Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

RUnB: Aplicativo para o Restaurante Universitário da Universidade de Brasília

Bruno Viana de Siqueira

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Aletéia Patrícia Favacho de Araújo

Brasília
2020

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

RUnB: Aplicativo para o Restaurante Universitário da Universidade de Brasília

Bruno Viana de Siqueira

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Aletéia Patrícia Favacho de Araújo (Orientador)
CIC/UnB

Prof. Dr. Jacir Luiz Bordim
CIC/UnB

Prof. Dr. Wilson Veneziano
CIC/UnB

Prof. Dr. Edil Guimarães
Coordenador do Curso de Engenharia da Computação

Brasília, 31 de janeiro de 2020

Dedicatória

Dedico este trabalho a minha família, especialmente aos meus pais, Fernando e Lília, por me proporcionarem as condições e o apoio incondicional que fizeram possível que eu chegasse até aqui.

Agradecimentos

Agradeço ao corpo docente de UnB, especialmente à professora Dra. Aletéia Patrícia, por ter, desde o início, acreditado e apoiado a ideia, além de ter proporcionado excelente orientação que fez com que este trabalho fosse possível. Agradeço também ao CPD/UnB, representado pela equipe do Felipe Evangelista, pelo trabalho conjunto e disponibilidade irrestrita durante o desenvolvimento do trabalho. Por fim, agradeço a direção do RU, representada pela diretora Cristiane Moreira, pela disponibilidade e apoio ao projeto.

Resumo

Este trabalho apresenta o aplicativo RUnB, um aplicativo móvel desenvolvido com o objetivo de prover soluções para o restaurante universitário da Universidade de Brasília. Dada a formação de grandes filas no RU para a compra de créditos e a demanda evidente da comunidade acadêmica por soluções que facilitem o dia a dia do estudante ao utilizar o restaurante, verificou-se a necessidade da elaboração deste trabalho. Para tanto, realizou-se um levantamento das funcionalidades presentes nos aplicativos com a temática de restaurante universitário no Brasil, em seguida efetuou-se uma pesquisa com os usuários do RU com o objetivo de validar e elencar funcionalidades do aplicativo e, então, partiu-se para a fase de implementação. Com o aplicativo RUnB, os usuários podem realizar a compra de créditos pelo celular, diminuindo, assim, as filas formadas diariamente para a aquisição de créditos. Além disso, o aplicativo implementou outras soluções para facilitar a vida do estudante da UnB, como o acesso simplificado ao cardápio e a geração do código de barras para acesso aos refeitórios. Os dados iniciais de avaliação mostraram que 90% dos usuários estão satisfeitos com a versão inicial do aplicativo. Este aplicativo se destina a todos os usuários do restaurante universitário da UnB.

Palavras-chave: Restaurante Universitário, Aplicativo, Android, Universidade de Brasília

Abstract

This paper presents the RUnB App, a mobile application developed with the aim of providing solutions for the university restaurant (RU) at the University of Brasilia (UnB). Given the long lines in the RU for the purchase of credits and the evident demand from the academic community for solutions that facilitate the student's daily life when using the restaurant, there was a need to carry out this work. For this purpose, a research of the features present in applications with the brazilian university restaurant theme was conducted, followed by a survey that was carried out with restaurant users in order to validate and list the application's features, and then, it went on to an implementation phase. With the RUnB application, users can make a purchase of credits by cell phone, thus reducing the number of lines formed daily for the acquisition of credits. In addition, the application implements other solutions to make life easier for UnB students, such as simplified access to the menu and the generation of bar codes for providing access to the dining halls. Recent evaluation data shows that 90% of users are satisfied with the initial version of the application. This application is intended for all users of UnB's university restaurant.

Keywords: University Restaurant, App, Android, University of Brasília

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A universidade pública se tornou uma verdadeira casa para grande parte dos estudantes brasileiros. O fato de os estudantes possuírem uma carga horária extensa, muitas vezes espalhada em horários com grandes intervalos de tempo, aliado à dificuldade de locomoção de suas casas para a universidade, fez com que muitos estudantes passassem mais tempo dentro da própria academia do que em seus lares.

Nesse contexto, inserem-se os restaurantes universitários. Esses restaurantes aliam a praticidade de realizar as refeições dentro da universidade ao preço subsidiado, se tornando, assim, um grande atrativo aos estudantes, que muitas vezes não possuem condições financeiras de realizar refeições em restaurantes particulares. Dessa forma, grande parte dos alunos realiza diariamente uma ou mais refeições em um restaurante universitário.

Consequentemente, o restaurante universitário deixa de ser apenas um local de alimentação e passa a fazer parte da rotina de seus usuários. Em um momento de inovações tecnológicas, no qual as pessoas se veem diariamente cercadas por novas tecnologias, é natural que a demanda por soluções que facilitem o dia a dia na universidade cresça.

Por conseguinte, a alta demanda por um serviço resulta, muitas vezes, na formação de filas. No ambiente do restaurante universitário da Universidade de Brasília as filas são um problema real e explícito. Enfrentar grandes filas com frequência não é algo incomum para um usuário do RU.

As filas estão presentes no cotidiano das pessoas em diferentes locais e situações. Um dos maiores transtornos causados pelas filas é o tempo que ela consome dos usuários. É consenso no mundo moderno que o tempo é um recurso de extrema importância, e a gestão desse tempo é algo indispensável.

Assim sendo, existe uma demanda natural dos usuários do Restaurante Universitário por uma solução que vise mitigar o problema das filas. O aplicativo proposto neste trabalho possui como um dos objetivos principais a redução das filas que se formam nos caixas

para a aquisição de créditos no RU. Para tal, a principal funcionalidade implementada pelo aplicativo é um mecanismo de compra de créditos por meio do aplicativo.

Para além da solução mencionada no parágrafo anterior, o aplicativo implementa outras funcionalidades que atendem demandas dos usuários. O aplicativo reúne em uma só plataforma soluções para diversos problemas, tais como:

- Acesso rápido ao cardápio;
- Acesso rápido e centralizado a informações importantes, como o saldo, extrato e grupo pertencente;
- Acesso ao refeitório através de um gerador de código de barras, tornando dispensável portar a carteirinha ou imprimir uma ficha de acesso;
- Visualização das avaliações do cardápio, ajudando assim no processo de tomada de decisão;
- Mecanismo de notificação, que possibilita informar os usuários do RU de maneira ampla, simples e rápida;

A partir de todas essas funcionalidades, o aplicativo RUnB possibilita que o usuário despenda menos tempo ao utilizar o restaurante universitário, otimizando sua experiência nesta dependência da universidade e melhorando sua qualidade de vida.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver um aplicativo para o restaurante universitário da UnB. Dentre as suas funcionalidades, o aplicativo deve possibilitar a compra de créditos e o acesso simplificado ao cardápio e a outras funcionalidades importantes no dia a dia do usuário RU. Dessa maneira, para cumprir o objetivo geral desse trabalho, foram definidos os seguintes objetivos específicos:

- Desenvolver funcionalidade de compra de créditos do RU por aplicativo;
- Implementar um mecanismo para acesso facilitado aos refeitórios;
- Prover acesso facilitado ao cardápio da semana, levando em conta o campus, o dia e a refeição;
- Centralizar informações essenciais ao usuário, tais como seu saldo e grupo de usuário;
- Aumentar a transparência da movimentação financeira do usuário através da apresentação de um extrato financeiro;

- Criar um mecanismo de avaliação do cardápio;
- Possibilitar o envio amplo de informes por parte da direção do RU.

1.2 Metodologia

O trabalho foi dividido em etapas. Assim, a primeira etapa, consistiu no amadurecimento e desenvolvimento da ideia, além do levantamento de funcionalidades através de uma pesquisa com os usuários. Nessa etapa foram realizadas pesquisas por trabalhos e aplicativos similares, de modo a levantar possíveis funcionalidades para o RUnB e definir os requisitos necessários para que essas funcionalidades fossem implementadas.

A segunda etapa focou na validação, junto a direção do RU e a UnB, da possibilidade de implementação das funcionalidades que dependeriam diretamente de acesso aos bancos de dados da UnB e de colaboração da gestão do RU. Nessa etapa foram realizadas diversas reuniões com o CPD/UnB, com a direção do RU e com membros da empresa administradora do restaurante. Dessa forma, foi possível definir quais funcionalidades viriam realmente a ser implementadas.

A terceira etapa do trabalho focou no desenvolvimento do aplicativo. Esse desenvolvimento contou com a colaboração constante do CPD/UnB, que desenvolveu vários dos serviços *web* utilizados por esse aplicativo, e da direção do RU, que acompanhou todo o processo de desenvolvimento e tomadas de decisão. A quarta e última etapa desse trabalho teve como foco a validação, por parte dos usuários, das funcionalidades implementadas no aplicativo.

1.3 Estrutura do Trabalho

Este trabalho está divido, além deste capitulo introdutório, nos seguintes capítulos:

- Capítulo 2: Este capítulo descreve o estado da arte dos aplicativos com temática de restaurante universitário no Brasil;
- Capítulo 3: Apresenta as tecnologias envolvidas no desenvolvimento do aplicativo RUnB;
- Capítulo 4: Este capítulo apresenta o aplicativo RUnB, descrevendo o levantamento de requisitos, as funcionalidades e os resultados;
- Capítulo 5: Este capítulo apresenta a conclusão do trabalho e algumas possibilidades para trabalhos futuros;

Capítulo 2

Aplicativos de RU no Brasil

Segundo Preece, Jenny *et al.* [3], o melhor ponto de partida quando se pensa em projetar produtos interativos usáveis consiste em comparar bons e maus exemplos. Assim sendo, esse será o objetivo principal deste capítulo. Este capítulo descreve os aplicativos desenvolvidos com a temática de restaurante universitário no Brasil. Primeiro, será exposto uma breve visão geral sobre a relação entre restaurantes universitários no Brasil e seus aplicativos móveis. Segundo, será explorado o funcionamento das aplicações por funcionalidade, realizando a comparação entre exemplos adequados e inadequados de implementação de cada funcionalidade. Por último, será feito um comparativo entre as aplicações analisadas, apresentando o que cada uma possui ou não possui.

2.1 Restaurante Universitário e Aplicativos

No Brasil, hoje, existem 68 Universidades Federais e 44 Universidades Estaduais em funcionamento, totalizando 112 universidades públicas. A maioria dessas universidades possui seu próprio restaurante universitário, com diferentes preços e estruturas. No entanto, a imensa maioria dessas universidades não tem um Aplicativo Móvel para uso da comunidade acadêmica. Em levantamento na loja de aplicativos dos dispositivos *Android*, a *Google Play Store*, foram observados 21 aplicativos relacionados a temática de restaurante universitário em universidades públicas disponíveis para *download*. Esses aplicativos são pertencentes as seguintes universidades: UFT, UFRJ (dois), UFPA (dois), USP, UFC, UnB, UESC, UFG, UFPR, UFAC, UFV (dois), UFRPE, UFMS, UFRN, UECE, UFRGS Mobile, Unicamp, UFMA e Unipampa. Além desses, a PUC-RIO (universidade particular) também possui dois aplicativos relacionados ao seu restaurante universitário.

É importante ressaltar que todos os aplicativos relacionados aos RUs citados possuem um número elevado de *download* (que variam entre 1.000 e 10.000), evidenciando

sua popularidade e a alta demanda da comunidade acadêmica por aplicações com essa temática.

Os aplicativos de RU desenvolvidos para as universidades públicas são, em sua maioria, aplicações desenvolvidas por alunos e ex-alunos e, dessa forma, propriedade desses alunos (publicados por eles nas lojas de aplicativos). Por essa razão, muitos acabam sem manutenção e por isso são descontinuados. Um exemplo disso é o aplicativo "RUMOR" da UnB, que é um aplicativo que apresenta o cardápio do RU da Universidade de Brasília. A aplicação não realiza mais nenhuma função, mesmo que ainda esteja disponível na loja de aplicativos. Aplicativos de outras universidades que não realizam nenhuma função podem ser encontrados, sendo quase que integralmente aplicativos publicados por alunos. Por outro lado, existem os aplicativos publicados e mantidos pelas próprias universidades. Alguns exemplos são o aplicativo "Cardápio" de propriedade da USP, o "UFMA Mobile" da UFMA, o "RU UFRN" da UFRN e o "UFMS Digital", da UFMS, dentre outros. Esses aplicativos, principalmente por serem de propriedade das universidades, funcionam corretamente e recebem atualizações constantes, como é possível verificar no histórico de atualizações disponível na página da aplicação na *Google Play Store*.

O aplicativo presente neste trabalho, por ter sido desenvolvido como projeto de conclusão de curso, será de propriedade do CPD/UnB e, com isso, receberá a devida manutenção e atualização.

2.2 Principais Funcionalidades

Na pesquisa de aplicativos, foram encontrados diferentes funcionalidades e maneiras de apresentar cada uma delas. Como funcionalidades identificadas, podemos citar, da mais recorrente para a menos recorrente: Apresentação do Cardápio, Apresentação do Saldo, Apresentação do Extrato, Notificação, Compartilhamento do Cardápio, Avaliação do Cardápio e Compra de Créditos. Cada funcionalidade identificada serve de inspiração para a implementação do aplicativo exposto nesse trabalho.

Dessa forma, cada funcionalidade será explorada individualmente nas seções seguintes desse capitulo, avaliando, de forma crítica e com suporte da literatura, implementações adequadas e inadequadas. Utilizaremos de capturas de tela dos aplicativos pesquisados para apresentar, na prática, a implementação e funcionamento da aplicação.

2.3 Funcionalidade Cardápio do Dia

A funcionalidade "Cardápio do Dia" está presente em todos os aplicativos relacionados a temática de restaurante universitário encontrados na pesquisa redijida. Nessa funci-

onalidade os possíveis parâmetros de entrada do usuário são a escolha do Campus (se existir mais de um), o dia da semana e a refeição (café da manhã, almoço e jantar). Nessa funcionalidade, o que difere uma aplicação da outra é a maneira na qual o cardápio é apresentado. Alguns apresentam o cardápio de maneira desorganizada, com muita informação junta e mal estruturada, além de um *design* pouco amigável à experiência do usuário. Já outros, fazendo bom uso dos padrões de *design* do *Android*, de testes de validação e até do bom senso, conseguem apresentar o cardápio de maneira mais clara, natural e limpa.

Nos aplicativos pesquisados, há alguns bons exemplos de apresentação clara e amigável do cardápio, assim como exemplos medianos ou inadequados. Um ótimo exemplo de boa implementação da funcionalidade de apresentação do cardápio e de seu conteúdo é o aplicativo "Cardápio RU UFRJ". Nele o usuário primeiro escolhe o campus de interesse e, depois, por meio do componente *Spinner* do *Android*, escolhe o dia da semana e a refeição. Dessa maneira, o usuário pode facilmente alterar a escolha de dia e refeição de interesse. Outro detalhe interessante é que o aplicativo pré seleciona o dia atual, o que facilita bastante a vida do usuário e melhora a usabilidade da funcionalidade. A Figura 2.1 apresenta a implementação dessa funcionalidade no aplicativo "Cardápio RU - UFRJ".

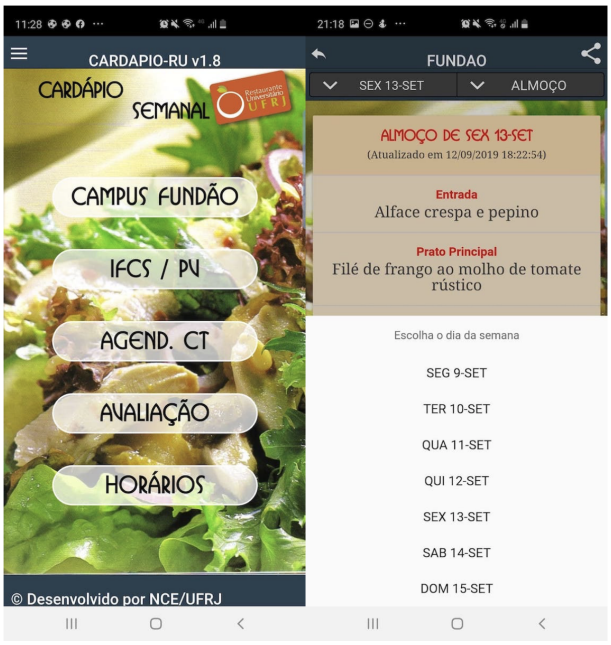


Figura 2.1: Escolha do Cardápio no Aplicativo "Cardápio RU - UFRJ".

Por outro lado, na pesquisa redijida foram encontrados diversos exemplos inadequados da implementação dessa funcionalidade. Dois desses exemplos são o aplicativo "Cardápio RU - UFBA" e o aplicativo "Guia do Aluno Unipampa", que podem ser vistos na Figura 2.2. O primeiro, apresenta as informações de forma visualmente desorganizada, o que dificulta bastante a leitura, principalmente para pessoas que possuem dificuldade

(a) Cardápio RU - UFBA.

(b) Guia do Aluno Unipampa.

Figura 2.2: Exemplos Inadequados de Apresentação do Cardápio.

com tecnologia. O segundo não possui nenhum tipo de estruturação dos dados. É exibido um PDF rotacionado, sem nenhum tipo de interação do usuário para filtragem dos dados. Ademais, a aplicação não possibilita o *download* desse documento, o que poderia facilitar a visualização posterior.

2.4 Funcionalidade de Saldo e Extrato

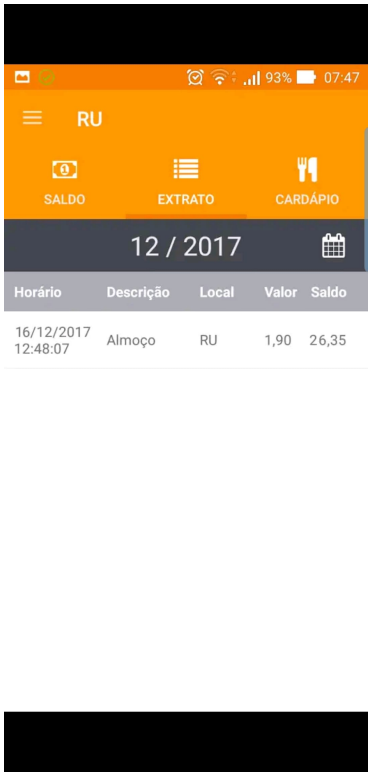
Geralmente, a funcionalidade de "Saldo no cartão do restaurante universitário" está presente nos aplicativos com a temática de RU administrados pelas universidades, visto que tal funcionalidade demanda acesso a dados dos alunos. Em alguns aplicativos, como "UFMA Mobile" e "Cardápio UFV", a funcionalidade de saldo está conectada a funcionalidade de "Extrato". A funcionalidade de "Extrato do restaurante universitário" funciona de maneira similar ao "Extrato Bancário". O extrato bancário é um relatório contendo informações sobre a movimentação e o saldo de uma conta bancária. O "Extrato do

RU" também é um relatório sobre movimentação e saldo (transação), porém, reflete a movimentação financeira da sua "conta" do RU.

Um ótimo exemplo de implementação e apresentação clara e bem estruturada do saldo e extrato é o aplicativo da UFMA, citado no parágrafo anterior. Nele, é possível ver e distinguir com clareza a adição e o débito de créditos, em ordem cronológica, e com a data e hora exata da transação. Além disso, a ordem na qual são apresentadas as transações possuem um sentido natural, da transação mais recente para a mais antiga. Outro ponto importante a ser ressaltado é o fato do "Saldo atual" ser apresentado uma única vez e de forma destacada, o que diminui a possibilidade de confusão do usuário quanto ao saldo atual. Na Figura 2.3 é possível essa funcionalidade implementada.

Figura 2.3: Funcionalidade de Saldo e Extrato Implementada no Aplicativo "UFMA Mobile".

Todavia, nos aplicativos "UFV Mobile" e "UFSM Digital" notou-se alguns pontos importantes a serem observados na implementação dessa funcionalidade. No primeiro, há duplicação desnecessária do "Saldo" e não há identificação da operação realizada (crédito ou débito), o que pode confundir o usuário que tenta verificar o extrato. No segundo, a apresentação das transações é feita na ordem anti natural, ou seja, de cima pra baixo (sentido de leitura do usuário) a partir do saldo anterior até o saldo atual, e não o contrário. Segundo [4], um dos pilares da compatibilidade entre o sistema e o mundo real é a disposição dos elementos em ordem natural e lógica. Por não dispor os elementos dessa maneira, o saldo anterior acaba recebendo maior destaque que o saldo atual, e não o contrário. Apesar disso, no mesmo aplicativo é possível destacar um ponto bastante



(a) Saldo e extrato no “UFV Mobile”.

(b) Saldo e extrato no “UFSM Digital”.

