela de configurações do aplicativo (Figura 5.2).

Figura 5.2 – Tela de login do aplicativo.



Fonte: O Autor

Nesta tela, o usuário terá acesso as diferentes configurações do aplicativo. Sendo elas:

- **IP Servidor**: Este é o endereço de IP onde se encontra rodando o servidor, de onde o aplicativo irá buscar as informações necessárias para o seu funcionamento.
- **IP Impressora**: Este é o endereço de IP onde se encontra a impressora IP, quando existe, que permite que o aplicativo imprima arquivos diretamente nela, sem a necessidade de enviar os mesmos ao servidor.
- **Botão de OK**: Este botão serve para validar o IP selecionado e carregar as informações necessárias para as próximas configurações.
- **Loja**: Aqui, o usuário pode selecionar a loja, na qual ele está trabalho, está informação vem do servidor, e, no caso de um servidor multiloja, temos mais de uma loja que pode ser selecionada pelo usuário.

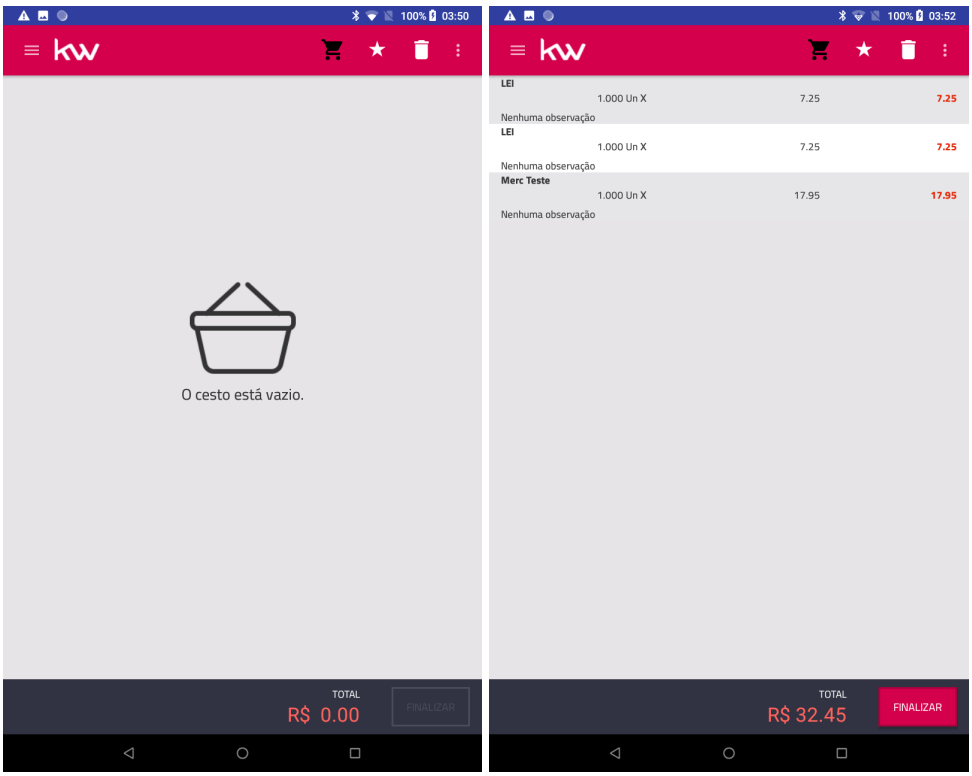
- **Setor**: Aqui, temos o setor onde este aplicativo está rodando, esta configuração irá alterar o menu principal de itens do aplicativo, permitindo que o usuário tenha diferentes menus de acesso rápido pré-cadastrados que podem ser alterados rapidamente através desta configuração.
- **Numero de Colunas**: Esta configuração altera o número de colunar que serão utilizadas para desenhar o menu principal, desta maneira, o usuário pode alterar entre 2 e 8 colunas, para que o menu principal fique melhor distribuído de acordo com o tamanho da tela do dispositivo.
- **Botão Carregar Logo**: Permite ao usuário personalizar o logotipo apresentado na tela de login, tornando o aplicativo um pouco mais personalizado.
- **Auto Inicio**: Permite ao usuário acionar ou não o retorno automático do menu de acesso para o nível mais superior.
- **Modo Tablet**: Esta configuração coloca o menu de acesso rápido ao lado da listagem de itens, ao invés de ter 2 telas separas (Figura 5.6).
- **Logo**: Esta configuração ativa ou não o logotipo carregado.
- **Pedir quantidade**: Esta configuração permite ao usuário solicitar ao aplicativo que sempre que um item for adicionado ao carrinho, a tela de quantidade(Figura 5.10) seja apresentada, quando desativada, o item irá ser adicionado com quantidade igual a 1, permitindo edição futura, se assim desejado.
- **Testar Conexão**: Um problema enfrentado pelos usuários era a perda de conexão com o servidor, esta configuração faz com que o aplicativo fique constantemente verificando a conexão com o servidor, avisando assim ao usuário quando a mesma for perdida, para que ele possa efetuar a reconexão.

