Avim: um aplicativo baseado em personagem para auxiliar a dieta alimentar de usuários

Bruno Henrique Bianchi[1], **Fernando Sambinelli**[1]

[1]Instituto Federal de São Paulo (IFSP), Avenida Thereza Ana Cecon Breda, s/n - Vila São Pedro, Hortolândia-SP - Brasil – CEP: 13183-250

brunohbianchi@hotmail.com, sambinelli@ifsp.edu.br

Abstract. *Through the analysis of people's behavior in relation to their diet and the consequences of it, it is observed that immediacy is a factor that is highly capable of demotivating a person during a nutritional reeducation. Therefore, this project consists of the creation of an application for mobile devices that has a character, whose body is modified according to the foods that the user informed that he has ingested. This paper details the development process of this application, which was developed with the incremental model and focused on the Android platform.*

Resumo. *Através da análise do comportamento das pessoas com relação à sua alimentação e às consequências dela, observa-se que o imediatismo é um fator altamente capaz de desmotivar uma pessoa durante uma reeducação alimentar. Diante disso, este projeto consiste na criação de um aplicativo para dispositivos móveis que possui um personagem, sendo que este tem o seu corpo modificado de acordo com os alimentos que o usuário informou que ingeriu. Este trabalho detalha o processo de desenvolvimento deste aplicativo, que foi desenvolvido com o modelo incremental e focado na plataforma Android.*

1. Introdução

O aplicativo Avim tem o intuito de conscientizar as pessoas nutricionalmente através do entretenimento. Esse objetivo pode ser alcançado através da conciliação entre os conceitos de aplicativos de alimentação saudável e jogos, onde o próprio jogador é o personagem. No aplicativo Avim, essa relação será apresentada através de um avatar que será feito pelo próprio usuário e que irá reproduzir esteticamente os efeitos da alimentação ingerida pelo usuário, além de ser complementado por uma ficha apresentando a estimativa do que ocorrerá com o corpo do personagem em prazos variados. Esse aplicativo será desenvolvido para dispositivos *Android*, utilizando a linguagem *Java* e a ferramenta *Android Studio*[1], sendo complementado por um sistema gerenciador de banco de dados (SGBD).

A decisão pela execução deste trabalho se dá em função das vivências do autor, que teve uma grande mudança em seu estilo de vida. Em diversas etapas do processo de adaptação alimentar, sentiu como é difícil manter o foco nas atividades físicas e na alimentação sem que o corpo expressasse de maneira clara os efeitos disso.

Depois de emagrecer, é comum que não haja mais objetivos ou mudanças visíveis, apenas a meta de manter-se como está, o que vai exigir fazer do estilo de vida saudável e equilibrado um hábito para toda a vida. Por outro lado, para voltar a se alimentar mal, o processo ocorre de maneira quase que imperceptível (CORTEZ, 2019).

A justificativa deste projeto é fazer com que o usuário se enxergue no seu personagem (NUTT; RAILTON, 2010), com a finalidade de melhorar a qualidade da sua própria

[1] ANDROID Studio provides the fastest tools for building apps on every type of Android device. *In*: **Android Developers**. [*S.l.*], 2022. Disponível em: https://developer.android.com/studio. Acesso em: 10 jan. 2022.

alimentação. Isso seria facilitado devido ao fato de que os efeitos da dieta alimentar do usuário seriam reproduzidos em seu personagem. Pelo contexto do aplicativo e cenários de utilização, o principal público alvo do aplicativo é o público adolescente.

2. Referencial teórico

O Avim se enquadra em dois conceitos: o de aplicativo e o de jogo. Nesta seção, ambos os conceitos serão detalhados de forma a compreender qual a categoria da aplicação.

2.1. Conceito do aplicativo

Como aplicativo, o Avim se enquadra na categoria de "aplicação para dispositivos móveis nativa". A composição desse nome advém dos seguintes conceitos:

- Aplicação para dispositivos móveis: "Software desenvolvido para ser utilizado em pequenos dispositivos de computação sem fio, como smartphones e tablets, ao invés de computadores desktop ou notebooks. Aplicações móveis são projetadas considerando as demandas e restrições dos dispositivos e também para conseguir vantagens sobre capacidades específicas que eles possuem." (WIGMORE, 2013, n.p.).
- Aplicação nativa: "São desenvolvidos especificamente para uma plataforma, e podem extrair vantagens sobre todos os recursos do dispositivo - eles podem usar a câmera, o GPS, o acelerômetro, a bússola, a lista de contatos, entre outros. Eles também podem incorporar gestos (gestos padrão do sistema operacional ou novos gestos definidos pelo usuário). Aplicações nativas podem usar o sistema de notificações do dispositivo e funcionar offline." (BIDIU, 2016, n.p.).