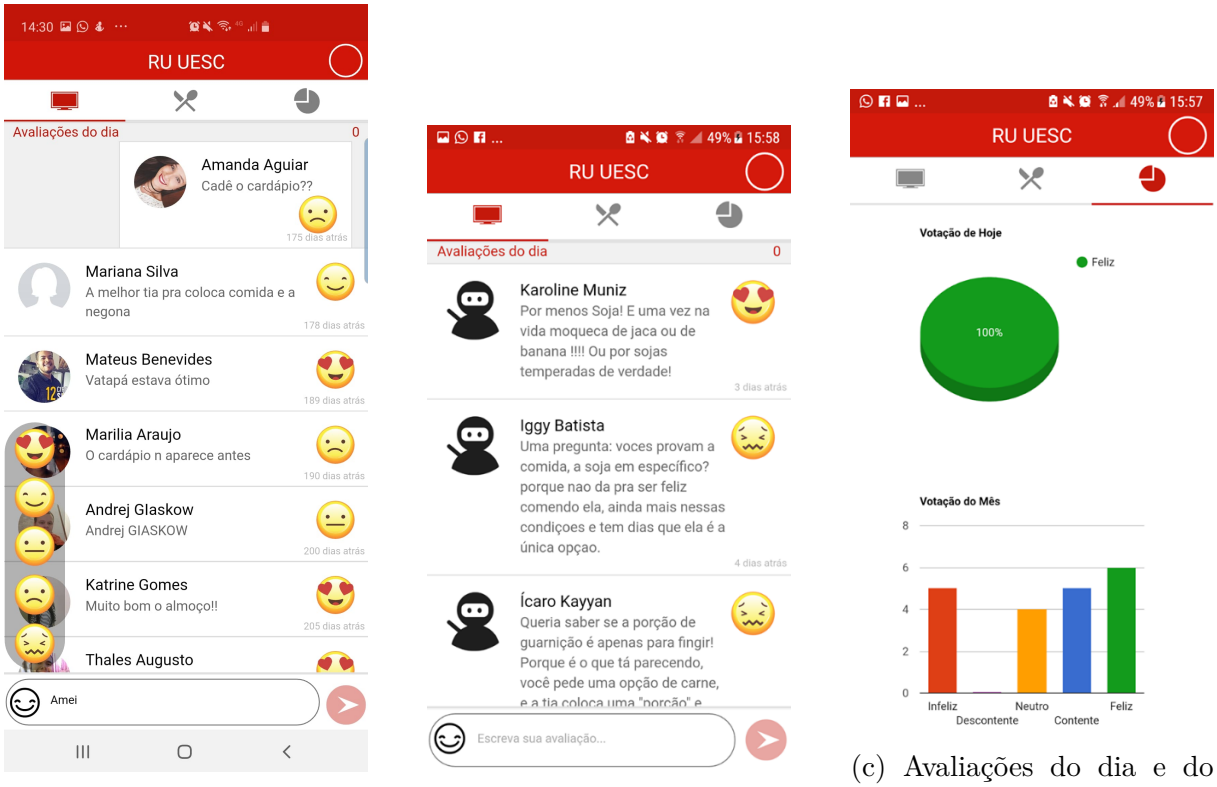
Figura 2.4: Saldo e extrato: exemplos de implementação.

positivo, que é a identificação do restaurante onde a transação foi realizada. A Figura 2.4 apresenta a implementação dessa funcionalidade em ambos os aplicativos.

2.5 Funcionalidade de Avaliação

A avaliação é um mecanismo fundamental nos dias de hoje. A sua importância não está apenas em informar os usuários consumidores, mas também em providenciar *feedback* relativo ao produto ou serviço, e assim fazer com que o mesmo seja melhorado. Assim sendo, sem um mecanismo para medir a reputação de um produto, os consumidores não tem acesso a sua qualidade antes de comprar e consumir [5]. Os consumidores dependem das avaliações para resolverem o problema da assimetria da informação, na qual os vendedores possuem mais informações que os compradores [5].

No nosso contexto, as avaliações podem ser realizadas de diversas maneiras. Por exemplo, o usuário poderia avaliar a comida em um dia específico em forma de enquete ou texto, poderia ser avaliado um prato específico ou até mesmo poderia existir uma avaliação mensal. Existem diversas possibilidade de avaliação. Uma observação importante a ser feita na implementação dessa funcionalidade, é o cuidado para que a avaliação seja



(a) Ação de avaliação. (b) Avaliações individuais. (c) Avaliações do dia e do mês.

Figura 2.5: Funcionalidade de Avaliação no Aplicativo “Cardápio UESC”.

permitida apenas por usuários reais (autenticados) do restaurante universitário, e que não haja duplicações de avaliação por um mesmo usuário.

Dentre os aplicativos citados na seção 2.1, apenas um possui a função de avaliação, que é o aplicativo “RU UESC”. Nele, a avaliação é realizada de maneira textual, em conjunto com uma reação ao nível de satisfação por meio de *emojis*, representando um dos cinco níveis de satisfação: infeliz, descontente, neutro, contente e feliz. Além de avaliar, o usuário pode ver os comentários e as avaliações individuais de cada pessoa. Ademais, o usuário consegue ver a votação do dia em forma de gráfico de setores (em porcentagem), e a votação do mês em forma de gráfico de barras (em números absolutos). Dessa forma, além da ferramenta de avaliação o aplicativo mostra para o usuário, de forma clara e transparente, a avaliação dos outros usuários. Vale ressaltar que o aplicativo mostra as avaliações para os usuários não autenticados, mas não permite a ação de avaliação para eles. Na Figura 2.5 é apresentada essa funcionalidade a partir do aplicativo “RU UESC”.

2.6 Funcionalidade de Notificação

A notificação foi definida por T. Iqbal e E. Horvitz [6] como um sinal visual, auditivo ou alerta háptico gerado por uma aplicação ou serviço que transmite informação para

um usuário fora do foco de atenção do usuário. Nos dias de hoje, as notificações são uma funcionalidade essencial dos telefones móveis. Enquanto em alguns casos o usuário pode tomar uma ação imediata ao ver a notificação, em outros casos as notificações são vastamente ignoradas, dependendo de sua importância para o contexto do usuário [7].

Notificações sobre eventos recebem mais cliques e interação mais rápida [7]. Em nosso contexto, essa afirmação é de extrema importância, tendo em vista que os Restaurantes Universitários possuem diversos eventos em seus calendários, como por exemplo, semana da culinária de países e regiões. Assim, focar na divulgação desses eventos por meio de notificações pode se mostrar um meio de divulgação extremamente eficiente.

Dentre os aplicativos analisados, poucos possuem a funcionalidade de notificação, com destaque para o único que armazena o histórico dessa notificações, que é o aplicativo "Cardápio RU UFRJ". Nele, além do usuário receber a notificação, é possível consultar o histórico das notificações enviadas. Isso é de extrema utilidade, tendo em vista que muitos usuários acabam ignorando as notificações, mas podem desejar vê-las posteriormente na tela com o histórico das notificações. Um ponto negativo do uso das notificações é que elas parecem ser utilizadas apenas para informes a comunidade e não para a divulgação de eventos. Na Figura 2.6 é possível ver a tela de notificação em questão.

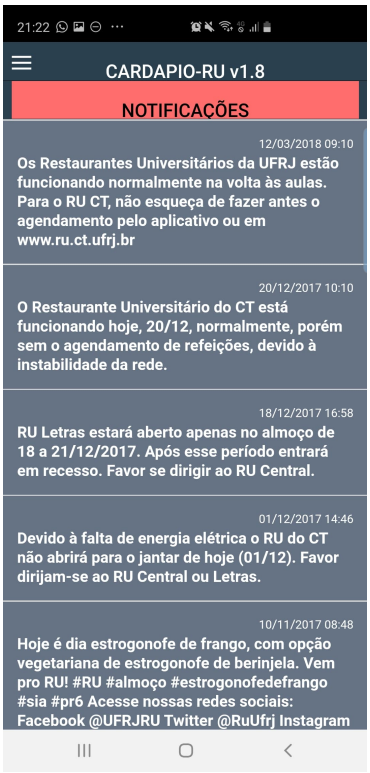


Figura 2.6: Tela de Histórico de Notificação do Aplicativo "Cardápio RU UFRJ".

2.7 Funcionalidade de Compartilhamento do Cardápio

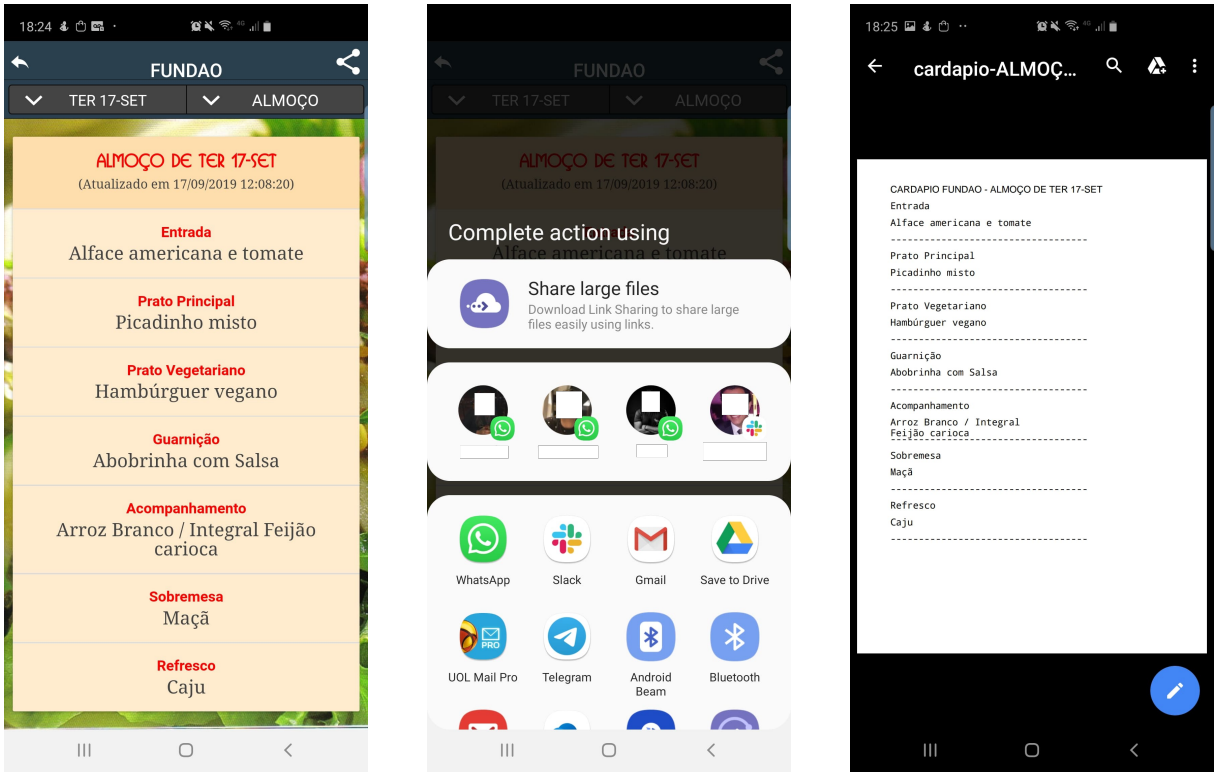
O ato de compartilhar é uma atividade fundamental na chamada *Web 2.0* [8] e, por isso, incontáveis sites possuem algum tipo de "Botão de Compartilhamento". Ainda de acordo com *Nicholas A. John* [8], compartilhar é uma ação de distribuição e disseminação do conteúdo, além de ser um ato de comunicação. Na prática, o compartilhamento para a computação nada mais é do que exportar o conteúdo de uma aplicação para outro aplicação externa. Na maioria dos casos, essas aplicações são as redes sociais, vastamente utilizadas nos dias de hoje e com crescimento cada vez maior. Assim sendo, atualmente há diversos aplicativos móveis que possuem essa funcionalidade para diferentes tarefas, tais como compartilhar as fotos, as notícias, os arquivos, dentre outros.

No contexto de aplicativos relacionados a temática de restaurante universitário, a funcionalidade de compartilhamento está ligada ao cardápio. Uma vez que o usuário possui uma visualização amigável do cardápio, ele pode querer compartilhar isso externamente com amigos por meio de outros aplicativos, como o *WhatsApp* e o *Facebook*. Dentre os aplicativos pesquisados, apenas o aplicativo "Cardápio - UFRJ" possui essa funcionalidade. No aplicativo em questão, o botão de compartilhamento é representado por um ícone característico da ação, diminuindo a possibilidade de confusão na interação do usuário. Neste aplicativo a ação de compartilhamento é realizada em três passos. O usuário clica no botão (ícone) de compartilhamento (1), é apresentado ao usuário os aplicativos sugeridos para o compartilhamento do documento (2), o usuário escolhe com qual aplicativo deseja completar a ação e, dessa forma, o PDF é gerado e compartilhado (3). Na Figura 4.12 é mostrado o passo a passo da ação de compartilhar cardápio.

2.8 Funcionalidade de Compra de Créditos

O pagamento via aplicativo móvel é definido por Au and Koffman [9] como um tipo de pagamento eletrônico no qual ao menos o pagador utiliza a comunicação móvel em conjunto com um dispositivo móvel para iniciar, autorizar e realizar o pagamento. A implementação de pagamento por meio de aplicativos móveis é esperado desde o inicio dos anos 2000, mas foi apenas nos últimos anos que esse serviço ganhou força, principalmente, nos setores de transporte, *fast-food* e bebidas [10]. O pagamento via aplicativo móvel torna o *checkout* mais simples e rápido.

No contexto de restaurante universitário, o pagamento *online* via aplicativo móvel é uma funcionalidade de extrema importância no processo de diminuição das filas que se formam diariamente nos Restaurantes Universitários, sendo assim vital para a melhoria

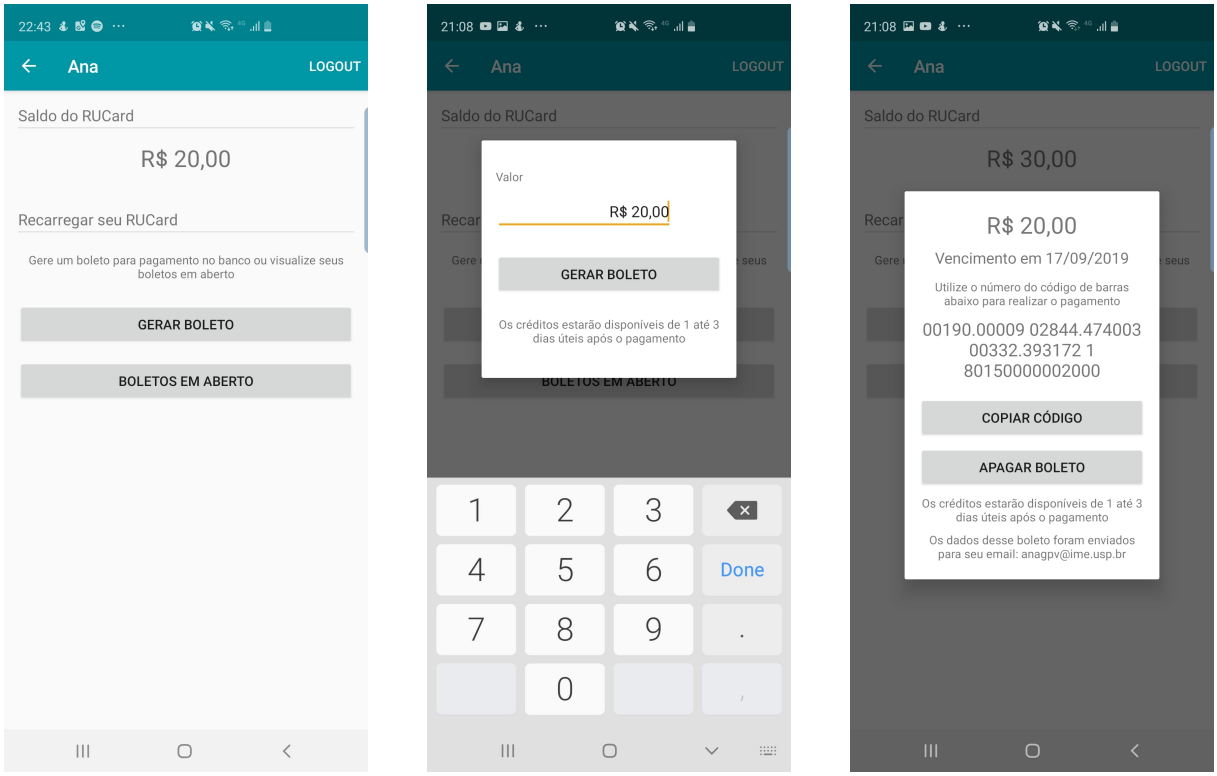


(a) Passo 1 (b) Passo 2 (c) Passo 3

Figura 2.7: Passos para o compartilhamento do Cardápio - App RU da UFRJ

da experiência do usuário do RU. O único aplicativo dentre os pesquisados que implementa essa funcionalidade é o aplicativo "Cardapio" da USP. Nele é possível realizar a compra de créditos apenas via boleto bancário. O usuário clica no botão "Gerar boleto" (1), uma caixa de diálogo é aberta para a definição do valor do boleto (2) e, então, ao clicar no botão "Gerar boleto" desta caixa de diálogo, o boleto é gerado e o código de barras para pagamento é disponibilizado para o usuário (3).

Apesar de ser um grande avanço, o pagamento via boleto bancário não é o ideal para pagamentos *online*, tendo em vista que o tempo de compensação (tempo gastos para o pagamento ser confirmado pelo banco) varia de 1 a 3 dias úteis após a realização do pagamento do boleto. Além disso, o boleto é um meio de pagamento que demanda um tempo maior para o usuário realizar o pagamento, pois além de gerar o boleto o usuário tem de efetivar o seu pagamento em algum banco. O meio mais utilizado, mais rápido e mais eficiente para a realização de pagamento online é o cartão de crédito. Isso ocorre porque utilizando o cartão de crédito, a compensação é quase imediata, ou melhor, a resposta de sucesso ou erro na transação pelo cartão de crédito é quase instantânea. Dessa maneira, em caso de sucesso o usuário já pode utilizar seus créditos no restaurante universitário em alguns segundos.



(a) Passo 1 (b) Passo 2 (c) Passo 3

Figura 2.8: Processo de geração de boleto bancário - App de RU da USP

2.9 Comparativo dos Aplicativos Analisados

Após a analise individual de todos os aplicativos com a temática de RU, disponíveis na loja de aplicativos do *Android*, foi elaborada uma tabela levando em conta todas as funcionalidades presentes e a presença ou não nessas aplicações. Esse comparativo pode ser visto na Tabela 2.1, na qual a presença da funcionalidade é indicada por "SIM", e a ausência pelo espaço em branco.

Assim, analisando a tabela, é possível afirmar que todos os aplicativos foram desenvolvidos com o objetivo inicial de proporcionar aos estudantes um acesso fácil ao cardápio do RU. Essa afirmação se torna possível pois foi observado que todos os aplicativos pesquisados apresentam o cardápio, e a maioria se limita a apenas essa função. Além disso, notou-se que os aplicativos não implementam a maioria das funcionalidades citadas na pesquisa apresentada na seção 2.2, tais como o compartilhamento do cardápio, a avaliação, o extrato e, principalmente, a compra de créditos. Por exemplo, a funcionalidade de compra via aplicativo só é disponível no software "Cardápio USP", sendo que neste a compra só é possível via boleto bancário. Além disso, a funcionalidade "Avaliar" só está implementada no aplicativo "RU UESC".

Tabela 2.1: Tabela de Funcionalidade por Aplicativo

Func. / App	Cardápio	Saldo	Extrato	Compartilhar	Notificar	Avaliar	Comprar
RUUFG	SIM						
Cardápio RU - UFPA	SIM						
BandejãoUFRJ	SIM						
RU UFPA	SIM						
Cardápio RU - UFAC	SIM						
Unipampa	SIM						
Unicamp Serviços	SIM						
UFRGS Mobile	SIM						
Sigaa UFC	SIM	SIM					
Cardápio UFV	SIM	SIM			SIM		
UFMA Mobile	SIM	SIM	SIM				
UFSM Digital	SIM	SIM	SIM				
UFV Mobile	SIM	SIM	SIM				
Cardápio RU - UFRJ	SIM			SIM	SIM		
RU UESC	SIM					SIM	
Cardápio USP	SIM	SIM					SIM

Diante do exposto, o aplicativo desenvolvido neste trabalho, a ser apresentado no Capítulo 4, implementa todas as funcionalidades apresentadas na Tabela 2.1, suprindo os requisitos levantados a partir da pesquisa realizada com os alunos.a

Capítulo 3

Tecnologia Envolvida

O objetivo deste capítulo é apresentar as principais tecnologias envolvidas no desenvolvimento deste projeto. Para cada tecnologia exposta neste capítulo, serão exploradas suas particularidades, características e importância para o desenvolvimento do projeto. Dessa forma, na Seção 3.1 será apresentada a linguagem de programação *Java*, utilizada para o desenvolvimento da aplicação. Na Seção 3.2 será abordado o sistema operacional *Android* e as particularidades do desenvolvimento de aplicações *Android* nativas. Em seguida, na Seção 3.3, serão explorados os conceitos da linguagem de marcação XML e sua utilização para a criação de telas no *Android*. Após isso, na Seção 3.4, serão abordados conceitos importantes dos serviços *web*, com foco na arquitetura REST. Por fim, na Seção 3.5 será apresentada a arquitetura utilizada pelo CPD/UnB (Centro de Informática da UnB) para a disponibilização dos serviços utilizados neste trabalho.

3.1 A Linguagem Java

A linguagem Java é uma linguagem de programação de propósito geral (multi-paradigma), baseado em classes concorrentes e orientadas a objetos. Ela foi criada para ser simples o suficiente para permitir que muitos programadores possam alcançar a fluência na linguagem [11], sem grandes dificuldades. É uma linguagem relacionada ao C++, que por sua vez é descendente da linguagem C, porém com inúmeras diferenças e aspectos dessas linguagens omitidas e, ainda, com algumas ideias de outras linguagens inclusas [12].

A tecnologia *Java* foi criada inicialmente como uma ferramenta de programação em um pequeno projeto denonimado *"the Green Project"*. O projeto, iniciado por Patrick Naughton, Mike Sheridan e James Gosling, foi executado a portas fechadas no verão de 1991. Porém, criar uma nova linguagem nunca foi o objetivo desse projeto [13].

O primeiro projeto do que hoje é o Java foi desenvolvido com o propósito de possibilitar a criação de programas portáveis, ou seja, que fossem executados independente do

processador. Originalmente desenvolvido para a TV Interativa, o projeto foi batizado de *Oak*, uma referência a árvore de carvalho que James Gosling visualizava do seu escritório. Posteriormente, seu nome foi alterado para Green e, consecutivamente, para Java como referência as sementes de café Java produzidas na Indonésia.