5.3 Tela do Carrinho

Ao efetuar o login, o usuário será direcionado para a tela de listagem dos itens (Figura 5.3, que aqui, iremos chamar de carrinho, para melhor entendimento. Nesta tela, o usuário, além de ver o total do pedido até o momento, poderá verificar também os itens já adicionados ao pedido.

A partir desta tela que temos todas as operações que podem ser feitas no aplicativo. No canto superior esquerdo, temos o menu lateral (Figura 5.4), onde, através dele, podemos acessar a tela de mesas (Figura 5.7), a tela de comandas (Figura 5.8), a tela

Figura 5.3 – Tela de listagem.



Fonte: O Autor

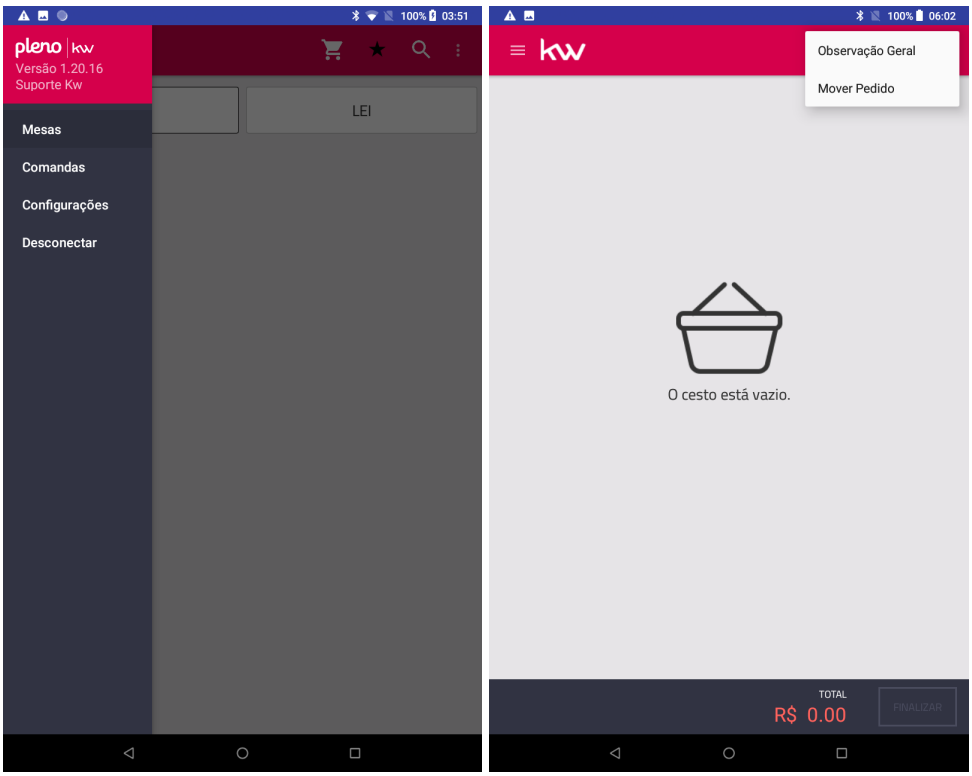
de configurações (Figura 5.2) ou desconectar do sistema, voltando para a tela de login (Figura 5.1).