2.2. Conceito do jogo

Como jogo, o Avim se enquadra na categoria de "jogo casual educativo online" (PEIRCE; WADE, 2011, n.p.). A composição desse nome advém dos seguintes conceitos:

- Jogo casual: jogo que não exige muita compreensão do usuário para que seja aprendido.
- Jogo educacional: jogo desenvolvido com o intuito de educar o usuário.
- Jogo online: jogo que é parcialmente ou completamente jogado através da internet.

2.3. Pesquisa bibliográfica sobre o tema

Na pesquisa bibliográfica, percebeu-se que existem diversos aplicativos referentes à nutrição, e outros referentes à criação de um personagem para o usuário, mas que não existia um aplicativo que interligava essas duas frentes, e alguns exemplos disso são trabalhos correlatos que poderiam servir de fonte para o que viria a ser desenvolvido, tal como o Tecnonutri e o Pou.

- Tecnonutri - Aplicativo no qual o usuário cadastra os alimentos que ingeriu e, com base nisso, o aplicativo calcula os nutrientes ingeridos e quão próxima a soma deles está da quantidade ideal para o usuário, mas que não tem o intuito de fazer qualquer analogia que represente o que acontecerá com o usuário de acordo com a alimentação
- Pou - Aplicativo no qual o usuário cria seu personagem, personaliza-o e, de acordo com as situações impostas pelo usuário para o personagem (quantidade de sono, qualidade da alimentação, cuidados), ele transparece as consequências delas, mas esse

personagem não busca criar um vínculo com a alimentação real do usuário, e sim criar um ‘pet alienígena’.

Ao verificar materiais relativos aos usuários mais assíduos de redes sociais e aos utilizadores do aplicativo a ser desenvolvido, apurou-se que havia uma intersecção que viria a ser o público-alvo do aplicativo, sendo que não foram encontrados aplicativos que possuíam integralmente as funcionalidades que viriam a ser desenvolvidas neste projeto. Visando esse cenário, notou-se que existia muito material disponível para as duas frentes, e que elas seriam interligadas a partir de lógicas. Nesse sentido, a parte da alimentação seria responsável por prover informações para a parte do personagem.

Ao fazer uma visualização mais técnica sobre o que viria a ser desenvolvido, percebeu-se que, para criar um personagem de alta qualidade, seria necessária uma especialização em *frameworks* específicos, tal como a *Unity*. Tendo em vista os prazos e o tempo a ser dedicado para o projeto, optou-se por utilizar imagens básicas para representar o personagem.

3. Metodologia

Durante todo o processo de idealização e desenvolvimento, vários artefatos da engenharia de software foram utilizados para a construção do aplicativo. Nesta seção, serão apresentados os referenciais teóricos dos materiais e métodos utilizados durante o desdobramento do trabalho.

3.1. Modelo de desenvolvimento

Um processo de software é um conjunto de atividades relacionadas que levam à produção de um produto de software. Tendo em vista que o escopo do trabalho era claramente definido, o projeto foi desenvolvido utilizando o modelo em cascata: “Esse modelo [modelo em cascata] considera as atividades fundamentais do processo de especificação, desenvolvimento, validação e evolução, e representa cada uma delas como fases distintas” (SOMMERVILLE, 2011, p. 19).

3.2. Requisitos de software

“Os requisitos de um sistema são descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento” (SOMMERVILLE, 2011, p. 57).

Antes do desenvolvimento, realizou-se o levantamento de requisitos. Essas descrições foram categorizadas em funcionais (prescrevem os serviços do sistema e como devem proceder os dados resultantes desse serviço) e não funcionais (abrangem requisitos de produto, externos ou da organização).

Para auxiliar no mapeamento desses requisitos, foi desenvolvida uma visão de negócio em formato similar a um fluxograma, cujo conteúdo era similar ao que é apresentado na Figura 1.