Em 1995, a empresa agora chamada Sun Microsystems lançou a primeira versão da linguagem *Java*, o *Java 1.0*. Por apresentar a possibilidade de execução dos códigos nas próprias páginas de programação, algo que era altamente inovador a época, a linguagem foi se tornando popular rapidamente.

A linguagem Java possui seus objetivos de *Design* muito bem definidos. A tecnologia deve proporcionar um desenvolvimento seguro e com um código robusto, ser simples, orientado a objetos, arquiteturalmente neutro e portável, além de executar em alta performance e ser interpretada, *threaded* e dinâmica [14].

Ainda hoje Java é considerada a linguagem de programação mais utilizada do mundo. Segundo o índice da TIOBE [15], Java é usada por *16,66%* dos usuários. Ao se observar o gráfico de evolução de uso da linguagem Java no século, observamos uma certa regularidade nos últimos 10 anos, com pequenas quedas e ascendências [16]. Além disso, é importante observar a grande queda em 2017, que fez com que a *Oracle* anunciasse uma mudança significativa no ciclo de atualizações do Java. Ao invés de soltar uma nova versão a cada 3 anos, o Java passou a ter uma nova *release* a cada 6 meses, similarmente ao que é feito pelas novas linguagens de programação, o que fez com que seu uso voltasse a crescer.

3.1.1 Características Técnicas

A principal, mais notável e inovadora característica da linguagem de programação *Java* é a possibilidade de executar o mesmo código, sem adaptações em sua escrita, em qualquer plataforma que possua a Máquina Virtual Java (*JVM*) instalada. Ainda hoje, a maioria das linguagens de programação apresentam problemas na transferência de uma plataforma para outra, ou seja, muitas vezes o programador tem que efetuar mudanças no código fonte para efetuar a compilação na nova plataforma. O Java por sua vez gera um *bytecode*, ao invés de instruções em linguagem de máquina (que variam para cada arquitetura). Esse *bytecode* gerado é interpretado pela *JVM*, que pode ser instalada em qualquer máquina. Por isso, o Java é considerado uma *Platform-Independent Language*, ou seja, uma linguagem independente de plataforma, na qual o programador escreve o código apenas uma vez e o executa em vários tipos diferentes de computadores e sistemas operacionais, desde que a máquina possua o *Java Runtime Environment (JRE)*, que contém a *JVM*, instalado. A Figura 3.1 ilustra o processo de interpretação da *JVM* em diferentes sistemas operacionais.

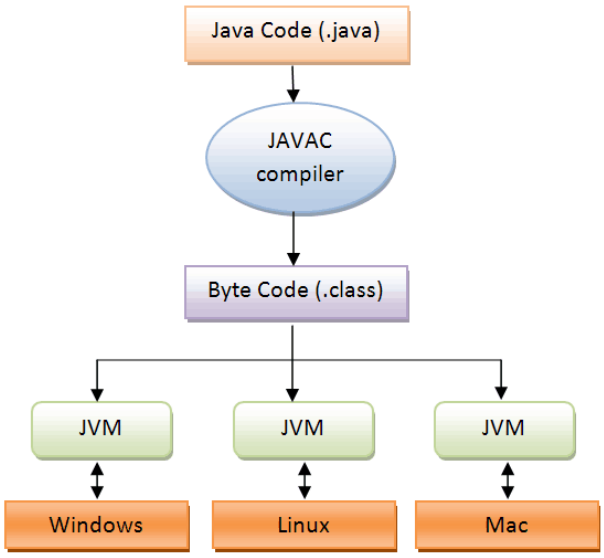


Figura 3.1: *JVM*: Independência de Plataforma.

A Programação Orientada a Objetos (POO) está no cerne da linguagem *Java*. É impossível falar de Java sem antes entender os princípios básicos do paradigma orientado a objetos. Antes de expor os três princípios da Programação Orientada a objetos, é preciso entender o conceito de abstração. A abstração nada mais é do que a maneira na qual os humanos encontraram para gerenciar a complexidade. Por exemplo, quando pensa-se em um carro, as pessoas não pensam em seus milhares de componentes individualmente. O carro é pensado como um objeto bem definido com seu comportamento único. Essa abstração é o que faz com que as pessoas utilizem seus carros sem se preocuparem com a complexidade de cada parte do veículo [12]. Essa mesma ideia pode ser utilizada na construção de uma linguagem de programação.

Encapsulamento, herança e polimorfismo compõem os três princípio de *POO*, e são esses princípios que proporcionam a implementação de um modelo orientado a objetos. Encapsulamento é o mecanismo que conecta os dados e o código que o manipula, mantendo os dois seguros de interferências externas e de mau uso. Uma maneira de se ilustrar o encapsulamento é pensar esse mecanismo como uma embalagem protetiva que previne que o código e os dados sejam arbitrariamente acessados por outro código definido fora dessa embalagem [12]. No *Java* a base do encapsulamento é a *classe*. Uma *classe* define a estrutura e o comportamento (dados e código) de um conjunto de objetos do mundo real. O *objeto*, por sua vez, é uma instância da classe, ou seja, enquanto a *classe* é uma construção lógica, o *objeto* possui realidade física. A *herança* é definida como o processo pelo qual um objeto herda propriedades de outro objeto. Esse conceito é extremamente importante pois torna possível o conceito de classificação hierárquica. Por exemplo, um

poodle é classificado como um cachorro, e cachorros são classificados como animais. Dessa forma, a *classe poodle* pode herdar atributos e métodos de cachorro, que herda de animal. Assim, uma classe pode herdar atributos gerais de classes acima na hierarquia de classes, e necessitam definir apenas aqueles atributos que a tornam única. O polimorfismo, como o próprio nome diz, é uma ferramenta que permite que uma interface seja usada para uma classe geral de ações.

Quando aplicados de forma correta, encapsulamento, herança e polimorfismo são combinados para produzir um ambiente de programação que suporta o desenvolvimento de programas muito mais robustos e escaláveis, que as linguagens de programação procedurais, não seriam capazes de proporcionar. Um ponto importante a ser ressaltado, é o fato de *Java* oferecer suporte apenas para a programação orientada a objetos, enquanto seu antecessor, a linguagem C++, apresenta suporte também para a programação procedural. Isso se deve ao fato de que todos os subprogramas em *Java* são métodos definidos em classes, e esses métodos podem ser chamados somente por meio do uso de classes ou objetos [11].

Como dito no inicio desse capítulo, *Java* é uma linguagem baseada em C++ e criada para superar algumas limitações impostas por essa linguagem. *Java* foi especificamente projetada para manter o poder e a flexibilidade de C++, mas para ser mais simples, menor e mais confiável que sua predecessora [17]. Por exemplo, as expressões, atribuições e instruções de controle são praticamente idênticas em C/C++ e Java, enquanto os vetores, subprogramas e a semântica diferem bastante de uma linguagem para outra. Um *trade-off* bastante importante entre as duas linguagens está no conflito entre confiabilidade e custo de execução. Em *Java*, todas as referências para os índices de um vetor são checadas para garantir que os índices estão no limite definido. Em C/C++, essa checagem não é necessária. Dessa forma, tem-se nos programas C/C++ uma execução mais rápida para o programa semanticamente equivalente de *Java*, e em *Java* tem-se maior confiabilidade em detrimento do custo de execução. Outro conflito bastante comum no *design* de linguagens de programação é o conflito entre capacidade de escrita e confiabilidade. *Java*, contrapondo a família C, optou por não prover a existência de ponteiros, devido a possíveis problemas de confiabilidade no uso de ponteiros, mas seus tipos de referência possuem parte dos recursos dos ponteiros [17].

Outra diferença importante entre *Java* e C++ é que C++ suporta herança múltipla diretamente na definição da classe. Java suporta apenas herança simples, ainda que alguns dos benefícios da herança múltipla possam ser alcançados através do uso de interfaces. *Java* não possui as famosas *structs* e *unions* de seus predecessores C/C++. Além disso, na linguagem *Java* é relativamente fácil criar processos concorrentes, que são chamados de *threads*, por meio de uma forma simples de controle de concorrência utilizando o modi-

ficador *synchronized* [12]. Além disso, *Java* possui um meio de desalocação implícita para seus objetos, que é chamado de *garbage collection*, ou *coleta de lixo*. Essa funcionalidade livra o programador da necessidade de deletar os objetos explicitamente quando eles não são mais necessários [17]. Dessa maneira, tendo como motivação todas as características expostas nesta Seção, o sistema operacional *Android* escolheu a linguagem *Java* como sua primeira linguagem oficial para desenvolvimento de aplicações nativas.

3.2 Android

O *Android* é um sistema operacional (SO), com foco principal em dispositivos móveis, mantido pela empresa *Google*, que comprou o projeto inicial do SO em 2005. O *Android* é um projeto de software *open-source*, isto é, de código aberto, regido por um grupo denominado *Open Handset Alliance*, liderado pela própria *Google* e por gigantes do mercado de dispositivos móveis, com o objetivo de atualizar e melhorar o sistema operacional. Além dos *smartphones*, o *Android* é utilizado por TVs, câmeras e até mesmo refrigeradores. Hoje, o *Android* domina quase 70% do mercado de dispositivos móveis [18] e possui o maior numero de aplicativos disponíveis em sua loja oficial, a *Google Play Store* [19], com mais de dois milhões de aplicativos disponíveis.

O sistema operacional *Android* funciona baseado no *kernel* do *Linux*, o que não significa que seja possível executar um *software Android* em distribuições conhecidas do *linux*, como o *Ubuntu* ou *Debian*, e vice-versa. Isso significa apenas que o *kernel* utilizado pelo *Android*, responsável por gerenciar a memória e os processos do sistema operacional, é o mesmo *kernel* das distribuições *Linux*. Dessa forma, o *kernel* é quem realiza o controle dos aplicativos que estão sendo executados e pode terminar processos para liberar memória caso haja necessidade.

O *Android* possui uma máquina virtual desenvolvida especialmente para dispositivos móveis, a *"Android Runtime"* (ART). Essa máquina foi implementada na versão 4.4 (*KitKat*) do sistema operacional, antes, a máquina virtual *Android* era denominada *Dalvik*. Como principais ferramentas do ART é possível citar o modo de compilação *"ahead-of-time"*, que melhora a performance do aplicativo consideravelmente, uma melhora na funcionalidade de coleta de lixo (*garbage collection*), e uma melhora nas ferramentas de desenvolvimento e *debug*.

Desde sua ideia inicial, o *Android* já passou por diversas versões. Iniciando com a nomenclatura das versões com *Beta* e *Alpha*, o *Android*, a partir de sua versão denominada *Cupcake* (1.5), passa a batizar um conjunto de versões com nome de doces, e sempre seguindo a ordem alfabética. Dessa forma, diversas versões foram lançadas, sendo a última (Android 10), disponível nos primeiros dispositivos a partir do dia 3 de setembro

de 2019, denominada *Q*, quebrando assim a tradição de nomes. Na Figura 3.2 é possível observar, em porcentagem, a cobertura das últimas versões do Android aos dispositivos que utilizam o sistema operacional atualmente.

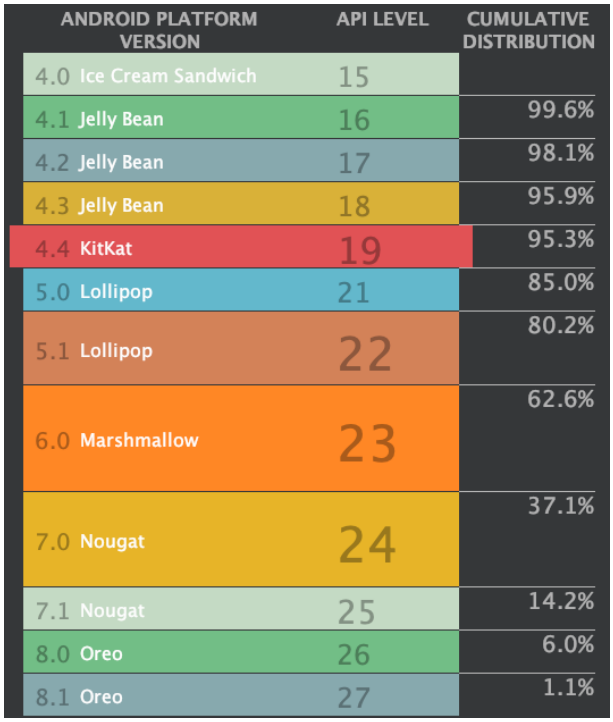


Figura 3.2: Cobertura das versões do *Android* aos dispositivos, em porcentagem.

3.2.1 Arquitetura

Como dito no início desta Seção, o *Android* é uma pilha de software de código aberto, baseado no Linux e que atende diversos dispositivos. A fundação da plataforma *Android* é o *kernel* do *Linux*. Assim sendo, utilizar o *kernel* do *Linux* possibilita ao *Android* tirar vantagem de funcionalidades de segurança chaves dele e permite que os fabricantes de dispositivo desenvolvam *drivers* para um *kernel* já conhecido [1]. A Figura 3.3 ilustra a pilha de software completa do sistema operacional *Android*.

A camada de abstração de hardware (HAL) providencia interfaces padrão que expõe a capacidade dos dispositivos de hardware para a camada mais alta, a *Java API Framework*. Essa camada consiste em diversos módulos, como por exemplo módulos para câmera e *bluetooth*. Para os dispositivos que executam a versão 5 do *Android* (API level 21) ou mais recentes, cada aplicativo roda em um processo diferente e na sua própria instância do *Android Runtime* (ART), camada já explorada neste capítulo. Além disso, é importante ressaltar que o desenvolvimento *Android* nativo suporta o uso de bibliotecas

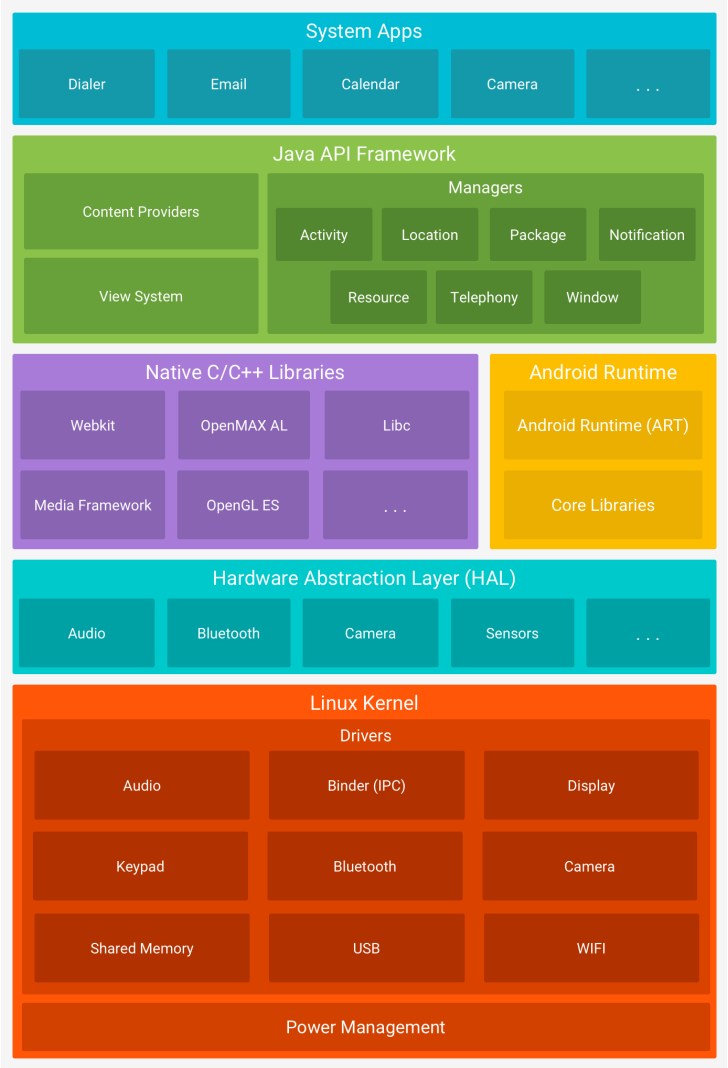


Figura 3.3: Pilha de software *Android* retirada de [1].

nativas escritas em C/C++. Muitos componentes e serviços do sistema *Android*, como o ART e HAL, foram construídos a partir de código nativo que utiliza alguma dessas bibliotecas. Dessa forma, é possível por exemplo utilizar a conhecida biblioteca *Open GL* para desenhar e manipular formas 2D e 3D no seu aplicativo [1].

Para o desenvolvedor, a camada mais importante da arquitetura *Android* é a *Java API Framework*. Todas as funcionalidades do sistema operacional *Android* estão disponíveis para os desenvolvedores através de *APIs* escritas na linguagem *Java*. Essas APIs formam os blocos de construção utilizados para criar aplicativos, simplificando o reuso do núcleo, de componentes modulares do sistema e serviços. Esses componentes e serviços incluem todo o sistema de visualização (listas, grids, botões, etc), os *Content Providers*, que permitem ao aplicativo acessar os dados de outras aplicações, como por exemplo o acesso a galeria de imagens; a *Activity Manager*, que gerencia o ciclo de vida do aplicativo; o

Notification Manager, que permite aos aplicativos apresentarem alertas customizados na barra de status; e o *Resource Manager*, o famoso *R* do *Android*, utilizado para acessar recursos sem código como *strings* declaradas e localizadas, gráficos e arquivos de *layout*.

Por fim, a camada mais alta da arquitetura do sistema é representada pelos aplicativos do sistema. Esses aplicativos são os aplicativos que vem com o dispositivo sem a necessidade de serem instalados, como apps de e-mail, SMS, calendário, contatos, etc. Esses aplicativos não tem *status* especial, e o usuário tem a liberdade de instalar e utilizar outros aplicativos que os substituam.

3.2.2 Desenvolvendo para Android

Um dos componentes cruciais de um aplicativo *Android* é a *Activity*, pois diferentemente dos paradigmas de programação nos quais a aplicação é iniciada no método *main*, o sistema *Android* inicia numa instância de *Activity*, invocando métodos de *callback* correspondentes ao estado específico do ciclo de vida (*lifecycle*) da aplicação. A classe *Activity* serve como ponto de entrada para a interação do aplicativo com o usuário, é ela quem providência a janela na qual o aplicativo desenha a sua *UI* (*User Interface*, ou interface de usuário). A maioria dos aplicativos possuem múltiplas telas, o que significa que eles possuem múltiplas *activities*. Além disso, comumente um aplicativo inicia *activities* não só do próprio aplicativo, mas também pertencentes a outros aplicativos. Por exemplo, um aplicativo web pode iniciar a *Activity* de compartilhamento de um *app* de rede social. Assim, a classe *Activity* provê *callbacks* que permitem ao desenvolvedor saber que o estado mudou, ou seja, que aquela tela está sendo criada, parada, resumida ou destruída no processo ao qual aquela atividade pertence [2].

Dessa forma, um ponto central no desenvolvimento de aplicativos *Android* é entender o ciclo de vida das classes *activities* e seus respectivos *callbacks*. Para navegar entre os estados do ciclo de vida, a classe *Activity* fornece seis *callbacks*: *onCreate(), onStart(), onResume(), onPause(), onStop()* e *onDestroy()*. O sistema invoca cada um desses metódos de *callback* quando a *Activity* entra em um novo estado. A Figura 3.4 ilustra esse processo, destacando os acontecimentos que ocasionam a transição de um estado da *Activity* para outro. Quando o usuário começa a sair da *Activity*, o sistema chama alguns métodos para desmantelar a *Activity*. Em alguns casos esse desmantelamento é apenas parcial e a *Activity* continua em memória (como por exemplo quando o usuário muda a visualização para outro *app*), e ele ainda pode retornar para o *app* sem precisar reconstruir aquela *Activity*. Assim, dependendo da complexidade da *Activity*, o desenvolvedor provavelmente não precisa implementar todos os métodos do ciclo de vida. No entanto, entender cada um deles é imprescindível para garantir que sua aplicação funcione da maneira desejada.

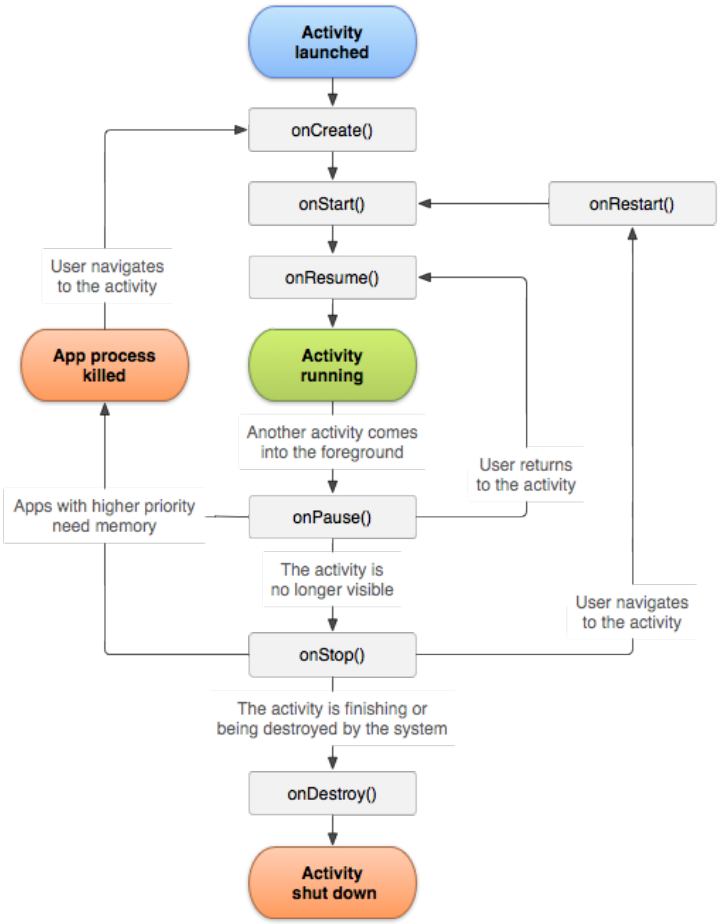


Figura 3.4: Ciclo de vida de uma *Activity* no Android [2].