No canto superior direito, temos a figura de um carrinho de supermercado, que funciona como atalho para acessar a tela de listagem de itens (Figura 5.3), temos uma estrela que serve para acessar a tela de menu principal ((Figura 5.5), quando em modo normal, também temos uma lixeira para limpar o carrinho e deletar todos os itens do pedido de uma única vez e o 3 pontos, para acessar o menu lateral direito (Figura 5.4), que nos permite efetuar diferentes ações de acordo com a tela em que nos encontramos.

No nosso canto inferior direito, conseguimos localizar o texto de totalização com o valor atual do carrinho, e um botão de finalizar, que só fica ativo no momento em que um ou mais itens são adicionados ao carrinho. O uso do botão finalizar irá nos direcionar para a seleção de mesa e comanda onde aquele pedido deve ser registrado, após isso, o aplicativo volta para a tela de carrinho vazio para iniciar um novo pedido.

No menu lateral desta tela, temos duas opções, sendo a primeira a Observação Geral, que permite ao usuário fazer uma anotação de texto livre que será impressa ao final do pedido para que as pessoas que forem preparar o pedido possam ser informadas

Figura 5.4 – Menus Laterais



Fonte: O Autor

de alguma coisa, como, por exemplo, "Preparar o Xis junto com as batatas". A segunda opção é o mover pedido, utilizado para mover o pedido de uma mesa ou comanda para outra mesa ou comanda, para isso, primeiramente é necessário que seja carregado um pedido já existente para efetuar o movimento.

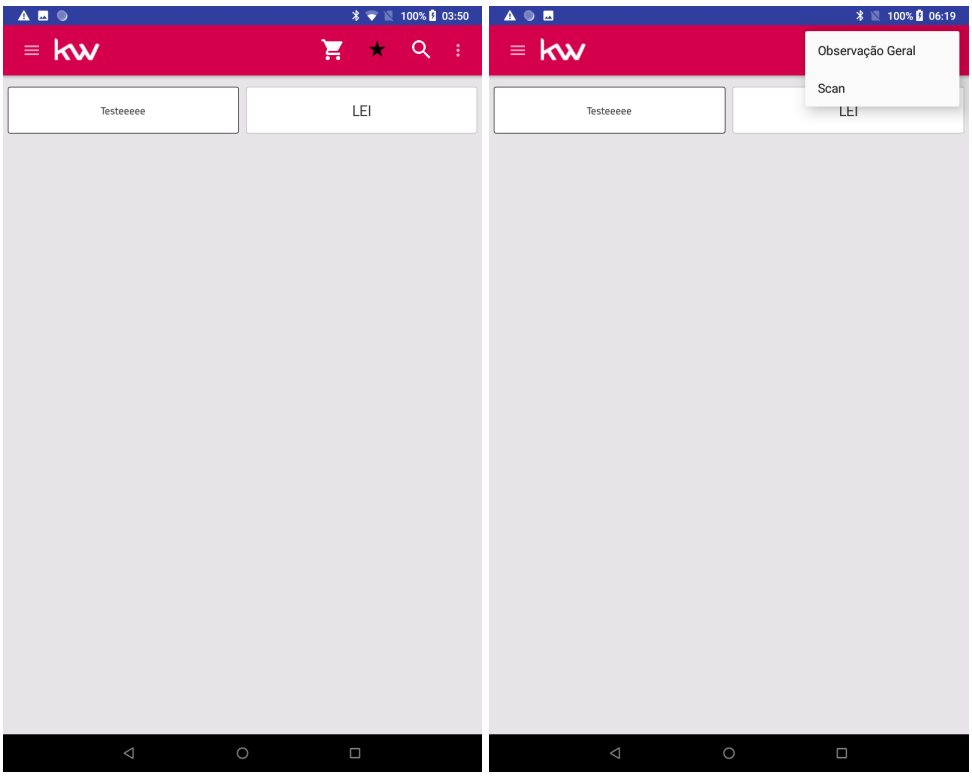
5.4 Tela Menu Principal

TODO - Trocar imagem do menu por uma com um menu mais bonito

Nesta tela temos o menu de acesso rápido de diferentes itens que podem ser configurados na interface Web já existente. O menu se comporta como um grafo, onde cada botão pode ser um nó do grafo ou então um vértice, sendo um vértice, obrigatoriamente ele será um item a ser adicionado, sendo um nó, ele será um grupo de itens e outros grupos, não tendo limite de profundidade.