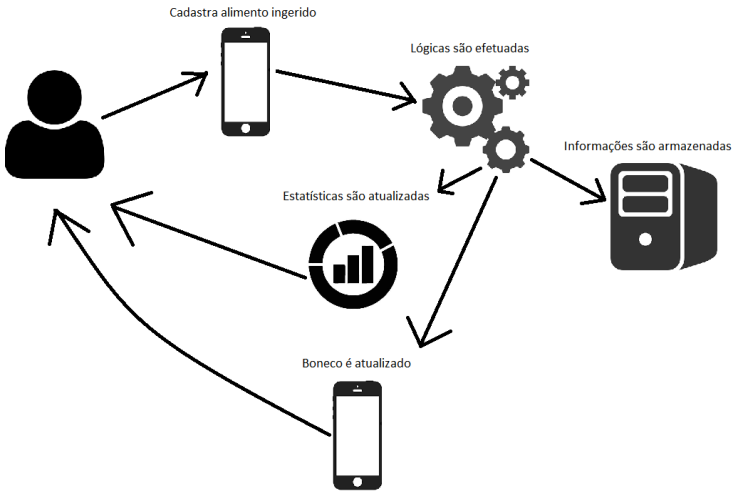

Figura 1. Visão de negócio do aplicativo Avim

3.3. Prototipação

Após a elucidação dos requisitos, criaram-se os primeiros protótipos referentes às telas do sistema em *mockups* de papel[2], com a finalidade de idealizar uma interface capaz de trazer todas as informações para o usuário de forma lúdica.

Essa abordagem costuma ser utilizada em metodologias ágeis, como o *Scrum*[3], mas também pode ser aplicado no desenvolvimento de softwares que utilizam o modelo em cascata. Esta etapa foi realizada com o intuito de ilustrar quais telas deveriam ser implementadas e as funcionalidades que deveriam ser criadas para elas.

3.4. Plataformas e ferramentas de desenvolvimento

O aplicativo foi idealizado para ser utilizado no sistema operacional *Android*, considerando sua liderança no mercado de smartphones no mundo (PIECHARTPIRATE, 2021).

A ferramenta *Android Studio*, fornecida pelo *Google*, foi utilizada para o desenvolvimento do código-fonte, a partir da utilização da linguagem de programação *Java*.

Há uma integração da aplicação com um *webservice rest*[4] que foi desenvolvido na linguagem PHP, responsável pela interação com o banco de dados *MySQL*[5] do aplicativo, possibilitando a tradução das informações para o aplicativo através do formato *JavaScript Object Notation* (JSON)[6].

[2] VIEIRA, T. O que é mockup? *In*: **Tecnoblog**. [*S.l.*], 22 set. 2020. Disponível em: https://tecnoblog.net/responde/o-que-e-mockup/. Acesso em: 20 jan. 2022.
[3] WHAT is Scrum? *In*: **Scrum.org**. [*S.l.*], 2022. Disponível em: https://www.scrum.org/resources/what-is-scrum. Acesso em: 10 jan. 2022.
[4] EQUIPE TOTVS. Arquitetura REST: Saiba o que é e seus diferenciais. *In*: **TOTVS**. [*S.l.*], 23 mar. 2020. Disponível em: https://www.totvs.com/blog/developers/rest/. Acesso em: 20 jan. 2022.
[5] THE WORLD'S most popular open source database. *In*: **MySQL**. [*S.l.*], 2022. Disponível em: https://www.mysql.com/. Acesso em: 10 jan. 2022.
[6] INTRODUCING JSON. *In*: **JSON.org**. [*S.l.*], 2022. Disponível em: https://www.json.org/json-en.html. Acesso em: 10 jan. 2022.

O pacote *Xampp* foi utilizado para o desenvolvimento, levando em conta a capacidade de executar servidores PHP e o banco de dados *MySQL*, fundamentais para o desenvolvimento da aplicação.

4. Desenvolvimento

O desenvolvimento se deu entre julho de 2021 e dezembro de 2021. Desde a concepção deste projeto, existia a intenção de fazer com que a interação do usuário com a aplicação ocorresse por dispositivos móveis. Inclusive, no início, havia a pretensão de utilizar o *framework Flutter*[7], no entanto, notou-se que, devido ao fato de ser um *framework* recente, ocorre com alta reincidência o encontro de soluções já obsoletas na internet, o que multiplicaria o trabalho investido para que uma única funcionalidade atingisse o funcionamento esperado (ou mesmo uma funcionalidade já desenvolvida deixar de funcionar por uma atualização).

Sabendo das dificuldades que seriam enfrentadas para desenvolver uma aplicação com tal nível de complexidade e importância utilizando esse *framework*, o projeto passou a ser desenvolvido utilizando ferramentas simples do *Android Studio*, que possui um ambiente de desenvolvimento baseado na linguagem *Java*, já durante a etapa de desenvolvimento, o que possibilitou um avanço com muito mais qualidade e velocidade. Abaixo, serão discorridas experiências e percepções que ocorreram durante o processo de desenvolvimento do Avim.