Outro ponto central no desenvolvimento de aplicações nos dias de hoje é o *kit* de desenvolvimento de software (*Software Development Kit*, ou *SDK*). Um *kit* de desenvolvimento é um conjunto de ferramentas para o desenvolvimento de uma aplicação móvel. O *Android SDK* é o kit de desenvolvimento de software para *Android*, o qual permite aos desenvolvedores criarem aplicativos para a plataforma *Android* de forma nativa. Todavia, embora as ferramentas do *SDK* possam ser usadas por linha de comando, a maneira mais comum é fazer uso de um ambiente de desenvolvimento integrado (IDE). Desde seu lançamento em 2013 [20], a IDE recomendada pela *Google* para o desenvolvimento de aplicações *Android* nativas é o *Android Studio*. O *Android Studio* veio para substituir o *Eclipse Android Development Tools* (ADT) como a IDE primária do *Google* para o desenvolvimento nativo para *Android*. Esse ambiente de desenvolvimento é gratuito e disponível para os três principais sistemas operacionais (*Windows, Linux e MacOS*), e foi criado pela própria *Google* baseado na interface de desenvolvimento *IntelliJ IDEA* [21].

O *Android Studio* facilita a codificação do aplicativo, assim como a integração de

plugins e ferramentas. O *SDK Tools*, que já vem instalado com o pacote inicial do *Android SDK*, disponibiliza ferramentas como o *Android SDK Manager*, *AVD Manager*, o emulador e o *Dalvik Debug Monitor Server*. O *AVD Manager (Android Virtual Device Manager)* fornece uma interface gráfica na qual é possível criar e gerenciar dispositivos virtuais *Android* que são executados no emulador. Dessa forma, ele permite que o desenvolvedor consiga simular a execução do aplicativo em diferentes dispositivos, como celulares, *tablets* ou TVs, em diferentes níveis de API e em diferentes níveis de resolução para um mesmo dispositivo. Por exemplo, o desenvolvedor pode simular a execução do aplicativo em diversos modelos de celulares, como mostrado na Figura 3.5.

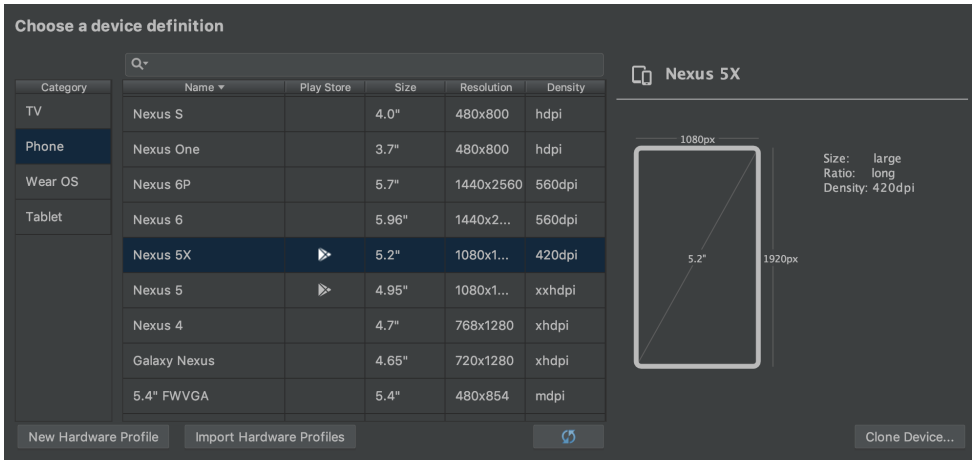


Figura 3.5: Criação de um Novo Dispositivo Virtual no *AVD Manager* do *Android Studio*.

Por fim, ressalta-se a importância da existência de um emulador rápido e eficaz dentro da própria *IDE*, que permite ao usuário testar seu aplicativo com agilidade e replicabilidade. Além disso, o *Android SDK Manager* disponibiliza uma maneira simples de gerenciar as ferramentas, plataformas e outros componentes necessário no desenvolvimento da aplicação. Assim, o *Android Studio* se tornou uma opção robusta, ágil e segura para o desenvolvimento de aplicativos *Android* nativos.

3.3 XML e Layout no Android

O XML (*eXtensible Markup Language*) é uma linguagem de meta marcação extensível usada para definir regras por meio de um documento, tal que esse documento seja legível para humanos e possa ser processado por computadores. Essa linguagem de meta marcação de texto é um dos subtipos da SGML (*Standard Generalized Markup Language*) e foi desenvolvido pelo *World Wide Web Consortium* (W3C) em 1996 [22]. O XML surge para suprir algumas necessidades que as aplicações necessitavam e que o HTML não re-

solvia, como por exemplo a troca de informação entre dispositivos. Dessa maneira, como o próprio significado de XML sugere (extensível), o objetivo principal da linguagem XML foi fornecer aos desenvolvedores uma maneira de definir e criar seus próprios marcadores e atributos em vez de estarem restritos ao esquema de marcação da HTML [23].

Segundo Furgeri [23], algumas das principais características e usos da linguagem XML são:

- É um importante recurso para criação de metadados, que, por sua vez, são recursos de vital importância para a recuperação de informações;
- Permite que um documento seja representado como uma estrutura padronizada, em forma de árvore, tornando possível para um *software* realizar uma varredura no documento com a finalidade de recuperar informações. Dessa facilidade derivam diversas aplicações da XML;
- Por ser uma linguagem flexível, na qual o criador pode elaborar suas próprias *tags*, diversas outras linguagens de marcação foram desenvolvidas a partir da XML;
- Pode ser utilizada para criação de ontologias entre comunidades. Cada tipo de documento pode possuir termos que representam conceitos específicos às necessidades de informação da área;
- A linguagem XML facilita de forma significativa a troca de dados. Por meio da XML, computadores em diferentes plataformas, sistemas e banco de dados podem trocar dados pela internet. Dessa forma, basta converter o formato do banco de dados em texto seguindo as regras dos documentos XML;
- Por permitir a definição dos marcadores, os dados armazenados têm significado, permitindo linguagens, como o *Java*, manipular esse conteúdo, de modo a facilitar a pesquisa, a inclusão, a alteração e a exclusão de dados.

No desenvolvimento *Android* nativo, um *layout* pode ser declarado de duas maneiras distintas. O desenvolvedor pode instanciar os elementos do *layout* de maneira dinâmica, ou seja, em tempo de execução. Normalmente, o desenvolvedor *Android* utiliza essa estratégia apenas quando realmente se faz necessário apresentar um componente de *layout* em tempo de execução. A segunda forma e mais comum, é declarar os elementos da interface de usuário (IU) em XML. O *Android* providencia um vocabulário XML próprio e direto, que corresponde às classes e subclasses de visualização. Dessa forma, ao declarar a interface de usuário, o desenvolvedor pode separar a apresentação do aplicativo do código que controla o comportamento dele [24].

Assim, por meio do uso do vocabulário XML do *Android*, o desenvolvedor pode projetar *layouts* de maneira rápida e similar a criação de páginas *Web* em HTML. Cada

arquivo de *layout* deve conter um único elemento raiz, que deve ser um objeto *View* ou *ViewGroup*. Assim, define-se objetos ou *widgets* de *layout* extras como elementos filho desse elemento raiz, de modo a construir gradualmente uma hierarquia de *View* que defina o *layout*. A Figura 3.6 ilustra parte da declaração XML da tela de menu do aplicativo proposto neste trabalho, e o resultado do XML na interface de usuário. É importante observar que o nome das *tags* definidas no documento XML correspondem aos nomes dos componentes do *Android* utilizados no código propriamente dito, como *LinearLayout*, *ScrollView*, *TextView*, dentre outros.

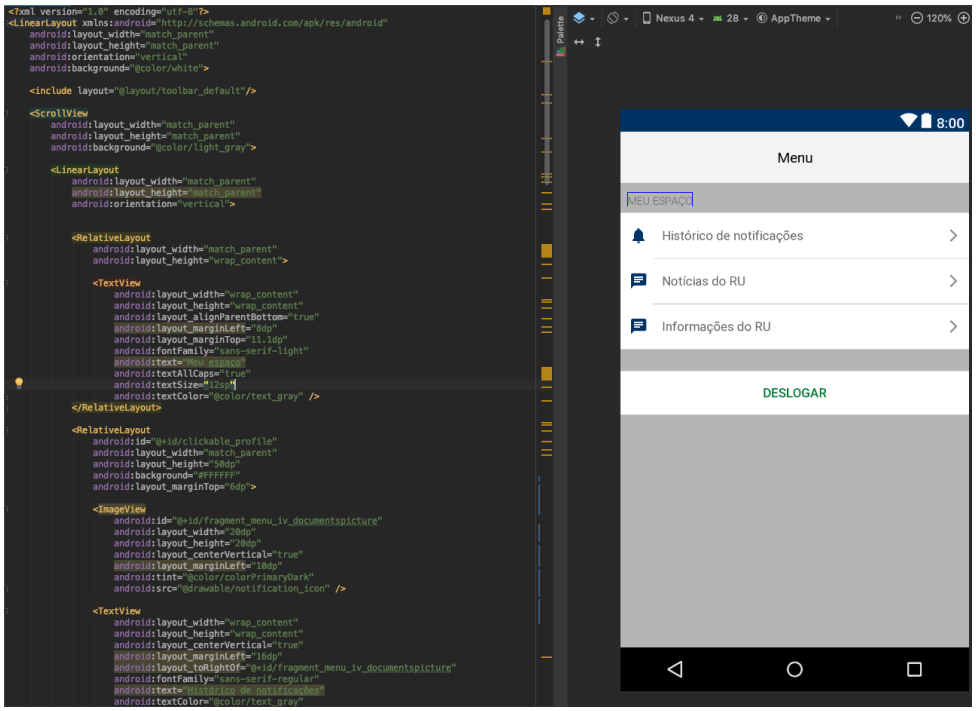


Figura 3.6: XML e Interface da Tela de Menu do Aplicativo RUnB.

3.4 Web Service

Os *Web Services* são componentes que permitem às aplicações realizarem troca de informação, ou seja, esses serviços tornam possível que sistemas diferentes possam enviar e receber dados através da *Web*. Por meio da utilização de protocolos de comunicação como o HTTP, as aplicações trocam informações fazendo uso de uma linguagem intermediária como o XML (mencionado na Seção 3.3) ou JSON, independente da linguagem utilizada no sistema. A W3C define *Web Service* como um sistema de software projetado para suportar a interoperabilidade entre máquinas através da rede. Alguns benefícios da utilização de *Web Services* são:

- Facilitam a integração de sistemas, uma vez que é possível trocar informação entre duas aplicações, sem necessidade de recolher informação detalhada sobre o funcionamento de cada sistema, como o sistema operacional ou a linguagem de programação;
- Redução no tempo de desenvolvimento do software, devido ao fato da reutilização de um serviço por aplicações distintas;
- Maior segurança, tendo em vista que o *Web Service* evita a comunicação direta da aplicação com a base de dados;
- Redução de custo, pois com o uso de serviços web se torna desnecessária a criação de aplicações para a integração de dados. Os *Web Services* tiram proveito de protocolos e da infraestrutura Web já existente, requerendo um menor investimento.

3.4.1 Arquitetura REST

Existem alguns padrões de *design* (ou arquitetura) na construção de *Web Services*. Dois dos mais conhecidos são o REST (*Representational State Transfer*) e o SOAP (*Simple Object Access Protocol*). Nos últimos anos, o modelo REST vem substituindo o SOAP na implementação de *Web Services*, e hoje é maioria absoluta no mercado de *Web Services* e APIs (*Application Programming Interface*) [25]. Isso se deve principalmente à facilidade de implementação e flexibilidade na implementação do padrão REST, que pode seguir diferentes protocolos de comunicação e linguagens de formatação de dados. O SOAP, por sua vez, requer por padrão a utilização do protocolo HTTP com XML serializado.

O REST foi proposto por Roy Fielding em 2000 em sua tese de doutorado, e é definido por seu criador como um estilo de arquitetura híbrido, proveniente da combinação de diferentes estilos arquitetônicos baseados em rede [26]. O REST utiliza o protocolo HTTP (*Hyper Text Transfer Protocol*) para realizar as trocas de mensagens, e podem usar as notações JSON (*JavaScript Object Notation*) ou XML.

A abstração chave do REST é o recurso. Qualquer informação que possa ser nomeada é potencialmente um recurso, como por exemplo um documento ou uma imagem. O REST utiliza um identificador único para cada recurso, chamado de *Uniform Resource Identifier* (URI). A troca de informações que ocorre por meio do protocolo HTTP possui uma semântica especifica para diferentes tipos de operações, as quais são denominadas verbos do protocolo HTTP:

- POST, utilizado quando o servidor aceita que a aplicação envie dados no corpo da requisição, normalmente para salva-los;
- GET, utilizado para recuperar informações;

- PUT, utilizado para atualizar os dados de um recurso específico;
- DELETE, utilizado para apagar os dados de um recurso específico;
- HEADER, usado para recuperar os metadados de uma representação do recurso;
- OPTIONS, utilizado para obter a descrição ou a documentação sobre o recurso desejado.

Como dito anteriormente, o padrão arquitetural REST faz uso dos formatos JSON e XML para intercâmbio de dados. O JSON, por ser uma alternativa mais simples e mais leve ao XML, é o formato utilizado preferencialmente neste tipo de arquitetura [27]. O formato JSON é derivado da linguagem de programação *JavaScript*, mas hoje a absoluta maioria das linguagens orientadas a objeto incluem suporte para gerar, analisar e editar sintaticamente dados em formato JSON, além de converter para objetos da linguagem. Um exemplo de um documento JSON pode ser visto na Figura 3.7.

```
{
  "product": {
    "name": "Produto Y",
    "category": "1",
    "price": "2030"
  },
  "event_id": "886"
}
```

Figura 3.7: Exemplo de Documento JSON.

3.5 A Arquitetura de Serviços da UnB

A maioria dos serviços utilizados neste trabalho foram desenvolvidos e disponibilizados pelo CPD/UnB, utilizando o modelo de arquitetura proposto em [28]. Nele, propõe-se a utilização de uma Arquitetura Orientada a Serviço (*Service Oriented Architecture*, ou SOA), particularmente seguindo o estilo arquitetural REST, o qual foi abordado na seção anterior. Além disso, faz-se o uso de um *Enterprise Service Bus* (ESB) desenvolvido na linguagem funcional Erlang, disponível no site `https://github.com/erlangms`.

O barramento suporta a mediação, roteamento, transformação de dados e a orquestração dos serviços [28]. O formato escolhido para o envio e recebimento de mensagens do cliente foi o JSON, já abordado na seção anterior, que foi escolhido com o objetivo de facilitar a implementação do barramento [28].

O esquema de comunicação da arquitetura se da por meio de duas vias. Na primeira, o cliente se comunica, por meio de uma interface REST, com o barramento afim de consumir algum serviço. O uso da interface REST obriga o cliente (que pode ser qualquer sistema, como o aplicativo proposto neste trabalho) a suportar chamadas de serviços em REST [28]. Na segunda via, acontece a comunicação do barramento com o serviço. O barramento ErlangMS é o responsável por rotear as requisições REST para determinado serviço processar e devolver o resultado de volta para o cliente. O esquema de roteamento é das mensagens é ilustrado na Figura 3.8.

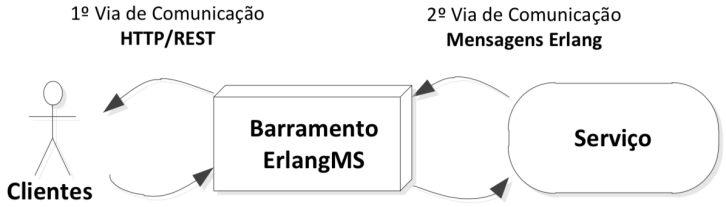


Figura 3.8: Esquema de Roteamento de Mensagens na Arquitetura utilizada pelo CPD/UnB.

É importante ressaltar que nessa arquitetura a gerencia da execução do serviço é realizada pelo barramento, que notifica o cliente responsável pela requisição quando ela estiver concluída. Nos sistemas da UnB que ainda utilizam a arquitetura antiga, o sucesso ou falha na execução de uma rotina é verificada manualmente por meio de consultas diretas as tabelas do banco de dados.

3.6 Considerações Finais

Neste capítulo foi apresentado, de forma geral, o conjunto de tecnologias envolvidas no desenvolvimento da aplicação que é objeto deste trabalho. Assim, na primeira Seção foram abordados os conceito da linguagem de programação *Java* e suas características técnicas, além de apresentar os conceitos do paradigma orientado a objetos. Em seguida, foi apresentado o sistema operacional *Android*, sua arquitetura e as nuances do desenvolvimento de aplicações *Android* nativas. Logo após, apresentou-se brevemente os conceitos da linguagem de marcação XML, seus conceitos e importância para a construção de *layouts* no *Android*. Além disso, discorreu-se sobre os conceitos, a importância e as características dos *Web Services*, enfatizando a arquitetura REST na implementação desses serviços. Por fim, foi apresentado o padrão arquitetural da UnB para a disponibilização de *Web Services*.

Capítulo 4

RUnB - Aplicativo para o RU da UnB

Este capítulo apresenta o aplicativo RUnB, que é um aplicativo desenvolvido para o Restaurante Universitário da UnB. Para tal, a Seção 1 apresenta uma visão geral do Aplicativo. A Seção 2 discorre sobre o Restaurante Universitário da Universidade de Brasília, sua importância e seus principais problemas. A Seção 3 expõe o levantamento de requisitos, imprescindível para entender as reais necessidades dos usuários. A Seção 4 apresenta a implementação das telas e as funcionalidades do aplicativo RUnB. A Seção 5 apresenta os resultados dos testes realizados por usuários do aplicativo RUnB. Por fim, a Seção 6 apresenta as considerações finais do capítulo.

4.1 Visão Geral

As grandes filas seguem sendo o principal problema dos restaurantes universitários no Brasil [29], e na Universidade de Brasília a realidade não é diferente. Devido ao preço reduzido e a consequente alta demanda, imensas filas se formam diariamente, principalmente nos caixas onde se compram os créditos. Assim sendo, existe uma demanda natural dos usuários do Restaurante Universitário por uma solução para o problema das filas, ou a diminuição dele, tendo em vista que a gestão do tempo é um fator cada vez mais valorizado no mundo moderno.

Dessa forma, o aplicativo RUnB proposto neste trabalho, tem como objetivo principal melhorar a experiência da comunidade acadêmica da Universidade de Brasília ao utilizar o Restaurante Universitário, principalmente no que diz respeito as grandes filas que se formam. Além disso, o RUnB possibilita a democratização do acesso a informações até então omitidas ou de difícil acesso. O aplicativo oferece ao usuário a visualização rápida do cardápio do dia das cinco unidades (Darcy Ribeiro, Gama, Ceilândia, Planaltina e

Fazenda). Ademais, o usuário, através do *login* com suas informações de matrícula e senha registradas junto à UnB, pode ter acesso ao seu saldo, extrato e pode realizar a compra de créditos para o seu cartão RU através de cartão de crédito. Outrossim, o aplicativo possibilita ao usuário o acesso aos refeitórios do RU por meio da geração de um código de barras, o que pode substituir o uso da carteirinha. A compra de crédito via aplicativo móvel e o acesso ao refeitório via aplicativo (código de barras) são duas funcionalidades que contribuem de forma significativa para a diminuição das filas dos refeitórios, e que diferenciam este aplicativo dos demais apresentados no Capítulo 2.

Dessa forma, as principais funcionalidades do aplicativo RUnB são:

- Apresentação do cardápio das cinco unidades do Restaurante Universitário da UnB;
- Login, por meio do seu registro acadêmico (matrícula no caso dos alunos) e sua senha do *Matrícula Web*;
- Visualização do saldo do cartão do RU, do grupo de usuário e do valor da refeição para esse grupo;
- Visualização do extrato (movimentação de valores) do cartão do RU;
- Geração do código de barras, a fim de possibilitar a entrada nos refeitórios sem a apresentação da carteirinha;
- Compartilhamento externo do cardápio;
- Acesso rápido à pagina de notícias do Restaurante Universitário;
- Avaliação do RU e especificamente das refeições;
- Notificação (*push notification*) enviada aos usuários.

O aplicativo foi desenvolvido para o sistema operacional *Android* e é compatível com todas as versões 5 (*SDK* 19, Lollipop) ou superiores. Para o correto funcionamento de todas as funcionalidades do aplicativo, é necessário que o dispositivo possua uma conexão ativa com a internet, tendo em vista que o aplicativo depende da conexão com alguns *Web Services*. Além disso, é importante ressaltar que a maioria dos *Web Services* utilizados nesse aplicativo foram disponibilizados pelo CPD/UnB.

Com o objetivo de criar uma identidade visual e estabelecer uma conexão com o usuário através de formas, símbolos e cores, desenvolveu-se a logomarca do aplicativo RUnB, assim como suas formas de utilização. Para tanto, foi utilizada a paleta de cores presente no manual de identidade visual da UnB [30] em conjunto com os traços arquitetônicos do restaurante universitário. Ademais, foram utilizados elementos que remetem aos serviços e produtos ofertados pelo RU, como o chapéu de cozinha, o qual faz referência ao

serviço de alimentação prestado. Os irreverentes traços do arquiteto José Galbinski [31], característicos do RU Darcy Ribeiro, destacam-se na versão completa da logomarca. As diferentes versões de utilização da logo podem ser vistas na Figura 4.1.

Figura 4.1: Versões de Utilização da Logomarca RUnB.