Aqui podemos ver que o ícone de lixeira é substituído por um ícone de lupa, este ícone permite ao usuário buscar por um item através do seu nome ou código de venda. No menu direito, temos aqui duas opções, a primeira, já apresentada na tela anterior,

Figura 5.5 – Menu Principal



Fonte: O Autor

a segunda, permite ao usuário utilizar a câmera do dispositivo para efetuar a leitura do código de barras do produto, quando este estiver disponível.

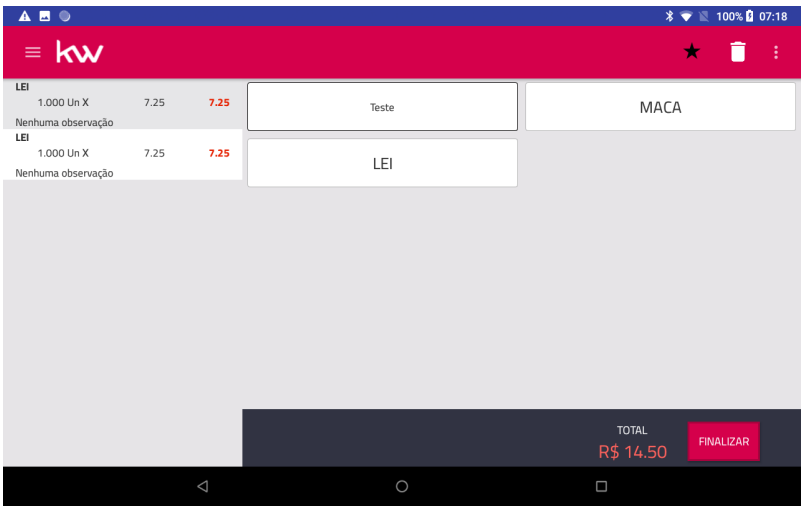
Nesta tela, ao selecionarmos um item do menu, se o mesmo for um produto, o produto será adicionado ao carrinho, e, se a opção "Auto Home"apresentada anteriormente estiver ativa, o menu volta para o nó raiz, caso contrário, ele se mantém no mesmo nível atual. A tecla de voltar nativa doa Android pode ser utilizada para regressar os níveis do menu quando necessário. Se o item selecionado for um grupo, a aplicação irá redesenhar o menu apresentando os novos componentes de acordo com o que estiver configurado dentro daquele grupo.

Uma requisição não funcional do aplicativo pode ser vista aqui, que é o uso de diferentes layouts de botões para que o usuário posso diferenciar entre grupos e itens, onde grupos possuem a borda preta e itens não possuem borda. Tal funcionalidade foi uma ideia apresentada durante os testes betas efetuados com clientes selecionados da empresa.

5.5 Tela Modo Tablet

Quando ativo o modo tablet, as telas de menu principal e listagem de itens se unem para formar uma única tela onde o usuário tem visualização dos itens na esquerda e o menu de acesso rápido na direita, sem a necessidade de trafegar de uma tela para a outra. Este modo, como o nome já diz, foi desenvolvido para clientes que utilizam tablets com telas maiores, permitindo assim um uso agradável neste modo.

Figura 5.6 – Tela em modo Tablet.