4.1. Levantamento de requisitos em softwares correlatos

Na etapa de levantamento de requisitos em softwares correlatos, utilizaram-se aplicativos que possuíam funcionalidades que viriam a ser interessantes para o projeto. Com isso, entendeu-se que os aplicativos relacionados à nutrição possuem, usualmente, uma interface sóbria, mas desempenham-se bem dentro do que se propõem a fazer, enquanto aplicativos de personagens possuem diversas interfaces, o que aparenta ter o intuito de aproximar esse aplicativo do seu público-alvo (usualmente crianças e adolescentes). Baseado nisso, chegou-se aos casos de uso apresentados na Figura 2 para o aplicativo:

[7] BUILD apps for any screen. *In*: **Flutter**. [*S.l.*], 2022. Disponível em: https://flutter.dev/. Acesso em: 10 jan. 2022.

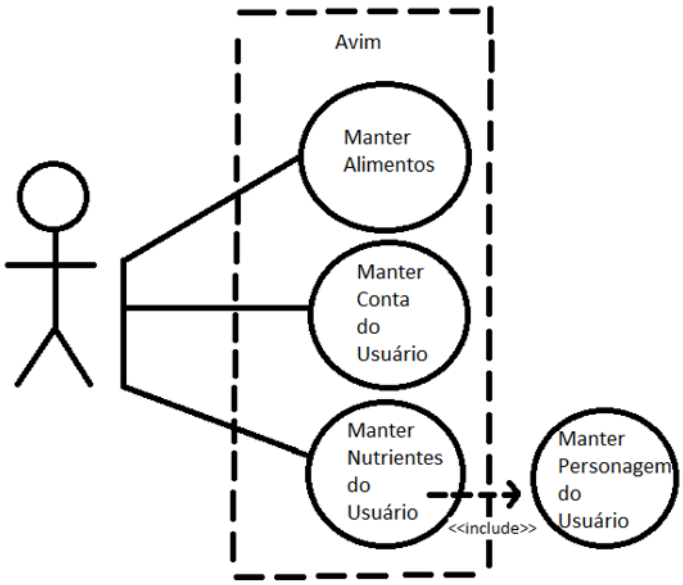

Figura 2. Casos de uso do aplicativo Avim

1. Manter alimentos: os alimentos e seus respectivos nutrientes estão pré-cadastrados no aplicativo, possibilitando que o usuário selecione quais alimentos ele ingeriu.
2. Manter conta do usuário: o usuário pode editar a informação de qual rosto ele gostaria que seu personagem tivesse. Demais informações que pertencem ao personagem, mas que não são diretamente determinadas pelo usuário, como tempo de vida do personagem e estatísticas providas por cálculos referentes à alimentação, estão vinculadas a outras frentes, tendo uma dependência dos hábitos do usuário para que sejam alteradas, ao invés das vontades.
3. Manter nutrientes do usuário: os nutrientes estão diretamente vinculados aos alimentos, mas eles existem com a finalidade de interferir no personagem do usuário, sendo gerados a partir dos cadastros de alimentação do usuário.

4.2. Criação e integração do banco de dados ao projeto

Referente à criação do banco de dados do projeto, a prioridade era a utilização de um banco de dados interno no aplicativo (ou seja, que ficaria armazenado dentro do celular). Nesta etapa, o intuito era utilizar o *SQLite*[8], uma tecnologia vista como mais performática, mas observou-se que, apesar de ser realmente excelente, exigia um nível técnico em aplicações de dispositivos móveis avançado para que sua utilização fosse amigável. Com isso, o banco de dados interno escolhido foi o *ObjectBox*[9], pois, uma vez integrado ao projeto, bastava ter uma classe vinculada a ele para que identificasse e criasse automaticamente as operações de comandos de criação, leitura, atualização e deleção (CRUD) para essa classe. A partir desse momento, o aprendizado e a implementação do banco de dados interno se mostraram mais dinâmicos.

Por outro lado, o *ObjectBox* não é relacional e, como consequência, na classe que representa a intersecção entre os alimentos e os usuários, todos os campos do alimento tiveram

[8] WHAT is SQLite? *In*: **SQLite**. [*S.l.*], 2022. Disponível em: https://www.sqlite.org/index.html. Acesso em: 10 jan. 2022.
[9] HIGH-PERFORMANCE NoSQL database with integrated Data Sync for decentralized Edge Computing. Get fast access to the data you need across Embedded Devices, IoT, and Mobile. *In*: **ObjectBox**. [*S.l.*], 2021. Disponível em: https://objectbox.io/. Acesso em: 10 jan. 2022.

que ser replicados, de forma a evitar a execução de duas *queries* diferentes. Considerando que se trata de um banco de dados interno – ou seja, as informações ali existentes estão apenas vinculadas aos usuários daquele dispositivo –, o banco de dados demoraria muito para ficar pesado, além de ser possível providenciar funcionalidades de backup que seriam capazes de armazenar informações de forma menos detalhada (e, consequentemente, deixariam o banco de dados mais leve). A Figura 3 apresenta o diagrama entidade-relacionamento do banco de dados interno elaborado.