4.2 O Restaurante Universitário da Universidade de Brasília

Na década de 60, quando o Restaurante Universitário (RU) iniciou suas atividades, o objetivo era atender os funcionários, os professores e suas famílias. Apenas em 1962, com as reivindicações dos alunos, é que se passou a atendê-los [32]. O propósito do Restaurante Universitário é fornecer refeição de baixo custo, balanceada e saudável à comunidade da Universidade de Brasília, visando apoiar o desenvolvimento de atividades de ensino, pesquisa e extensão, minimizando a evasão e favorecendo a diplomação [33].

O RU da Universidade de Brasília possui cinco unidades: Darcy Ribeiro, Gama, Ceilândia, Planaltina e Fazenda. A unidade que serve o maior número de refeições é a do campus Darcy Ribeiro, a qual pode ser vista na Figura 4.2. Segundo dados da direção, o RU serve mais de 2 milhões de refeições por ano, tendo como base os anos de 2016, 2017 e 2018. O valor das refeições do RU varia de acordo com o grupo do aluno, sendo

os usuários grupo 1 composto pelos alunos do PNAES (Plano Nacional de Assistência Estudantil) e isentos do pagamento das refeições. Essa informação é importante, pois esses alunos não podem realizar a compra de créditos, incluindo a compra por aplicativo, tendo em vista que são isentos da cobrança de qualquer valor. O quantitativo de refeições realizadas no restaurante de 2016 a novembro de 2019 pode ser conferido na Tabela 4.1.

Figura 4.2: Restaurante Universitário do Campus Darcy Ribeiro.

Tabela 4.1: Número de Refeições nos Últimos Anos

Ano / Grupo	2016	2017	2018	2019
PNAES	551.387	588.549	645.253	551.193
Recursos Próprios	1.595.543	1.841.134	1.444.087	920.032
TOTAL	2.146.930	2.429.683	2.089.340	1.471.225

Mesmo que ainda haja muito para evoluir, o RU já melhorou bastante em relação à tecnologia. Hoje os cardápios são disponibilizados em formato PDF no site do restaurante [33], o que facilitou consideravelmente o acesso dos alunos a essa informação. Antes disso, os alunos eram obrigados a ir fisicamente ao RU com o objetivo de descobrir o cardápio do dia.

Segundo o relatório de gestão do restaurante universitário de 2008, o sistema utilizado pelo RU apresentava diversos problemas, principalmente relacionados a inconsistências nos relatórios financeiros e a fraudes. Dessa forma, visando corrigir esses problemas, a Auditoria da Universidade de Brasília orientou a suspensão do sistema e a realização manual do controle financeiro até que outro sistema fosse implantado. Assim, o controle

manual passou a ser feito e fichas impressas passaram a ser emitidas para realizar o acesso dos alunos ao refeitório.

Por conta disso, o RU solicitou ao CPD a correção dos erros no sistema, e ficou decidido que um novo sistema seria desenvolvido. Assim surgiu o SISRU (Sistema do Restaurante Universitário), um sistema automatizado de controle de acessos de usuários ao restaurante que começou a ser desenvolvido em 2010. O sistema foi desenvolvido pelo CPD/UnB com o objetivo de controlar o número de refeições servidas, por grupo de preço e por tipo de refeição, dificultando fraudes, as quais costumavam ser recorrentes antes da existência do sistema. Assim, por meio do SISRU a administração do RU passou a ter controle de toda a movimentação financeira, do extrato dos alunos, de seus grupos, e realizar vendas de forma segura e controlada, dentre outras funcionalidades. Segundo a direção do restaurante, desde 2012, ano em que foi entregue e passou a ser utilizado o SISRU, problemas de inconsistência deixaram de existir e fraudes passarem a ser muito menos frequentes.

Contudo, a implantação do SISRU resolveu o problema da informatização na parte administrativa e financeiro do RU. Atualmente, o grande problema da informatização do RU segue sendo o acesso dos alunos a informações primordiais para o dia a dia do usuário do restaurante, e a falta da facilitação na realização de algumas funções, como a compra de créditos. Hoje os alunos tem acesso apenas à informação de saldo do RU, através do MatriculaWeb (sistema *online* de matrícula da UnB), o qual demanda abertura do endereço do MatriculaWeb em um *browser* e o *login* no sistema, conforme apresentado na Figura 4.3. Em 2015, um grupo de alunos desenvolveu o RUMOR, aplicativo que exibia o cardápio do RU, mas com o passar do tempo o aplicativo acabou sendo descontinuado.

Figura 4.3: Informação de Saldo no MatriculaWeb.

Informações como o extrato do aluno e o grupo pertencente, apesar de estarem disponíveis aos administradores do RU, não são disponibilizadas aos usuários. Além disso, para realizar a compra de créditos do RU o estudante necessita se locomover fisicamente aos caixas do RU, resultando assim na formação de grandes filas diariamente. Dessa forma, o aplicativo RUnB surge para facilitar a realização de atividades cotidianas do usuário do RU, como a compra de créditos e o acesso ao refeitório, e para suprir a necessidade do acesso a informação, a partir de uma ferramenta que permite a aproximação da comunidade universitária da UnB com o restaurante universitário.

Diante do exposto, notou-se a necessidade evidente de desenvolver um aplicativo que pudesses satisfazer as principais necessidades dos usuários do RU. Para isso, adotou-se uma metodologia para o levantamento de requisitos, que será descrita na próxima seção.

4.3 Levantamento de requisitos

Para que haja um entendimento completo das reais necessidades dos usuários de um sistema é necessário levantar requisitos funcionais e não-funcionais [34]. Tendo isso como objetivo, foi elaborada uma pesquisa com a comunidade acadêmica da Universidade de Brasília. Para que a pesquisa atingisse o maior número possível de pessoas, o questionário foi elaborado utilizando o *Google Forms* e divulgado para a comunidade acadêmica, principalmente para os estudantes. Dessa forma, foram obtidas 267 participações na pesquisa, o que representa uma participação bastante satisfatória da comunidade acadêmica.

As três primeiras perguntas do questionário dizem respeito ao perfil dos entrevistados. Primeiro, foi perguntado a respeito da faixa etária, e o resultado mostrou que mais de 50% dos entrevistados possuem idade entre 19 a 21 anos. A segunda pergunta indagou aos entrevistados sobre sua residência, e os resultados foram diversos, com destaque para a Asa Norte com 13.5%. Por fim, indagou-se quanto a função junto à universidade: aluno ou ex-aluno, professor, servidor ou sem vínculo com a UnB. Ao se escolher a opção "Não tenho vínculo com a UnB", o questionário se encerrava, devido ao fato que nesse caso a pesquisa não se aplicava. O resultado dessa pergunta mostrou que quase 100% dos entrevistados (99.6%) são alunos ou ex-alunos da UnB. O resultado completo dessas perguntas pode ser conferido na Figura 4.4.

As duas próximas perguntas do formulário estão relacionadas ao uso de aplicativos móveis e o sistema operacional predominante entre os entrevistados. A pergunta "Quanto tempo você gasta com aplicativos móveis por dia, aproximadamente", revelou que mais da metade dos entrevistados dizem passar mais de três horas por dia em aplicativos móveis. O questionamento relativo ao sistema operacional dos entrevistados expôs que 76.3% deles utilizam o *Android* como sistema operacional, mais que o triplo dos usuários do *iOS*, conforme apresentado na Figura 4.5. Dessa forma, esse resultado suporta a escolha do sistema operacional *Android* para o desenvolvimento da aplicação presente neste trabalho. O resultado dessas perguntas perguntas é apresentado na Figura 4.5.

A pergunta "Você frequenta ou já frequentou o Restaurante Universitário da UnB?" foi inserida com o objetivo de filtrar os interrogados que nunca frequentaram o RU. Dessa forma, os entrevistados que responderam negativamente essa pergunta tinham seus questionários encerrados, enquanto os demais prosseguiam para as próximas questões. Dessa

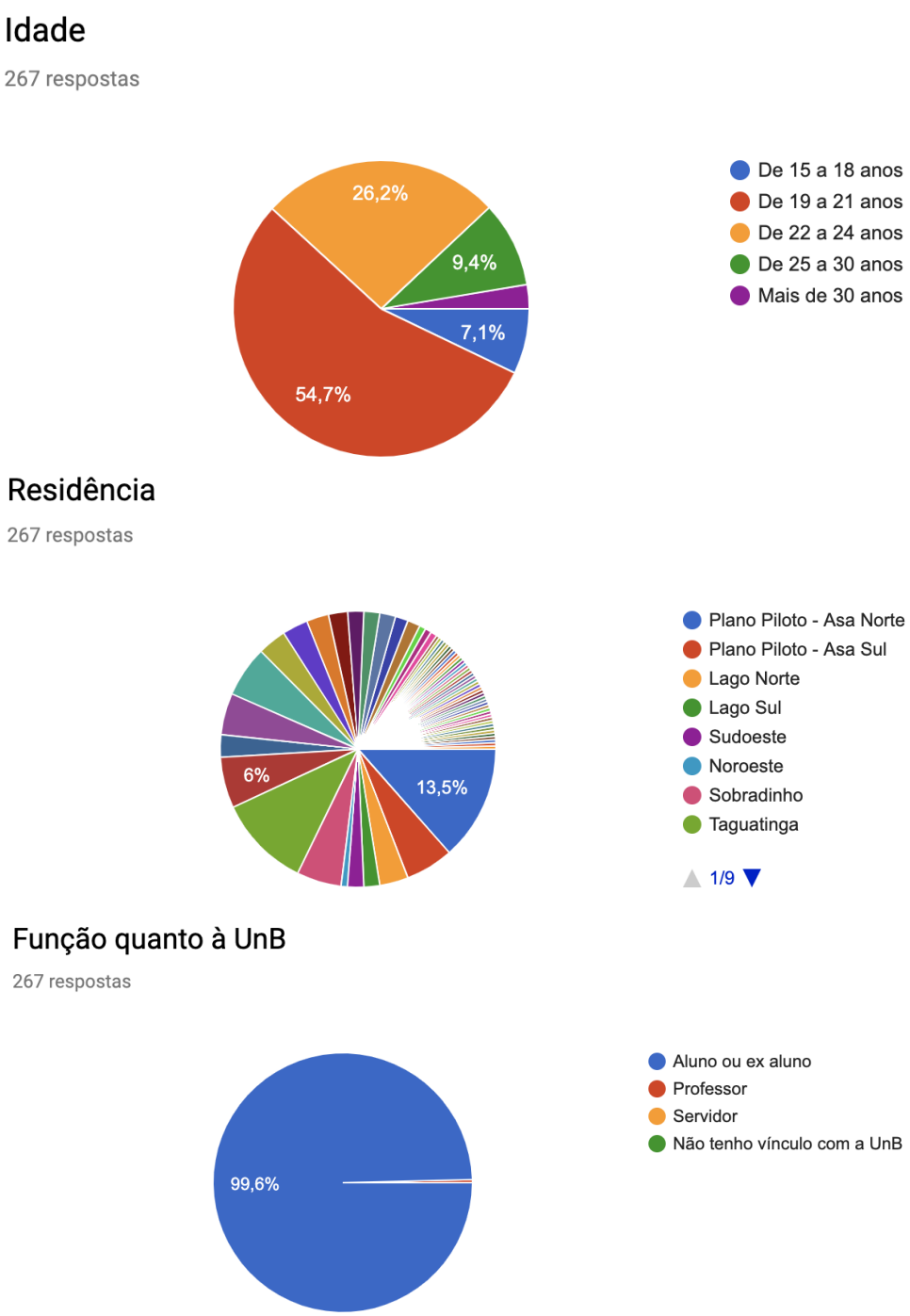


Figura 4.4: Respostas às perguntas relativas ao perfil do entrevistado.

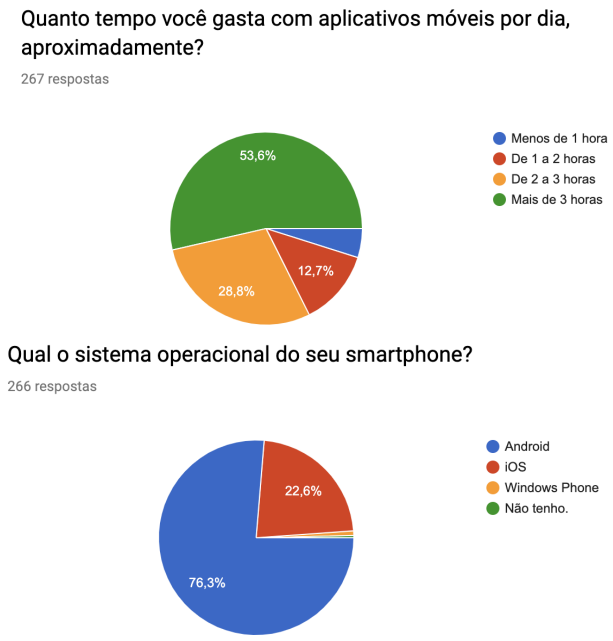


Figura 4.5: Respostas às perguntas relativas ao tempo de uso de aplicativos e do sistema operacional utilizado pelos usuários.

forma, as próximas perguntas se destinam exclusivamente aos usuários do RU, e são relativas a experiência desses usuários com o RU.

As próximas duas perguntas foram inseridas com o objetivo de identificar a frequência de utilização do RU pelos entrevistados e a refeição principal que os entrevistados costumam realizar no restaurante. Assim, detectou-se que quase 80% dos usuário interrogados realizam alguma refeição no restaurante duas ou mais vezes por semana e a refeição predominante realizada pelos entrevistados é o almoço. Na Figura 4.6 é possível observar o resultado completo para essas perguntas.

A pergunta "Você já teve que esperar em filas para comprar créditos no RU?" foi inserida com o propósito de validar a questão das filas para a compra de créditos, que é o principal problema abordado neste trabalho. O resultado, apresentado na Figura 4.7, mostrou que 95.9% dos usuários do RU já enfrentou filas.

A próxima pergunta feita com os futuros usuários da aplicação foi "Quais funcionalidades você acha interessante para um aplicativo do RU?". Esse questionamento foi adicionado com a finalidade de validar possíveis funcionalidades, e abrir espaço para os usuários sugerirem novas funcionalidades para o aplicativo. Para tal, foi apresentado ao entrevistado algumas opções de funcionalidade, como a compra de créditos *online*, a avaliação, as notificações e as notícias do RU. Além disso, o entrevistado podia adicionar sugestões.

Nessa pergunta, as funcionalidades como a compra de créditos e a disponibilização do

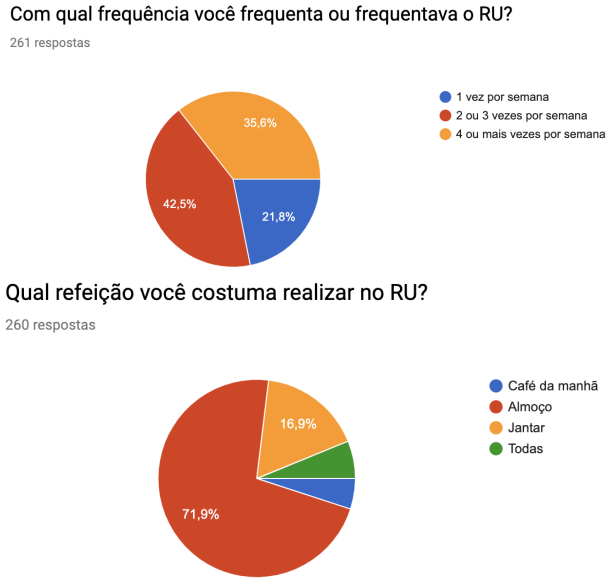


Figura 4.6: Respostas às perguntas relativas ao tempo de uso de aplicativos e do sistema operacional utilizado pelos usuários.

Figura 4.7: Reposta da pergunta: "Você já teve que esperar em filas para comprar créditos no RU?".

cardápio da semana foram validadas por mais de 88% dos entrevistados, enquanto a funcionalidade de notificações foi selecionada apenas por pouco mais de 18%. A avaliação do cardápio foi escolhida por 66.7%, enquanto o compartilhamento externo do cardápio por 43.3% das pessoas. Importantes sugestões de funcionalidades, como a disponibilização das informações de saldo e extrato, foram sugeridas nas respostas dessa pergunta. Essas funcionalidades foram posteriormente incorporadas ao aplicativo durante o seu desenvolvimento, como será apresentado na Seção 2.2.

A próxima pergunta indagou os entrevistados em relação à forma de pagamento utilizado para a compra de crédito. O resultado, mostrado na Figura 4.8, expôs que mais de 90% dos entrevistados preferem o pagamento *online* via cartão de crédito, e 30% escolheram também o pagamento por boleto bancário. Esse resultado confirma o que foi

discutido na Seção 2.8, e valida a compra *online* por meio do cartão de crédito.

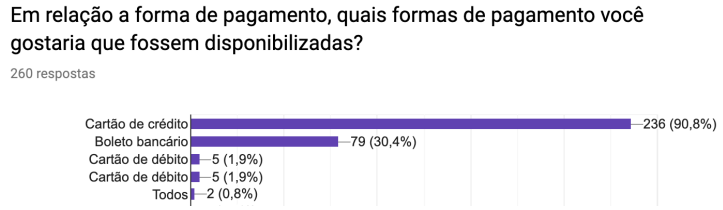


Figura 4.8: Reposta da pergunta: "Em relação a forma de pagamento, quais formas de pagamento você gostaria que fossem disponibilizadas?".

A última pergunta se refere a propensão dos usuários do RU a utilizar um aplicativo móvel que trouxesse soluções relacionadas ao restaurante universitário. Assim, perguntou-se ao entrevistado "Quão propenso você estaria a utilizar um aplicativo móvel que trouxesse soluções relacionadas ao RU?", e as opções de resposta variaram de 1 a 5. O resultado, apresentado na Figura 4.9, mostrou que mais de 80% dos usuários estariam bastante propensos a utilizar o aplicativo (4 e 5, na escala de 1 a 5), enquanto pouco mais de 6% estariam pouco propensos a ser usuários do aplicativo (1 e 2, na escala de 1 a 5).

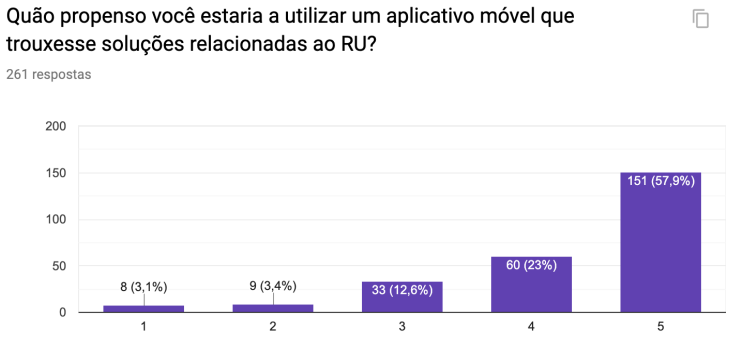


Figura 4.9: Resultado da pergunta: "Quão propenso você estaria a utilizar um aplicativo móvel que trouxesse soluções relacionadas ao RU?".

Portanto, pode-se dizer que a pesquisa alcançou o objetivo de entender o perfil e as necessidades do usuário, e levantar requisitos para o aplicativo. Por meio da análise do resultado da pesquisa, foi possível identificar os principais problemas e demandas dos usuários do RU e, dessa forma, validar, invalidar ou adicionar novas funcionalidades à aplicação.

Após essa primeira etapa de levantamento de requisitos, foram realizadas algumas reuniões com a direção do RU, a empresa responsável pela gestão do restaurante e o CPD, para que os requisitos levantados fossem avaliados e validados, de acordo com a viabilidade de cada um e a realidade do restaurante. Dessa forma, a próxima seção apresenta as funcionalidades implementadas pelo aplicativo.

4.4 Funcionalidades do Aplicativo RUnB

A tela inicial do aplicativo, apresentada na figura Figura 4.10, consiste em uma *Activity* com quatro *Fragments* principais dispostas sobre o componente *Bottom Navigation View*. Como definido pelo *Android Material Design Guidelines* em [35], o *Bottom Navigation* deve estabelecer de três a cinco destinos importantes. Isso é seguido pelo aplicativo que define quatro *Fragments* principais com as seguintes funções:

- Apresentar o cardápio por campus, dia e refeição, com a possibilidade de compartilhamento externo.
- *Login*, para acesso a funcionalidades como a compra de créditos, saldo, extrato, visualização do grupo pertencente e geração do código de barras de acesso.
- Avaliação do RU e do cardápio.
- *Menu* do usuário, que providencia acesso às notícias e às informações do RU e do aplicativo.

O aplicativo inicia na *Fragment* de apresentação do cardápio, por ser a funcionalidade mais utilizada no aplicativo, e possibilita a navegação para as outras três *Fragments* principais.

Dessa forma, as funcionalidades presentes no aplicativo RUnB serão exploradas, em detalhes, nas próximas seções.

4.4.1 Login do Usuário

A tela de *login* do usuário é exibida anteriormente a exibição de ações e informações que dependam da autenticação do usuário, como, por exemplo a visualização do extrato e a compra de créditos. Essa tela possui um *layout* simples, com campos para o usuário (aluno ou servidor) entrar utilizando seu registro acadêmico - RA (matrícula, no caso dos alunos) e sua senha. Para realizar essa autenticação, o aplicativo faz uso do mesmo serviço de autenticação que outros sistemas da UnB, como o MatriculaWeb, utilizam. Assim sendo, por meio de uma requisição HTTP e utilizando o protocolo OAuth2, o aplicativo envia as informações do usuário (RA e senha) para o serviço de autenticação da UnB. Esse serviço por sua vez responde com sucesso (*Status Code 200*) ou erro. Em caso de sucesso, o *Web Service* responde com um arquivo JSON contendo as informações do usuário, como a matrícula, o nome e o email. Outro parâmetro importante retornado é o *token* de acesso do usuário, o qual deve ser armazenado para ser utilizado em todas as outras chamadas aos serviços da UnB. Vale ressaltar que esse *token* tem duração determinada e, desse

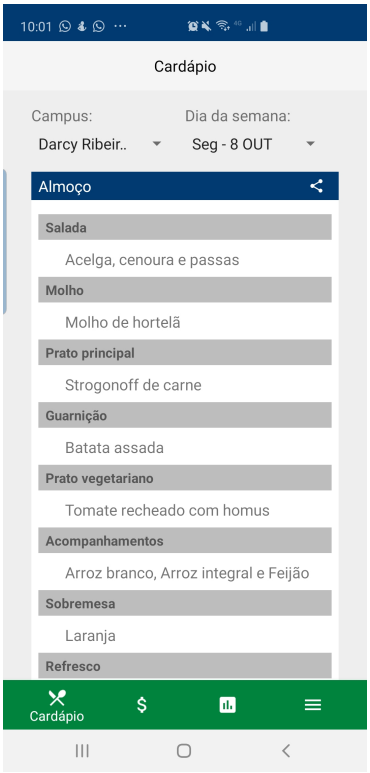


Figura 4.10: Tela Inicial do Aplicativo RUnB.

modo, deve ser atualizado quando expirar. A tela de *login* do aplicativo pode ser vista na Figura 4.11.