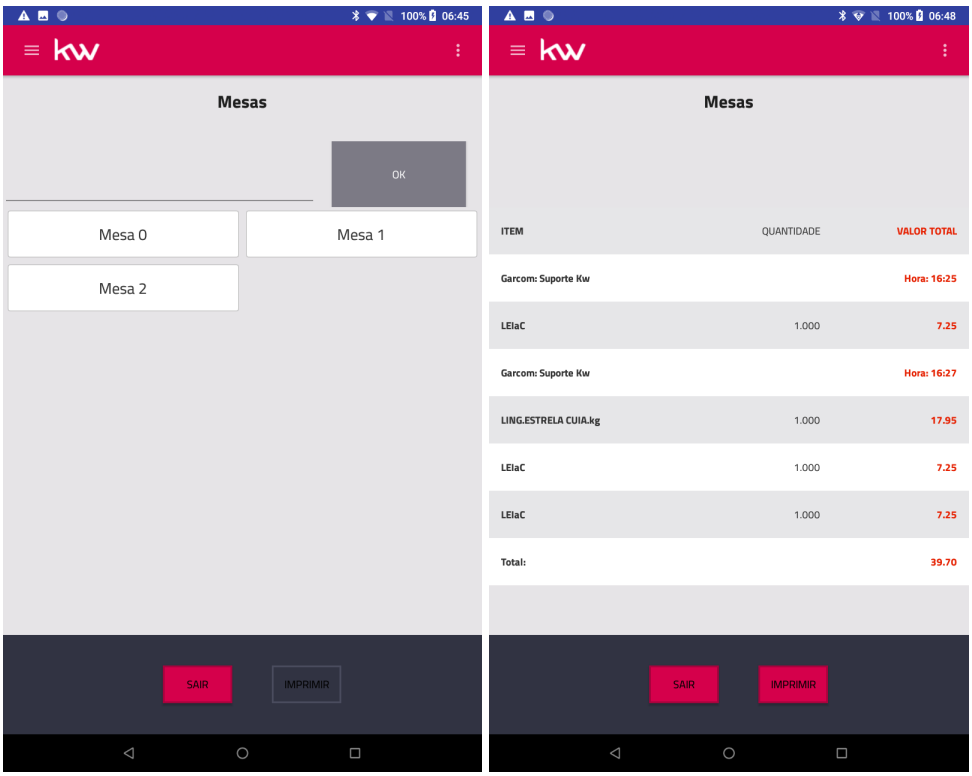
Fonte: O Autor

5.6 Tela de Mesas

Nesta tela temos a listagem de todas as mesas que possuem um ou mais pedidos com status "em aberto"registrado nelas. Ao selecionar qualquer uma das mesas apresentadas na listagem, o usuário será redirecionado para uma tela de conferência (Figura 5.7) onde ele poderá verificar todos os itens registrados naquela mesa e sua totalização, bem como qual foi o usuário que registrou cada item daquela mesa.

No caso de existirem muitas mesas no estabelecimento, o usuário pode acessar a mesa deseja digitando o número dela e utilizando o botão OK para carregar a mesma.

Figura 5.7 – Conferencia de Mesas



Fonte: O Autor

5.7 Tela de Comandas

Muito parecida com a tela de mesas, a tela de comandas traz para o usuário todas as comandas com pedidos em aberto no estabelecimento, enquanto a mesa trás todos os pedidos de todas as comandas daquela mesa, a tela de comandas trás apenas os pedidos registrados naquela comanda específica.

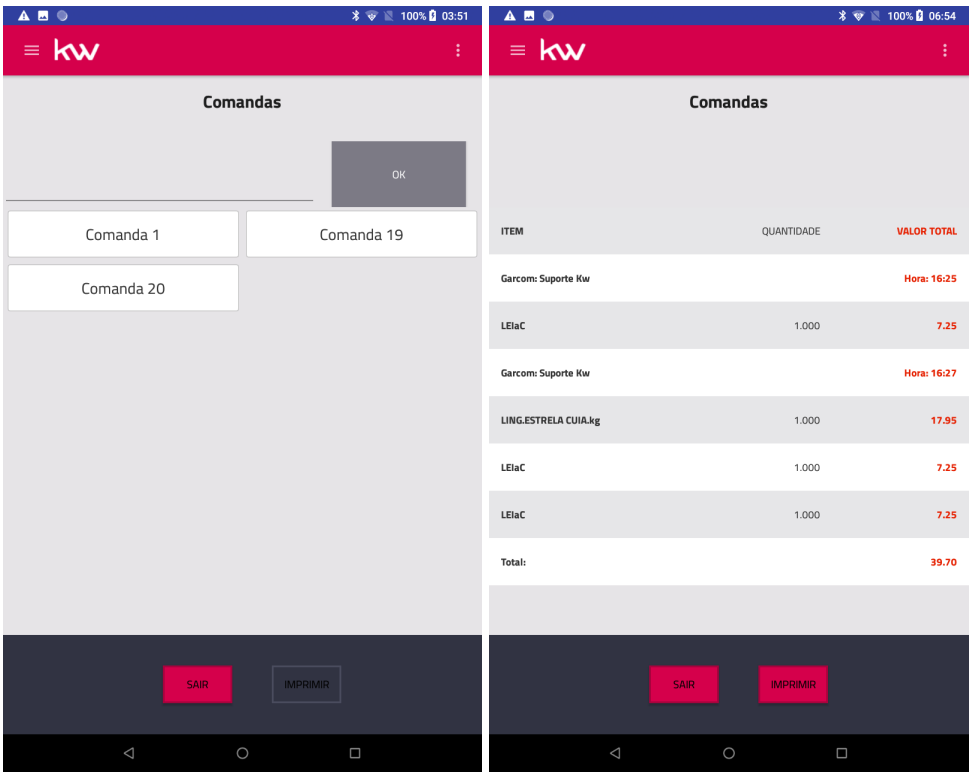
Novamente, no caso de existirem muitas comandas, o usuário pode acessar uma comanda específica apenas digitando o número da mesma e utilizando o botão de OK.

5.8 Conferência

Em ambos os casos, seja na tela de mesas ou na tela de comandas, no menu lateral direito, temos acesso a 3 funcionalidades, como podemos ver na figura 5.9.

- **Carregar pedido**: Permite ao usuário carregar todos os itens lançados naquela mesa ou comanda para efetuar a manutenção do pedido.

Figura 5.8 – Conferencia de Mesas



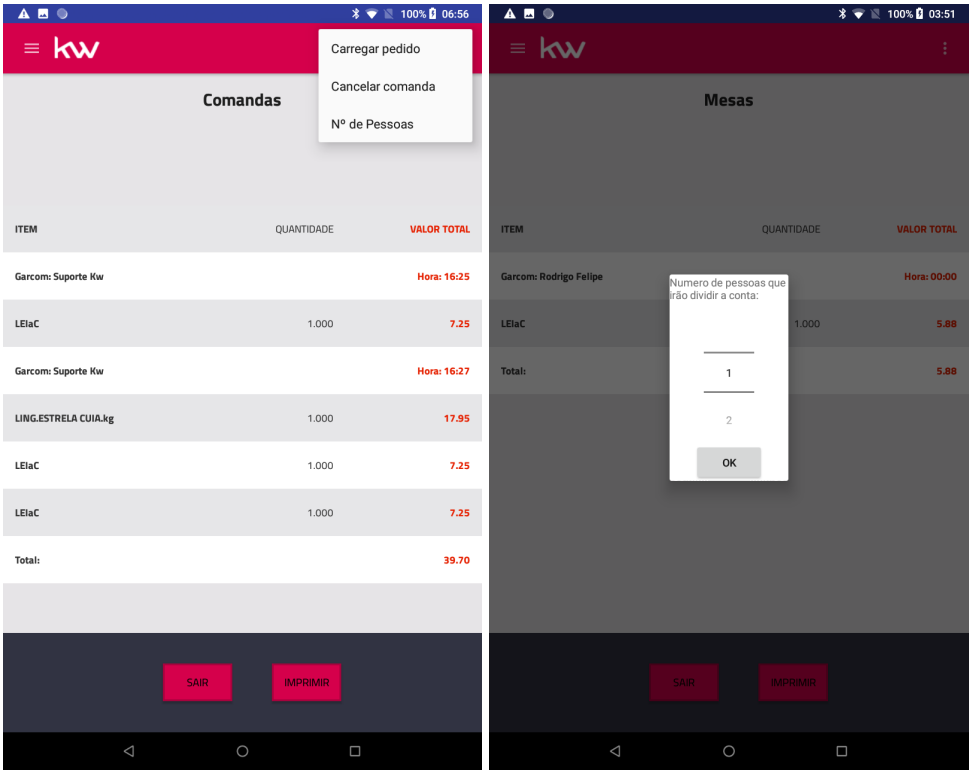
Fonte: O Autor

- **Cancelar Comanda/Mesa**: Permite ao usuário cancelar todos os pedidos daquela comanda ou mesa selecionada.
- **Número de Pessoas**: Permite ao usuário dizer o número de pessoas que irão dividir o valor total daquela comanda ou mesa (Figura 5.9, desta maneira, ao solicitar a impressão do cupom de conferencia, no mesmo será adicionado uma linha trazendo o valor por pessoa já calculado.