No caso de resposta de erro do serviço de autenticação, o aplicativo deve tratar o erro e apresentar ao usuário. O serviço pode retornar uma resposta de erro por diversos motivos alheios ao aplicativo, como, por exemplo uma instabilidade ou queda dos servidores nos quais o serviço encontra-se hospedado. No entanto, para o usuário do aplicativo, apenas dois tipos de *feedback* devem ser apresentados, a depender da resposta do *Web Service*. O primeiro é o caso do usuário que tenta logar com dados de *login* (matrícula e/ou senha) incorretos, nesse caso, o sistema informa o usuário o erro no preenchimento. O segundo *feedback* abrange todos as outras respostas de erro do serviço e, para o usuário final, deve ser apresentada uma mensagem de erro no servidor. Esses *feedbacks* são apresentados no aplicativo em forma de caixas de diálogo, que podem ser vistas na Figura 4.12.

4.4.2 Cardápio e Compartilhamento do Cardápio

O cardápio, presente em todos os aplicativos analisados na Seção 2.1, foi a funcionalidade mais demandada pelos estudantes durante o levantamento de requisitos 4.3. Dessa forma, o desafio na implementação da funcionalidade de apresentação do cardápio está em apre-

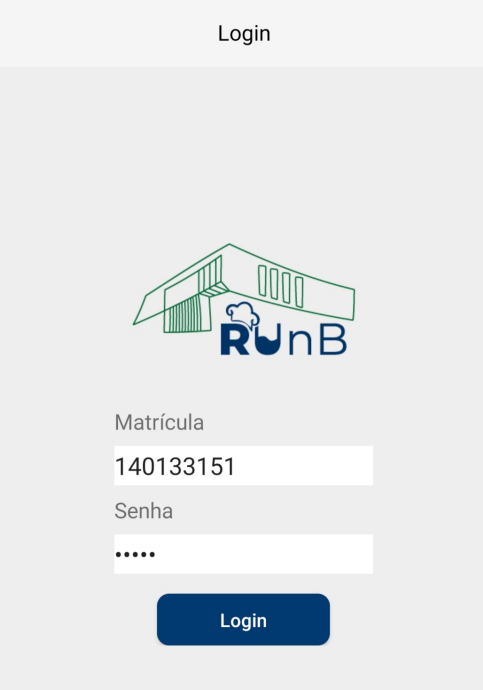


Figura 4.11: Tela de *login* do aplicativo RUnB.

sentar o conteúdo do cardápio para o usuário da maneira mais clara e intuitiva possível de forma a evitar a confusão do usuário enquanto ele seleciona campi, data e refeição.

Tendo isso em mente, o aplicativo RUnB apresenta o cardápio como primeira tela do aplicativo. As opções de seleção (campus e data) que levam ao cardápio são apresentadas por meio do componente *Spinner* do *Android*, que possibilita escolher entre múltiplas opções. Além disso, as refeições (café da manhã, almoço e jantar) foram separadas em diferentes *layouts*, visando evitar a sobrecarga de informação na mesma tela. Para navegar entre as refeições, o usuário pode deslizar a tela lateralmente ou clicar no botão indicativo de troca de *layout*. A primeira tela exibida é a de almoço, a refeição que possui a maior demanda. Seguindo a ordem cronológica, a sua esquerda temos o café da manhã e a direita o jantar. A tela de cardápio pode ser vista na figura Figura 4.13.

O *Web Service* de cardápio, que dá suporte ao aplicativo, ainda está em fase de desenvolvimento pelo CPD/UnB. Dessa forma, devido ao fato do cardápio não estar totalmente integrado a um serviço, que ainda está em construção, esta é a única funcionalidade do aplicativo que não pode ser completamente testada. De todo modo, o aplicativo encontra-se integrado com a versão do *Web Service* de cardápio até então disponibilizado pelo CPD/UnB.

Além do cardápio em si, o aplicativo possibilita ao usuário compartilhar o cardápio externamente através de um botão de compartilhamento, que pode ser observado na Figura 4.13. o usuário pode compartilhar o cardápio, em forma de imagem, com seus

(a) Dados inválidos. (b) Erro no servidor.

Figura 4.12: *Feedbacks* na falha do *login*.

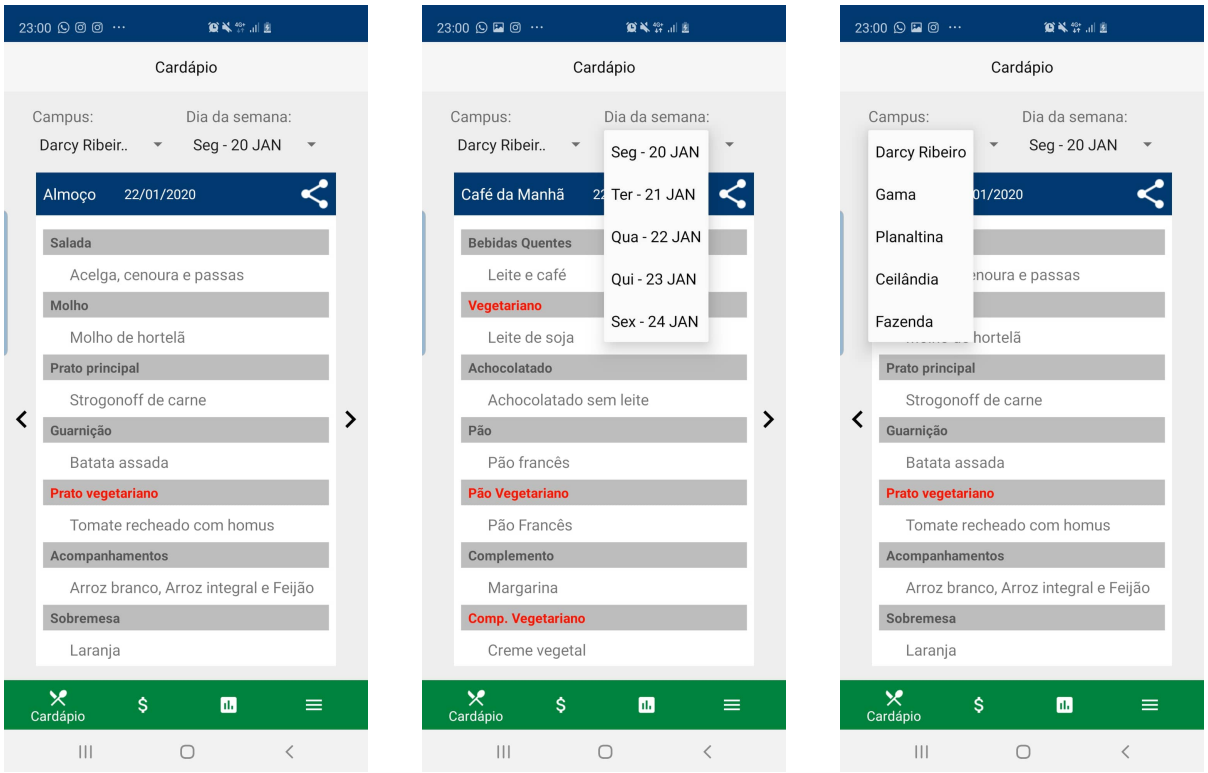
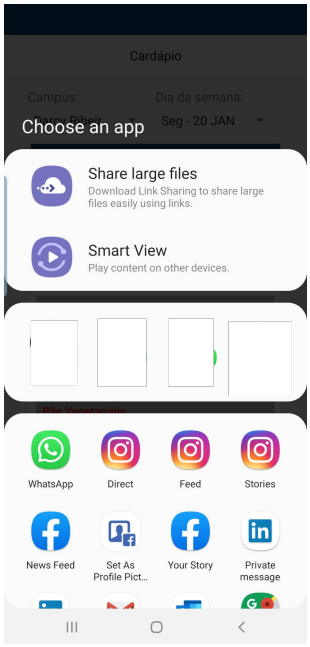


Figura 4.13: Tela de cardápio do aplicativo RUnB.



(a) Opções de compartilhamento externo.

(b) Imagem gerada para compartilhamento.

Figura 4.14: Compartilhamento Externo do Cardápio.

amigos em outros aplicativos externos ao RUnB, como o *WhatsApp* e o *Facebook*. Desse modo, o acesso ao cardápio pode ser disseminado de forma exponencial, fazendo com que mais pessoas possam visualiza-lo.

Para fazer com que o compartilhamento do cardápio seja possível, o aplicativo transforma a *View* que exibe o conteudo da refeição (café, almoço e jantar) em uma imagem e faz com que essa imagem seja passível de compartilhamento. O processo de compartilhamento é apresentado na figura Figura 4.14.

4.4.3 Saldo, Grupo e Valor da Refeição

O acesso ao saldo, grupo de usuário e valor da refeição para determinado aluno são informações simples, porém, que não são de fácil acesso. O saldo, por exemplo, hoje pode ser acessado via Matricula Web, o que demanda a abertura do site do Matricula Web no *browser* e o *login* no portal. Com o aplicativo, o aluno consegue ver seu saldo em questão de segundos. O grupo ao qual o aluno pertence é uma informação importante, principalmente quando se trata dos alunos do PNAES (Plano Nacional de Assistência Estudantil). O acesso a essa informação via aplicativo possibilita aos alunos que solicitaram participar desse programa verificar se ainda são beneficiários do PNAES.

As informações de saldo, grupo e valor da refeição do grupo pertencente são apresentadas em conjunto, devido ao fato de que estão, de certa forma, interligadas. Para recuperar

essas informações o aplicativo realiza duas chamadas HTTP GET a *Web Services:* uma para recuperar o saldo, e outra para recuperar o grupo e o valor da refeição. Para realizar essa chamada, o aplicativo envia o *token* de acesso do usuário, salvo após o *login*, no cabeçalho da requisição, devido ao fato de que essas são informações restritas ao aluno. A tela que exibe essas informações é apresentada na Figura 4.15.

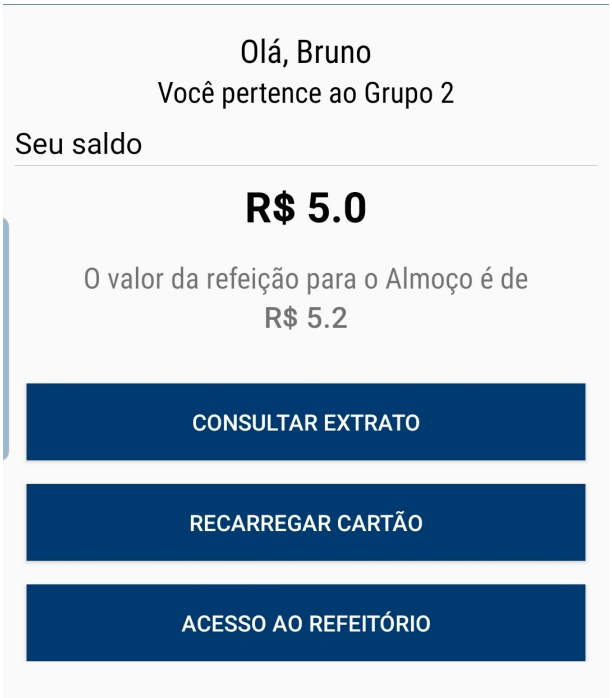
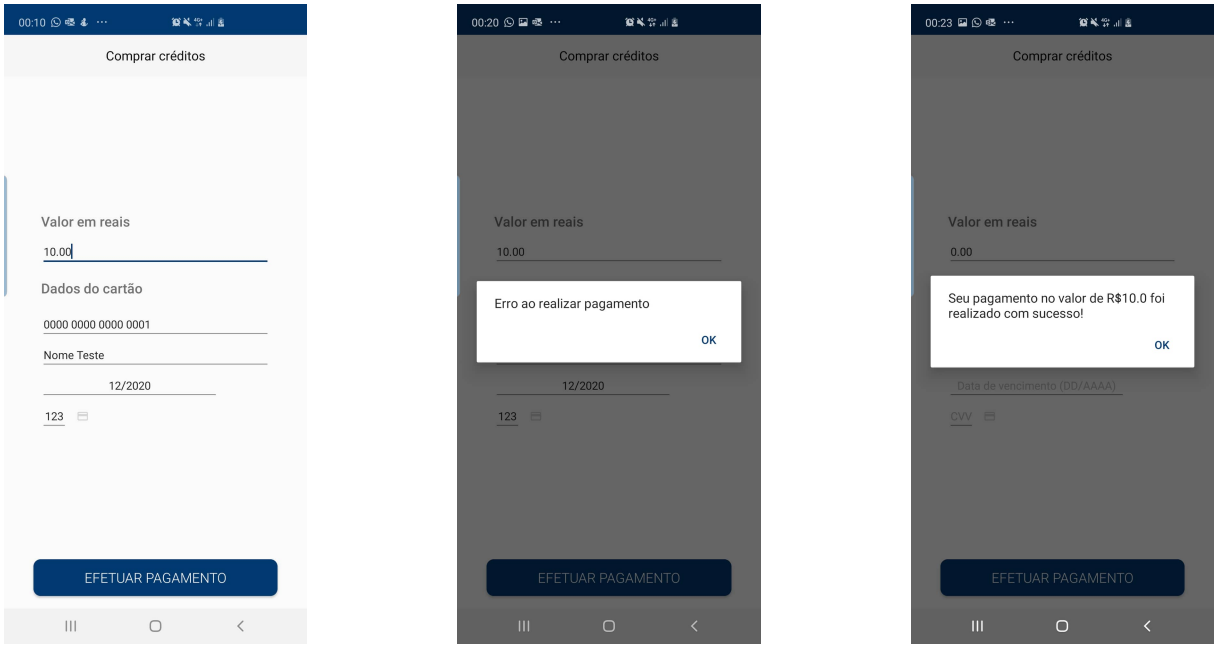


Figura 4.15: Tela de Saldo, Grupo e Valor da Refeição do Aplicativo RUnB.

Como notado na Figura 4.15, abaixo da apresentação dessas informações, é possível observar três botões, que redirecionam para outras telas com funções diferentes. O primeiro redireciona o usuário para a tela de extrato, na qual o usuário pode ter acesso a sua movimentação financeira do RU. O segundo botão redireciona o usuário para o fluxo de compra de créditos, que será explorada posteriormente. O terceiro e último botão, abre a tela que apresenta o código de barras para acesso ao refeitório. Como exposto na Seção 4.2, os alunos do grupo 1 são isentos do pagamento do valor da refeição. Dessa forma, o botão de "Recarregar Cartão" não aparece para alunos desse grupo.

4.4.4 Compra de Créditos

A compra de créditos é uma das funcionalidades mais invadoras do RUnB. Como foi exposto no Capítulo 2, apenas um aplicativo possui essa funcionalidade, e ela é implementada por meio do pagamento via boleto bancário. No aplicativo RUnB, a compra de créditos do RU é realizada por meio do cartão de crédito. O usuário escolhe um valor a



(a) Tela de comprar créditos.

(b) Mensagem de erro na compra.

(c) Mensagem de sucesso na compra.

Figura 4.16: Fluxo de Compra de Créditos.

ser cobrado, adiciona os dados do cartão de crédito e realiza o pagamento. Dessa forma, em caso de sucesso na compra, o pagamento é computado na mesma hora, e os créditos estão prontos para serem utilizados.

Para realizar essa operação, o aplicativo realiza chamadas a dois serviços *web*: a API de pagamentos da Cielo e ao serviço de venda de créditos do RU. O primeiro foi disponibilizada pela empresa administradora do RU, que recebe os pagamentos. O segundo foi desenvolvido pelo CPD/UnB, e é chamado apenas em caso de sucesso do primeiro.

O fluxo de pagamento pode ser resumido da seguinte maneira. O usuário preenche o valor (em reais) a ser comprado e as informações do cartão e aperta o botão de pagamento. O aplicativo envia a requisição, com o valor e os dados do cartão, para a API de pagamento. A API de pagamento responde com sucesso ou erro. Em caso de erro, o aplicativo exibe uma caixa de dialógo com uma mensagem de erro. Em caso de sucesso, o aplicativo chama o serviço de venda do RU com o valor da compra realizada. O serviço de venda, por sua vez, responde com sucesso ou erro. Em caso de erro, o aplicativo realiza uma chamada de cancelamento/estorno a API de pagamento, que automaticamente cancela a compra. Em caso de sucesso no serviço de venda, é exibida uma mensagem de sucesso juntamente com o valor comprado pelo usuário. O fluxo de compra de créditos no aplicativo pode ser visto na figura Figura 4.16.

Seguindo esse fluxo de pagamento, garante-se ao usuário a confiabilidade em realizar a compra de créditos online, tendo em vista o fato de que não há a possibilidade de perda

do dinheiro.

4.4.5 Extrato

A funcionalidade de acesso ao extrato do RU funciona de maneira semelhante ao extrato bancário. Através dela, o usuário pode ter acesso a movimentação financeira do seu cartão RU, ou seja, o usuário tem acesso a data da transação, a descrição dela e ao valor. Para recuperar essas informações, o aplicativo realiza uma chamada HTTP GET ao serviço de extrato disponibilizado pelo CPD/UnB.

A transação pode ser de dois tipos: compra de créditos, que pode ser de qualquer valor, e consumo, que pode ter sido realizado no café da manhã, almoço ou jantar. No caso da compra de créditos, a descrição apresentada para o usuário é de compra, e o valor corresponde ao da compra. No caso do consumo, o aplicativo realiza uma verificação do horário em que foi realizado o consumo para apresentar ao usuário, no campo de descrição, o tipo de refeição realizada: café da manhã, almoço ou jantar.

Além disso, o aplicativo permite ao usuário selecionar o período de sua movimentação financeira. O usuário pode escolher três períodos de visualização: um, dois e três anos antes da data corrente. Dessa forma, o aplicativo realiza a chamada ao serviço de extrato passando os parâmetros de data selecionado pelo usuário. Além disso, na tela de extrato, o usuário também tem acesso ao seu saldo atual.

Dessa forma, o usuário pode ter acesso a sua movimentação financeira (extrato) de forma clara e organizada, que é uma informação importante pelo fato de se tratar do uso de dinheiro. A figura Figura 4.17 exibe a tela de extrato do aplicativo RUnB, com o resultado para diferentes seleções de período.

4.4.6 Acesso ao Refeitório

A funcionalidade de acesso ao refeitório por meio do aplicativo é de extrema importância, pelo fato de que através dela os usuários podem acessar o refeitório sem necessidade de portarem a carteirinha de estudante. Uma situação recorrente no RU é a de os estudantes não possuírem ou esquecerem a carteirinha e terem que recorrer à administração do restaurante portando a identidade e o comprovante de matrícula, a fim de realizar a compra de crédito e assim poder realizar o acesso o refeitório. Nesse caso, a administração imprime um *ticket* unitário de acesso ao restaurante, válido apenas para aquele dia.

Com o objetivo de fornecer aos alunos acesso rápido e simplificado ao refeitório, a aplicação gera um código de barras idêntico ao presente na carteirinha de estudante. Para gerar o código, a aplicação realiza uma requisição HTTP do tipo GET a um serviço, que retorna o número do código de barras. O aplicativo codifica esses números em um

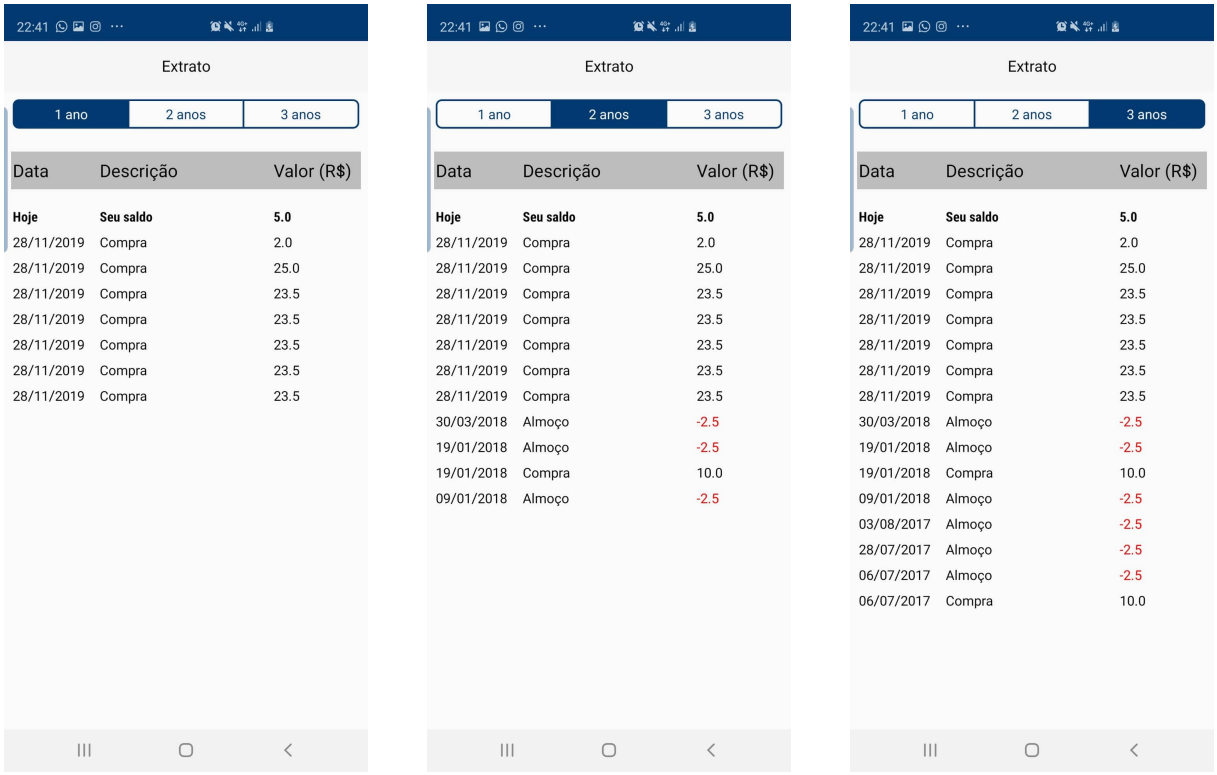


Figura 4.17: Tela de Extrato do Aplicativo RUnB

código de barras. Existem diversos formatos para a codificação de um código de barras. Um desses é o *Code39*, que é o formato utilizado pelas máquinas leitora de código de barra do RU. Devido a isso, esse foi o formato escolhido para a codificação no aplicativo. Dessa forma o aluno logado no aplicativo pode apresentar o código de barras ao fiscal na catraca, que realiza a leitura do código de barras e o sistema automaticamente libera a entrada do usuário. A tela de acesso ao refeitório é mostrada na Figura 4.18.