5.9 Tela de Edição de Item

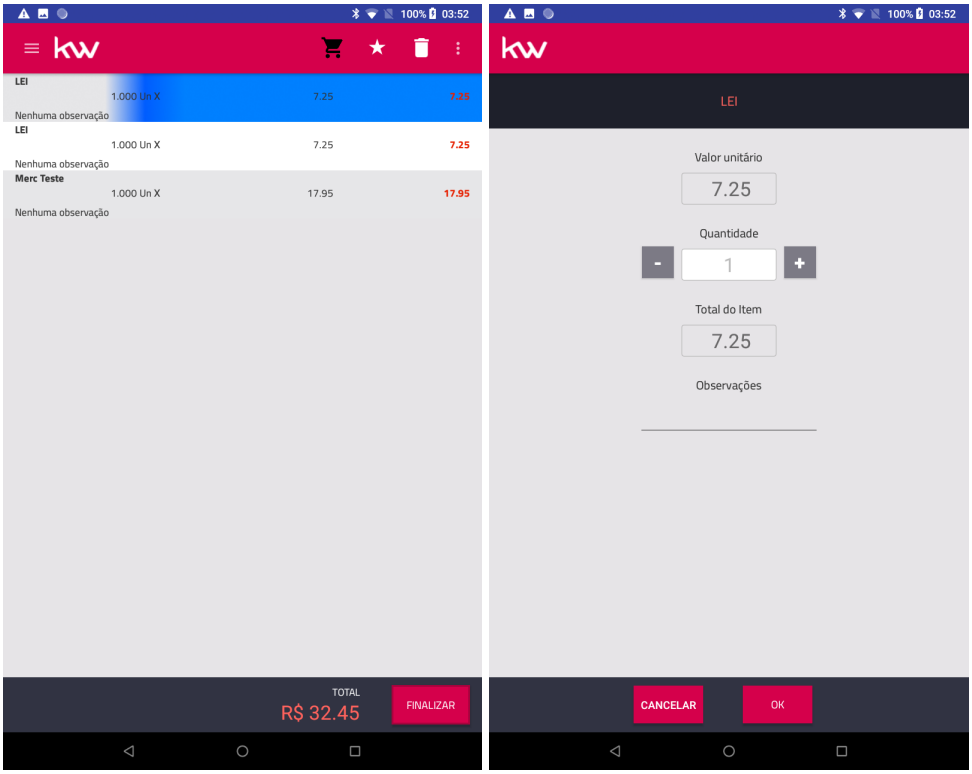
Como comentado anteriormente, os itens do pedido podem ser editados através de uma tela de edição do item. Esta tela tem como sua principal funcionalidade permitir que o usuário ajuste a quantidade de cada item do pedido.

Figura 5.9 – Menu lateral de conferencia e Popup de seleção do número de pessoas.



Fonte: O Autor

Figura 5.10 – Feedback de edição de item e tela de valor e quantidade



Fonte: O Autor

6 CONCLUSÃO

A ideia do projeto era poder fornecer uma solução simples e eficaz para resolver a dificuldade em registrar pedidos e otimizar estes mesmos registros de maneira que, ao final, tudo estaria integrado com um sistema já existente, permitindo ao usuário poder reaproveitar boa parte da sua infra-estrutura e não exigir do mesmo grandes invertimentos para a implantação do aplicativo em seu estabelecimento.

6.1 Avaliação dos Usuários

Para validar a usabilidade do aplicativo, foi criado um formulário SUS (*System Usability Score*) com 10 perguntas com respostas entre 1 e 5, onde 1 representa "Discordo Totalmente"e 5 represente "Concordo Totalmente". O resultado foi de 79 pontos, colocando o aplicativo na classificação B de usabilidade pela escala do sistema utilizado, o que nos mostra que, apesar de alguns problemas ainda existentes no sistema, em um âmbito geral, ele atende as necessidades do usuário, tornando a rotina do mesmo mais simples e prática.

Adicionei também ao formulário um campo de texto simples para que os usuários pudessem deixar algum comentário em relação ao aplicativo, e aqui destaco dois deles: "Sistema fácil de usar, prático e objetivo."e "A aplicação funciona bem, Instalei em um restaurante e esta integrado ao servidor de pedidos para impressão dos pedidos na Cozinha e Pizzaria", onde o primeiro é de um funcionário de um dos restaurantes piloto do projeto e o segundo, de um parceiro nosso que participou do processo criativo do aplicativo.