Outro beneficio do uso do código de barras no aplicativo é a diminuição da probabilidade de falhas. Hoje o acesso ao refeitório é realizado por meio da leitura das carteirinhas pelas catracas e, segundo a administração do RU, as falhas e necessidades de manutenção dessas catracas são constantes. Com o código de barras no aplicativo, os fiscais poderão utilizar a pistola de leitura de código de barras para realizar a leitura, a qual é mais eficiente e apresenta menos falhas.

4.4.7 Avaliação

A funcionalidade de avaliação é de extrema importância tanto para os usuários quanto para a gestão do RU. A possibilidade de visualizar avaliações de outros usuários é muito útil, tendo em vista que isso pode influenciar de maneira substancial o processo de tomada de decisão do usuário, como explicado na Seção 2.5.

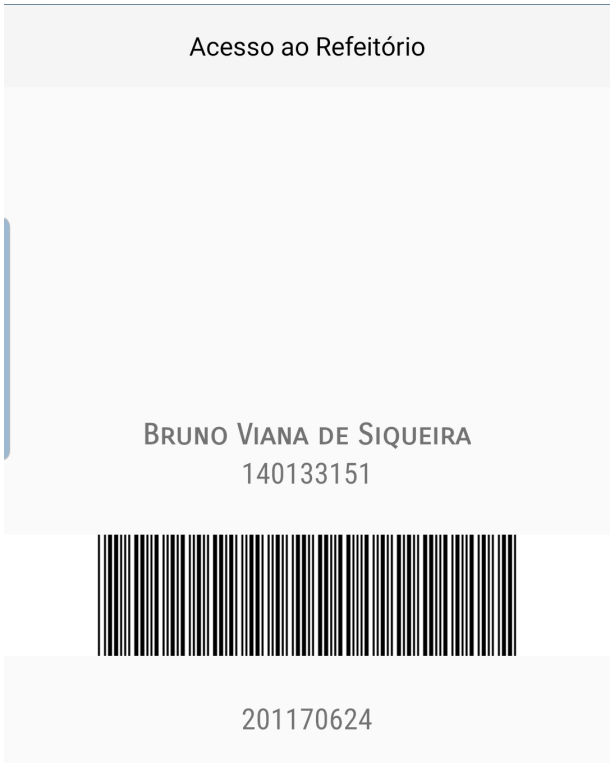


Figura 4.18: Tela de acesso ao refeitório do aplicativo RUnB.

Além disso, a funcionalidade de avaliação é de grande valor também para a gestão do restaurante universitário. É através do *feedback* dos usuários que a gestão do RU pode identificar o que deve ser melhorado e, dessa forma, fazer com que o restaurante esteja em constante evolução e atenda cada vez mais as necessidades da comunidade acadêmica.

No aplicativo RUnB, essa funcionalidade é implementada utilizando o banco de dados gratuito da *Google*, o *Firestore*, que é parte do *Firebase*. O *Firestore* é um banco de dados *NoSQL* bastante utilizado nos dias de hoje. Nele, os dados são armazenados em documentos que possuem mapeamento de campo para valores [36]. As consultas no *Firestore* são rápidas e eficientes, e sua flexibilidade permite adicionar qualquer estrutura de dados que seja necessário à aplicação.

A avaliação é exibida para o usuário como a terceira aba do aplicativo. Nela, é possível visualizar as avaliações de outros usuários, que estão dispostas em uma lista e ordenadas por ordem cronológica, e também escrever sua própria avaliação. Quando o usuário escreve a avaliação e aperta o botão de enviar, uma caixa de diálogo, com a opção de escolher a refeição correspondente à avaliação e a nota, é exibida. A nota é representada pelo componente *RatingBar* do Android. O usuário deve escolher um valor numa escala de 1 a 5 estrelas, e cada estrela pode ser não preenchida, preenchida pela metade ou completamente preenchida. Dessa forma, o usuário possui dez valores na escala

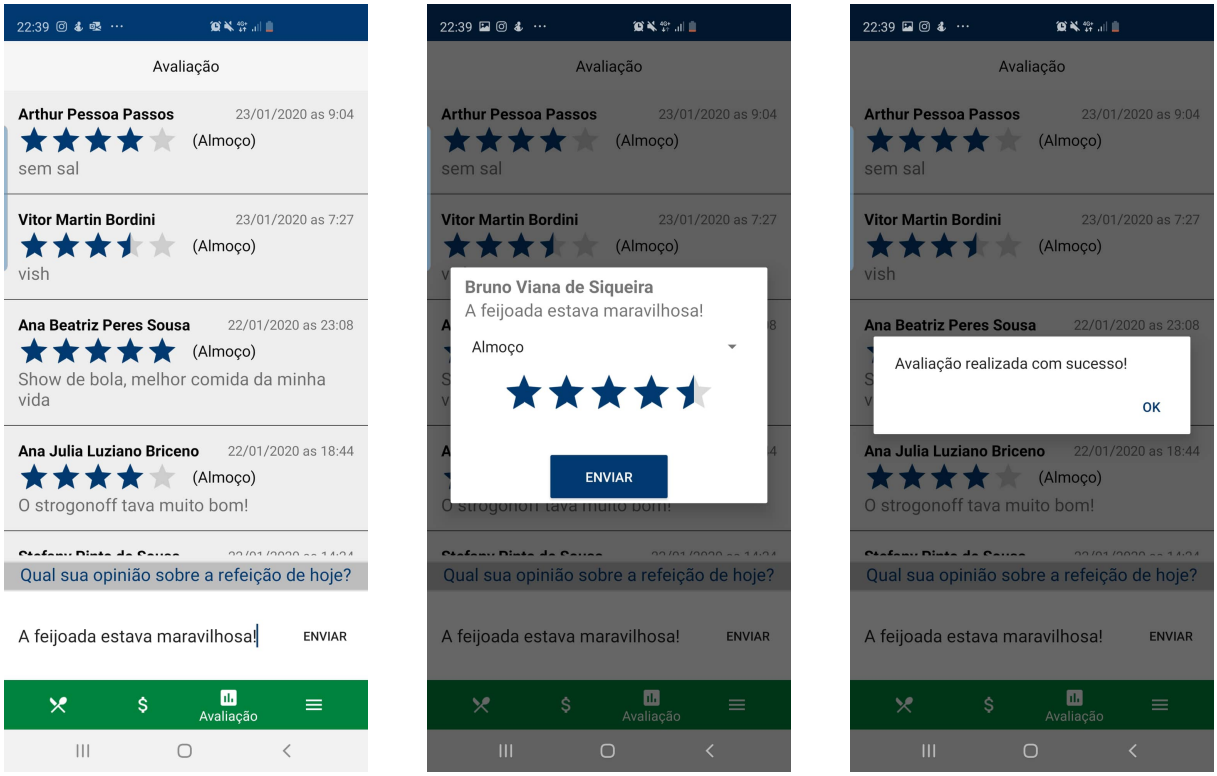


Figura 4.19: Fluxo completo de avaliação.

de avaliação e, assim, pode expressar sua opinião de maneira mais assertiva.

A escolha de aliar a avaliação por nota com a avaliação escrita vem da ideia de dar ao usuário a liberdade de expressar sua opinião de maneiras mais aberta, ao mesmo tempo que avalia a refeição objetivamente. O fluxo completo de avaliação pode ser visualizado na figura Figura 4.19.

4.4.8 Notificação

A funcionalidade de notificação é implementada utilizando o *Firebase Cloud Messaging* (FCM). O FCM é uma solução de mensagens entre plataformas que permite o envio de notificações sem custo [37]. Por meio dele, é possível enviar notificação a todos os usuários que possuem o aplicativo ao mesmo tempo e, dessa forma, informar a comunidade acadêmica rapidamente a respeito de informes relacionados ao RU, como a realização de um evento ou o fechamento do restaurante em uma data específica.

Nas reuniões realizadas com a direção do RU, foi exposta a dificuldade de realizar um comunicado a comunidade acadêmica de forma efetiva. Assim, essa funcionalidade surgiu para solucionar a dificuldade mencionada. A elaboração e envio de notificações por meio do painel do FCM será feito pela próprio direção do RU, e as mensagens serão replicadas nos celulares que possuírem o aplicativo. Um exemplo de notificação recebida

pelo aplicativo pode ser visto na Figura 4.20. O painel de envio de notificação, que será utilizado pela direção do RU, pode ser visto na Figura 4.21.

O aplicativo recebe a mensagem enviada pelo painel por meio de um serviço que estende de *FirebaseMessagingService*. Esse serviço sobrescreve o método *OnMessageReceived*, que é o *callback* chamado sempre que uma nova notificação chega ao aplicativo e o aplicativo se encontra em primeiro plano (*foreground*). Caso o aplicativo esteja em *background*, o método *OnMessageReceived* não é chamado, mas a mensagem de notificação também é exibida.

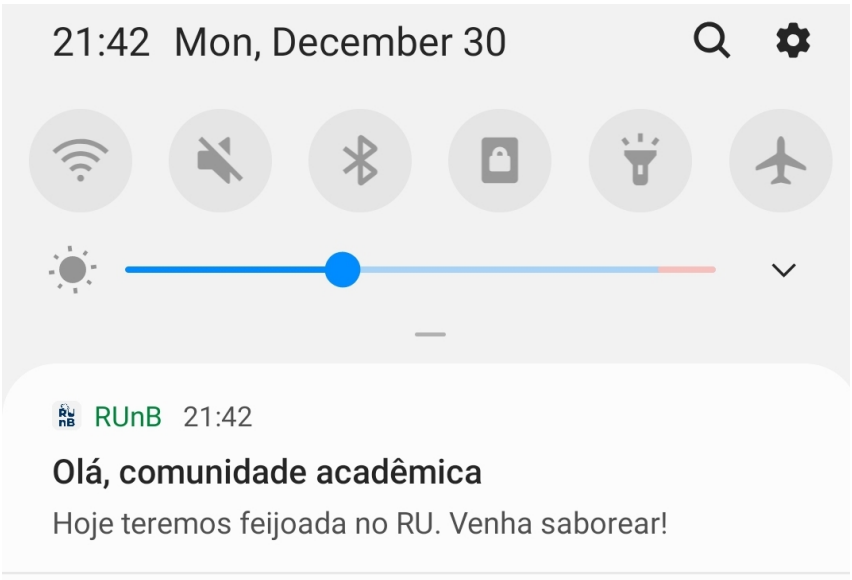


Figura 4.20: Mensagem recebida pelos aplicativos.

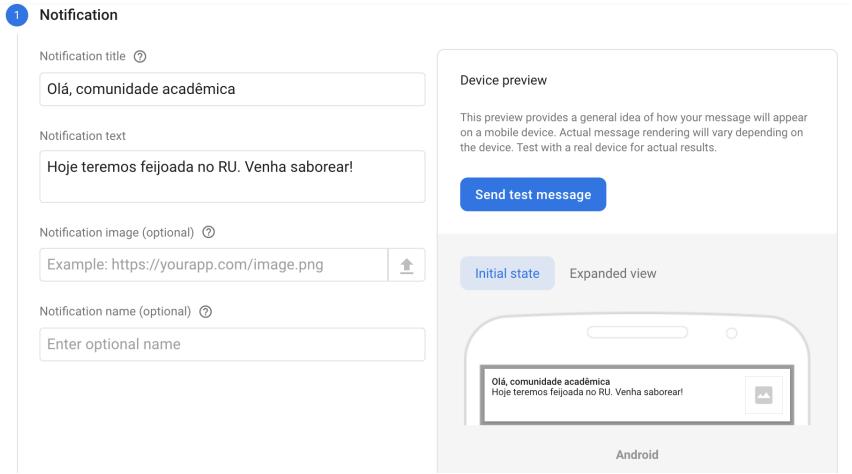


Figura 4.21: Painel do FCM para envio de notificação.

4.4.9 Notícias do RU

A tela de notícias do RU consiste apenas em uma visualização via aplicativo da página de notícias presente no site do restaurante, facilitando, assim o acesso dos usuários a essas notícias. Para apresentar o site de notícias, o aplicativo utiliza o componente *WebView*, utilizado no *Android* para exibir páginas da *web*. Ao realizar o toque na notícia com o objetivo de realizar a leitura completa, o aplicativo redireciona o usuário para o *browser*. Parte da visualização da página de notícias do RU pelo aplicativo é apresentada na Figura 4.22.

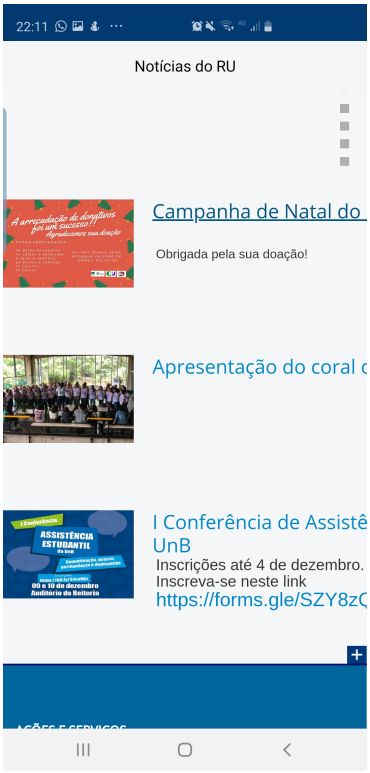


Figura 4.22: Tela de Notícias do Aplicativo.

4.4.10 Informações do RU e do Aplicativo RUnB

As telas de informações do RU e do aplicativo RUnB servem para possibilitar aos usuário do aplicativo um acesso rápido a informações importantes, como por exemplo o horário de funcionamento do restaurante e as informações de contato. Além disso, a tela de informações do aplicativo contém uma visão geral do aplicativo e dos seus objetivos, e apresenta as entidades e pessoas envolvidas no desenvolvimento do aplicativo. As duas telas de informações podem ser vistas na figura Figura 4.23.

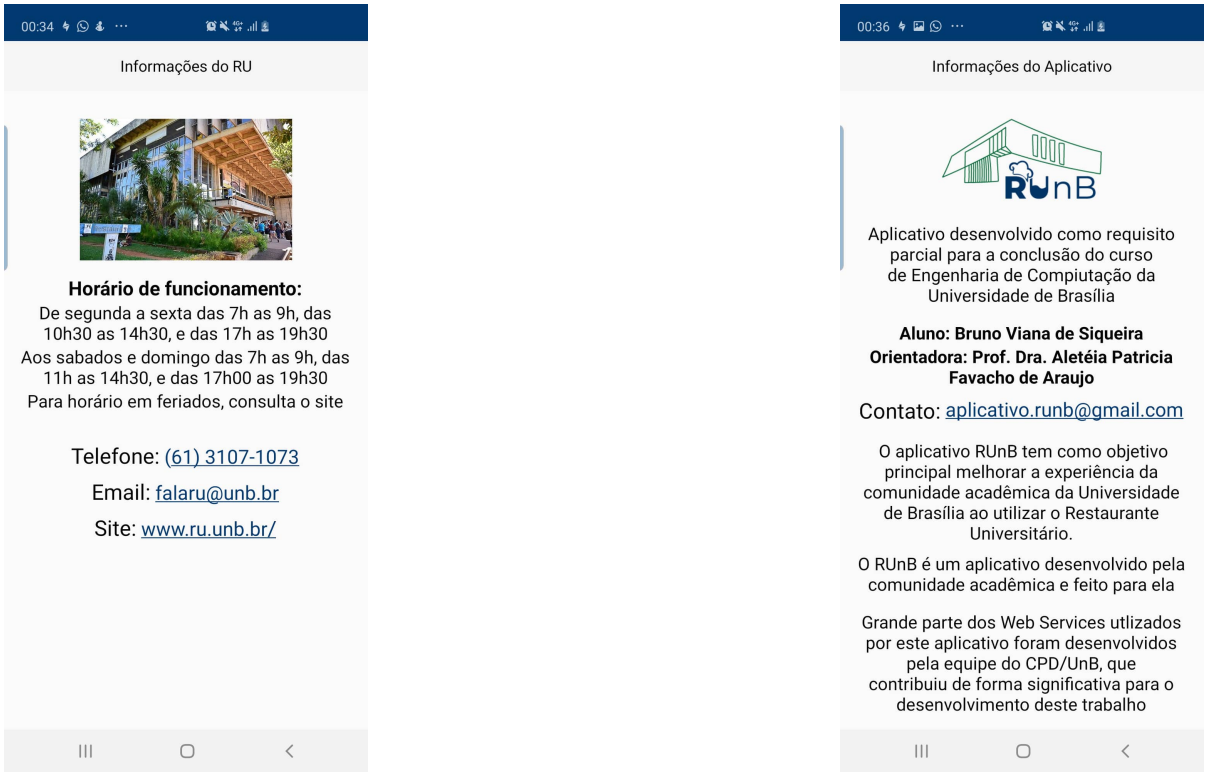


Figura 4.23: Telas de Informação do RU e do Aplicativo.

Os campos de *email* e telefone possibilitam ao usuário, através do toque nesses campos, o acesso direto aos aplicativos de *email* e telefone. Dessa forma, o usuário não possui a necessidade de abrir outro aplicativo e redigitar o conteúdo dos campos caso haja a necessidade de entrar em contato com o restaurante ou com o aplicativo.

4.5 Resultados

Para que fosse possível avaliar os resultados do uso do aplicativo RUnB na prática, foi levantada a necessidade da realização de testes da aplicação com usuários reais, ou seja, com a comunidade acadêmica da Universidade de Brasília. Dessa forma, foi criado um grupo de estudantes da UnB com pessoas que se dispuseram a realizar os testes no aplicativo e a responder um questionário avaliando o uso de algumas funcionalidades e do aplicativo de maneira geral.

Assim, foi elaborado visando avaliar a usabilidade e experiência dos usuários ao utilizarem as principais funcionalidades do aplicativo. Dentre alguns pontos avaliados no questionário se destacam a visualização do cardápio, do extrato e do saldo, a geração do código de barras de acesso e a funcionalidade de avaliação. Todas as questões do questionário aceitavam respostas do usuário variando de um a cinco.

Avaliar a experiência de usuário e usabilidade do aplicativo é de suma importância para a validação de um produto de *software*. É a partir do *feedback* dos usuários que estratégias de implementação de funcionalidades podem ser validadas ou revistas, além de abrir a possibilidade de identificar erros, problemas e futuras melhorias do aplicativo.

Dessa forma, dezoito alunos da UnB se dispuseram a testar o RUnB. Esses usuários foram orientados a instalar o aplicativo e a responder o questionário à medida que avançavam no teste das funcionalidades.

A primeira funcionalidade a ser avaliada pelos usuários foi o cardápio, justamente por ser a tela de entrada do aplicativo. Os testadores foram questionados em relação ao cardápio em três pontos: intuitividade para perceber a alternância entre refeições, apresentação do cardápio e compartilhamento externo do cardápio.

O resultado da primeira pergunta mostrou que apenas um usuário dentre os dezoito que testaram não achou intuitivo a alternância entre refeições. Todos os demais classificaram a intuitividade desse mecanismo como quatro e cinco, numa escala de um a cinco. A Figura 4.24 mostra o resultado completo para a primeira pergunta.

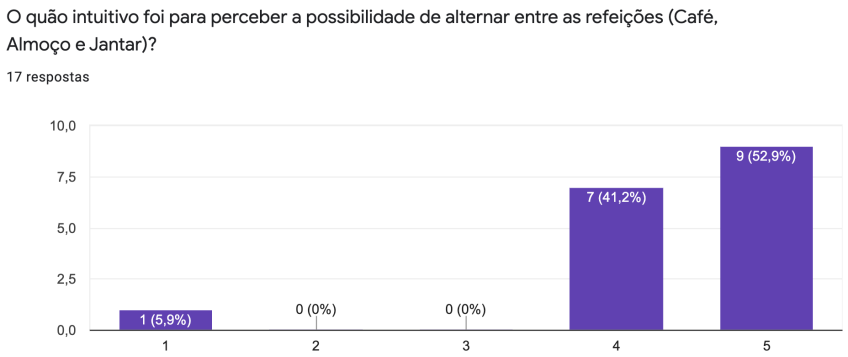


Figura 4.24: Resultado da pergunta: "O quão intuitivo foi para perceber a possibilidade de alternar entre as refeições (Café, Almoço e Jantar)?".

A segunda pergunta do questionário se refere à funcionalidade do cardápio de modo geral. Perguntou-se aos entrevistados "De maneira geral, levando em conta todos os elementos dispostos na tela, como você avalia a maneira que o cardápio é apresentado?" e o resultado pode ser conferido na Figura 4.25. De modo geral, considerando a complexidade em dispor diversos elementos (informações da refeição e botão de compartilhamentos) e *inputs* de escolha do usuário (restaurante, dia e refeição) em uma mesma tela, pode-se considerar que o resultado foi satisfatório.

A próxima pergunta feita aos usuários, ainda a respeito da funcionalidade de cardápio, foi "Como você avalia a funcionalidade de compartilhamento do cardápio?". O resultado, apresentado na Figura 4.26, mostrou que mais de 90% dos testadores avaliaram positivamente essa funcionalidade, atribuindo notas de 4 e 5 numa escala de 1 a 5.

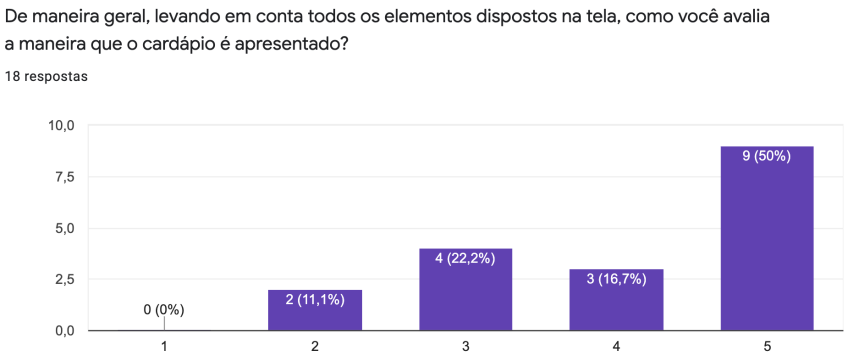


Figura 4.25: Resultado da pergunta: “De maneira geral, levando em conta todos os elementos dispostos na tela, como você avalia a maneira que o cardápio é apresentado?”.

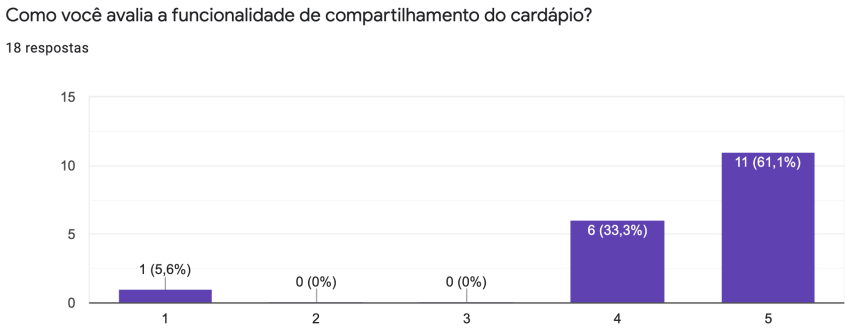


Figura 4.26: Resultado da pergunta: “Como você avalia a funcionalidade de compartilhamento do cardápio?”.

A quarta questão do questionário foi uma afirmação e se referiu ao painel de funcionalidades, onde se encontram as funcionalidades de “Saldo, Grupo do Aluno e Valor da Refeição” apresentadas na Seção 4.4.3. Os usuários foram indagados quanto a possibilidade de identificação de todos elementos de forma clara e objetiva. O resultado completo pode ser visto na figura Figura 4.27.

Então, afirmou-se o seguinte “Consigo visualizar o código de barras e minhas informações de maneira clara e objetiva.” e, como de praxe, os usuários deveriam se posicionar quanto a essa afirmação numa escala de 1 a 5. O resultado, que avalia a funcionalidade de acesso ao refeitório descrita na Seção 4.4.6, mostrou que 15 dos 19 usuários testadores concordam completamente com a afirmação, como mostra a figura Figura 4.28.

A sexta pergunta referiu-se à funcionalidade de extrato, abordada na Subseção 4.4.5. A pergunta, cujos resultados podem ser visualizados na Figura 4.29, mostrou que mais de 90% dos testadores avaliaram positivamente essa funcionalidade.

A próxima pergunta indagou os usuários acerca da funcionalidade de compra de créditos. Assim, perguntou-se ao entrevistado “Como você avalia o fluxo de compra de

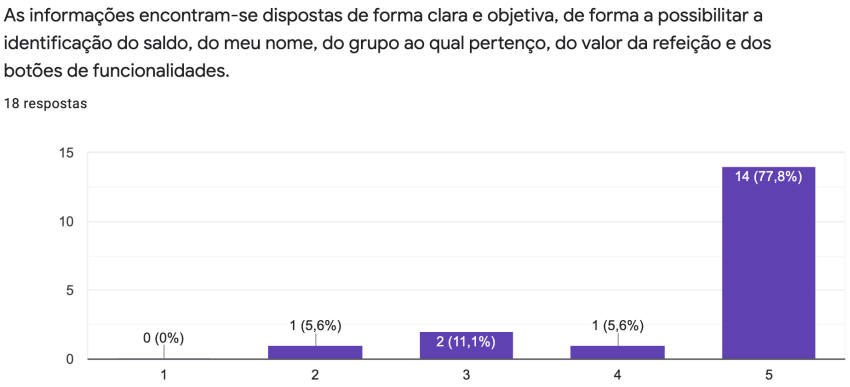


Figura 4.27: Resultado da afirmação: “As informações encontram-se dispostas de forma clara e objetiva, de forma a possibilitar a identificação do saldo, do meu nome, do grupo ao qual pertenço, do valor da refeição e dos botões de funcionalidades.”.

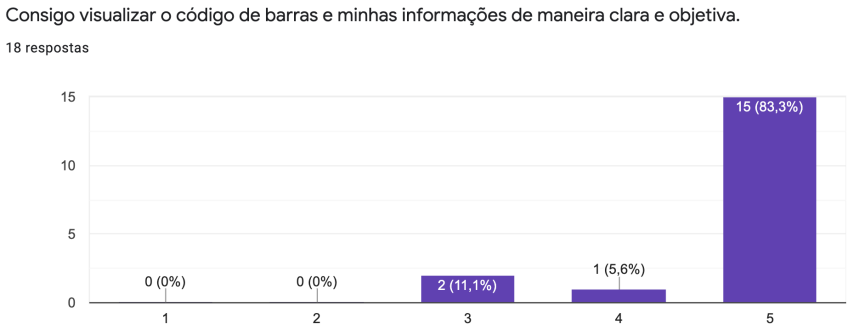


Figura 4.28: Resultado da afirmação: “Consigo visualizar o código de barras e minhas informações de maneira clara e objetiva.”.

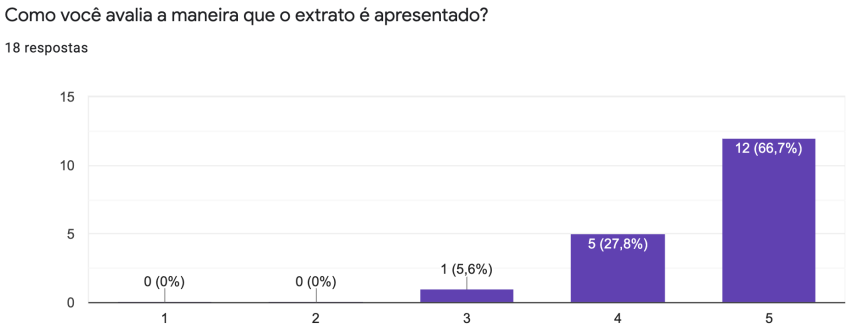


Figura 4.29: Resultado da pergunta: “Como você avalia a maneira que o extrato é apresentado?”.

créditos?” e as respostas variaram bastante, como pode ser visto na Figura 4.30. O questionário possibilitou ao usuário escrever sobre possíveis erros ou sugestões, e alguns usuários relataram problemas ao preencher o campo de valor da compra, problema que foi

logo corrigido. Dessa forma, os testes e *feedbacks* de usuário mostram-se importantes para a identificação e correção de problemas no aplicativo. É importante ressaltar que, para a realização dos testes, o ambiente de *Sandbox* da Cielo (API de pagamento) foi utilizado, de modo a fazer com que nenhum dinheiro real fosse utilizado.

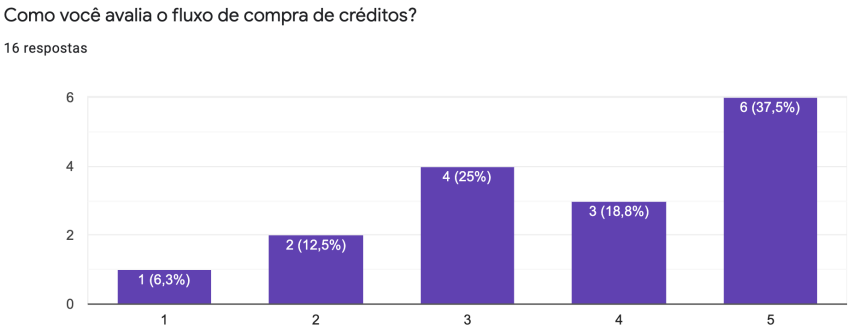


Figura 4.30: Resultado da pergunta: "Como você avalia o fluxo de compra de créditos?".

Em seguida, indagou-se a respeito da funcionalidade de avaliação. Nesse caso, perguntou-se "Como você avalia a funcionalidade de avaliação? Considere tanto a visualização das avaliações quanto o processo de avaliação em si." e o *feedback* dos usuários, que pode ser visto na , foi positivo, resultando em mais de 80% de aprovação da funcionalidade (avaliação 4 e 5).

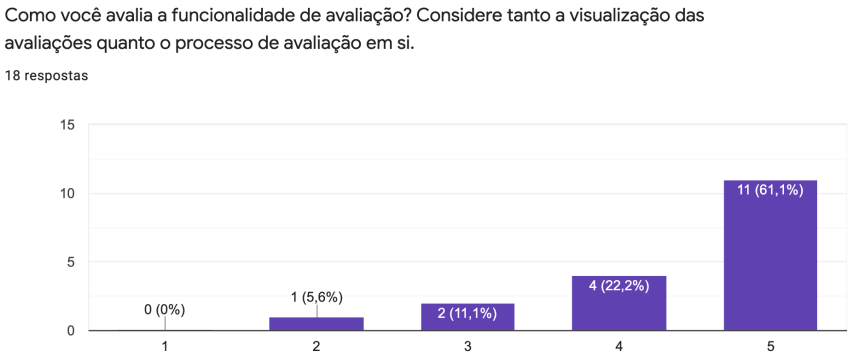


Figura 4.31: Resultado da pergunta: "Como você avalia a funcionalidade de avaliação? Considere tanto a visualização das avaliações quanto o processo de avaliação em si.".

Por ultimo, foram feitas duas perguntas com o objetivo de se obter uma avaliação geral dos usuários em relação ao aplicativo RUnB. A primeira "De forma geral, como você avalia a versão de testes do aplicativo RUnB?" obteve resultados positivos, como expõe a Figura **??** mostrando que quase 90% dos testadores deixaram uma avaliação positiva do aplicativo. A segunda, repetiu uma pergunta realizada no levantamento de requisitos, descrito na Seção 4.3. Perguntou-se aos entrevistados "De 1 a 5, o quão

propenso você estaria a utilizar este aplicativo?" e o resultado apresentado na figura Figura 4.33 foi parecido ao resultado obtido durante o levantamento de requisitos. No entanto, é importante ressaltar a diferença na amostragem (número de participantes) entre as duas entrevistas.

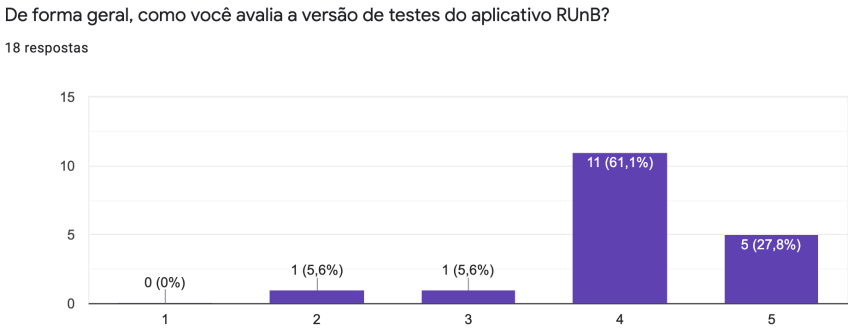


Figura 4.32: Resultado da pergunta: "De forma geral, como você avalia a versão de testes do aplicativo RUnB?".

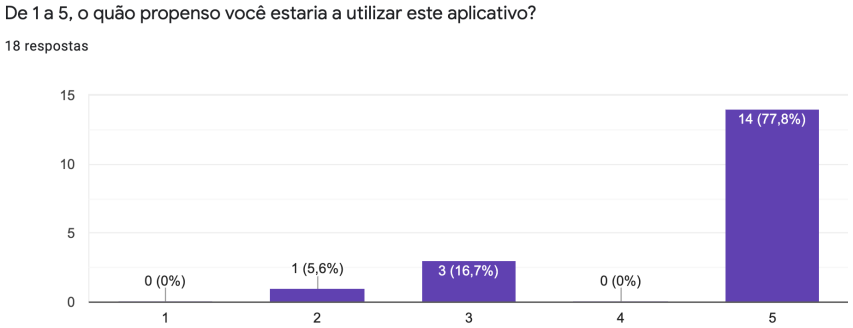


Figura 4.33: Resultado da pergunta: "De 1 a 5, o quão propenso você estaria a utilizar este aplicativo?".

Por fim, ao analisar os resultados do questionário de validação, podemos concluir que, de maneira geral, o aplicativo RUnB obteve avaliação positiva da comunidade acadêmica que testou o aplicativo. Além disso, é importante mencionar os *feedbacks* de usuários recebidos, que serviram para corrigir pequenas falhas e para ajudar na visualização de futura melhorias.

4.6 Considerações Finais

Neste capítulo, abordou-se a história da informatização do RU da Universidade de Brasília, bem como seus problemas e limitações. A partir disso, foi descrito o levantamento de requisitos do sistema, realizado através de uma pesquisa com a comunidade acadêmica.

Dessa forma, foi apresentado o aplicativo RUnB e suas principais funcionalidades foram descritas individualmente. Por fim, o capítulo mostrou o resultado dos testes com usuários.

Capítulo 5

Conclusão

O aplicativo RUnB é um aplicativo móvel composto por um conjunto de funcionalidades voltadas para o usuário do restaurante universitário da UnB. Dentre estas funcionalidades, destaca-se a compra de créditos via aplicativo, que objetiva mitigar de forma significativa o problema das filas.

Tendo como base as funcionalidades apresentadas a partir da pesquisa realizada no Capítulo 2, pode-se concluir que o aplicativo RUnB possui todas as funcionalidades encontradas em aplicativos de RU no Brasil e, além disso, proporciona mais algumas funções. É importante ressaltar que o aplicativo destina-se a todos usuários que utilizam o RU, dentre estes alunos, servidores e visitantes.

O aplicativo disponibilizou para os usuários outras funcionalidades importantes além da compra de créditos. Dentre elas podemos citar a geração do código de barras para o acesso ao refeitório, tornando dispensável portar a carteirinha e diminuindo os transtornos causados quando o estudante não se encontra com ela em mãos, e a apresentação do saldo e extrato, funcionalidade que proporciona maior clareza na relação usuário-restaurante ao apresentar dados relacionados a movimentação financeira. Além disso, a funcionalidade de cardápio, de grupo de usuário e de avaliação proporciona aos usuários um acesso claro, rápido e objetivo a informações importantes para o dia a dia do usuário.

Analisando os resultados dos testes realizado por usuários, pode-se concluir que o aplicativo cumpriu o objetivo de ser uma ferramenta que traz soluções voltadas a melhorar a experiência da comunidade acadêmica, atendendo, de maneira geral, as expectativas e necessidades dos usuários.

Como melhorias e trabalhos futuros, existe a possibilidade de se implementar as seguintes funcionalidades:

- Avaliação por prato específico, além da avaliação por refeição, fazendo com o que o usuário tenha um *feedback* mais detalhado de cada produto;

- Funcionalidade de status das filas do refeitório, que seria uma funcionalidade colaborativa;
- Realizar análise (*Machine Learning*/Processamento de imagens) dos refeitórios do RU Darcy Ribeiro através das imagens das câmeras, apresentando o status de cada refeitório ao usuário (lotado, cheio ou vazio);
- Transferência de créditos de um usuário para outro usuário, de maneira a diminuir ainda mais as filas para a compra de créditos.

Referências

[1] *Platform architecture*. `https://developer.android.com/guide/platform`. ix, 21, 22

[2] *Understand the activity lifecycle*. `https://developer.android.com/guide/components/activities/activity-lifecycle`. ix, 23, 24

[3] Yvonne Rogers, Helen Sharp, Jenny Preece: *Interaction Design: Beyond Human-Computer Interaction*. 2011. 4

[4] Neto, Olibário José Machado: *Usabilidade da interface de dispositivos móveis: heurísticas e diretrizes para o design*. 2013. 8

[5] Parikh, Anish, Carl Behnke, Mihaela Vorvoreanu, Barbara Almanza e Doug Nelson: *Motives for reading and articulating user-generated restaurant reviews on yelp.com*. Journal of Hospitality and Tourism Technology, 5, agosto 2014. 9

[6] Iqbal, Shamsi T. e Eric Horvitz: *Notifications and awareness: a field study of alert usage and preferences*. Em *CSCW*, 2010. 10

[7] Shirazi, Alireza, Niels Henze, Martin Pielot, Dominik Weber e Albrecht Schmidt: *A large-scale assessment of mobile notifications*. abril 2014. 11

[8] John, Nicholas A.: *Sharing and web 2.0: The emergence of a keyword*. Sage Publications, 15(2):167–182, 2012. 12

[9] Yoris A. Au, Robert J. Kauffman: *The economics of mobile payments: Understanding stakeholder issues for an emerging financial technology application*. Electronic Commerce Reserach and Applications, 7(2):141–164, 2008. 12

[10] Taylor, E.: *Mobile payment technologies in retail: a review of potential benefits and risks*. International Journal of Retail and Distribution Management, 44(2):159–177, 2016. 12

[11] Steele, G.; Bracha, G.; Buckley A.; Gosling J.; JOY B: *The Java Language Specification Java SE 8 Edition*. 2015. 16, 19

[12] Naughton, Patrick e Herbert Schildt: *Java: The Complete Reference*. McGraw-Hill, Inc., New York, NY, USA, 1st edição, 1996, ISBN 0078822319. 16, 18, 20

[13] Byous, Jon: *Java technology: An early history*, 2003. `http://java.sun.com/features/1998/05/birthday.html`, Acessado pela última vez em 3 de Outubro de 2019. 16

[14] https://www.oracle.com/technetwork/java/intro-141325.html. 17

[15] *Tiobe index for september 2019*. https://www.tiobe.com/tiobe-index/. 17

[16] *Tiobe index for september 2019: The java programming language*. https://www.tiobe.com/tiobe-index/java/. 17

[17] Sebesta, Robert W.: *Concepts of Programming Languages, Global Edition*. Pearson Education Limited, janeiro 2016, ISBN 978-1-292-10056-2. 19, 20

[18] *Operating system market share: Mobile*. https://www.netmarketshare.com/operating-system-market-share.aspx. 20

[19] *Number of apps available in leading app stores as of 3rd quarter 2019*. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. 20

[20] *Android studio release notes*. https://developer.android.com/studio/releases. 24

[21] *Meet android studio*. https://developer.android.com/studio/intro. 24

[22] W3C: *Extensible markup language (xml) 1.0 (fifth edition)*. 2008. 25

[23] Furgeri, Sérgio: *The role of marking languages in Information Science*. página 16, 2006. 26

[24] *Layouts*. https://developer.android.com/guide/topics/ui/declaring-layout?hl=pt-br. 26

[25] *The rise of rest api*. https://blog.restcase.com/the-rise-of-rest-api/. 28

[26] Fielding, Roy Thomas: *Architectural styles and the design of network-based software architectures*. página 180, 2000. 28

[27] Richardson, Leonard: *RESTful Web Services*. 29

[28] Vargas Agilar, Everton de: *Uma abordagem orientada a serviços para a modernização de sistemas legados*. Universidade de Brasília, Instituto de Ciências Exatas, Departamento de Ciência da Computação, 2014. 29, 30

[29] Jesus Moreira Junior, Caroline Pafiadache Fernando de: *SatisfaÇÃo dos usuÁrios do restaurante universitÁrio da universidade federal de santa maria: Uma anÁlise descritiva*. 2015. 31

[30] *Manual de identidade visual da unb*. http://www.marca.unb.br/manual1.php. 32

[31] *Ru darcy ribeiro*. https://ru.unb.br/index.php/darcy-ribeiro. 33

[32] Souza, Francisca Aparecida de: *Análise do desempenho financeiro e a opinião dos usuários e não-usuários do serviço de alimentação : estudo de caso do restaurante universitário da universidade de brasília - unb*. 2007. 33

[33] http://www.ru.unb.br/. 33, 34

[34] Fernandes, João M. e Ricardo J. Machado: *Requisitos em projetos de software e de sistemas de informação*. Novatec Editora, 2018, ISBN 978-85-7522-660-5. 36

[35] https://material.io/develop/android/components/bottom-navigation-view/. 41

[36] *Cloud firestore*. https://firebase.google.com/docs/firestore?hl=pt-br. 50

[37] *Firebase cloud messaging*. https://firebase.google.com/docs/cloud-messaging?hl=pt-br. 51