6.2 Problemas não resolvidos

Com o desenvolver do projeto, e com a interação dos usuários de mundo real, se tornou claro que a escolha por uma comunicação socket acabou se tornando muito mais problemática do que o esperado, não tendo compensado o ganho de tempo em desenvolvimento. O Ideal para o projeto seria ter desenvolvido um conjunto de micro-serviços acessíveis através de uma API REST sem a necessidade de manter a comunicação socket aberta durante todo o tempo de execução do aplicativo.

Outro ponto foi algumas escolhas de design que poderiam ser melhoras, algu-

mas funcionalidades se encontram em pontos um pouco "Escondidos"do aplicativo, não sendo utilizados pelos usuários que não saber que estas funcionalidades existem, como por exemplo, o Popup com o número de pessoas para dividir a conta.

6.3 Aprendizados

O desenvolvimento deste aplicativo foi o meu primeiro contato com o desenvolvimento mobile, tudo que foi desenvolvido aqui, eu tive que aprender do zero, e acredito que, ao desenvolver o aplicativo, eu tive um grande aprendizado, que será de extrema importância no futuro da empresa, uma vez que, ao apresentar o aplicativo ao meu chefe, não só recebi uma bonificação pelo desenvolvimento, como também fui selecionado para ser um dos desenvolvedores do novo PDV Android que a empresa está começando a desenvolver.

Poder desenvolver este aplicativo em parceria com tantas pessoas que trouxeram inúmeros Feedbacks ao londo do desenvolvimento me permitiu entender que, no final do dia, a pessoa que sente as "dores"é a melhor analista de sistemas que um desenvolvedor pode ter, pois ela entende não só do problema a ser resolvido, mas também sabe dizer quais soluções não funcionariam devido a outros problemas que nem passaram pela nossa cabeça durante o desenvolvimento

6.4 Desenvolvimento Futuro

Apesar de atender com sucesso os objetivos propostos aqui neste trabalho, ainda tem muito o que ser feito para tornar o aplicativo ainda mais atraente e funcional para os seus usuários. O desenvolvimento de uma API REST para efetuar consultas e registros no banco de dados ao invés do uso de comunicação soket seria o primeiro grande ponto de mudança.

Alguns outros ajustes dentro do que já se encontra hoje no aplicativo também são necessários, bem como a integração com hardwares mais novos, para podermos, por exemplo, utilizar de biometria para efetuar o login do usuário, ou então, utilizar da tecnologia RFID para a leitura de comandas, sem a necessidade de digitar ou escanear os códigos.

REFERÊNCIAS

ALMEIDA, M. B. Uma introdução ao xml, sua utilização na internet e alguns conceitos complementares. **Ciência da informação**, SciELO Brasil, v. 31, p. 5–13, 2002.

DE, B. Api management. In: ____. **API Management: An Architect's Guide to Developing and Managing APIs for Your Organization**. Berkeley, CA: Apress, 2017. p. 15–28. ISBN 978-1-4842-1305-6. Available from Internet: <https://doi.org/10.1007/978-1-4842-1305-6_2>.

GOOGLE. **Android para desenvolvedores**. 2023. Available from Internet: <https://developer.android.com/?hl=pt-br>.

GOOGLE. **Android Studio Home page**. 2023. Available from Internet: <https://developer.android.com/studio>.

GROFF, J. R.; WEINBERG, P. N.; OPPEL, A. J. **SQL: the complete reference**. [S.l.]: McGraw-Hill/Osborne, 2002.

INC, A. **The standards-based, open source printing system developed by Apple for iOS®, iPadOS®, and macOS®. CUPS uses IPP Everywhere™ to support printing to local and network printers.** 2022. Available from Internet: <https://www.cups.org/>.

LIMA, G. **Saiba tudo sobre o IDE - Integrated Development Environment**. 2022. Available from Internet: <https://www.alura.com.br/artigos/o-que-e-uma-ide>.

MARINS, T. C. C. P. e F. A. S. Sistemas erp: características, custos e tendências. **Production**, v. 15, n. 1, 2005. ISSN 1980-5411. Available from Internet: <https://doi.org/10.1590/S0103-65132005000100009>.