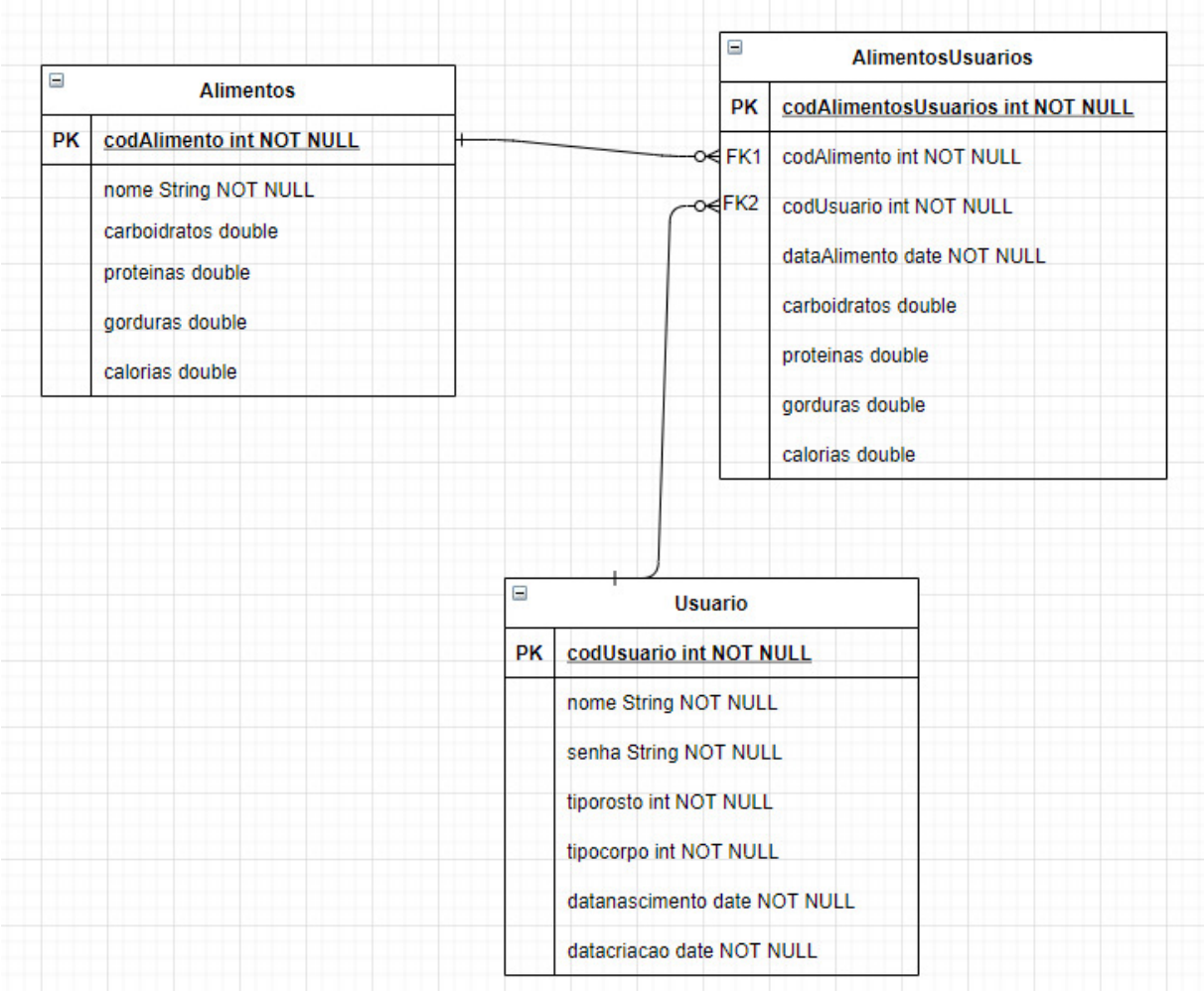

Figura 3. Diagrama entidade-relacionamento

Além do banco de dados interno, também foi criado e integrado ao aplicativo um banco de dados *MySQL*, com a finalidade principal de armazenar usuários (para o futuro desenvolvimento de uma possível funcionalidade capaz de transportar o personagem e suas informações de um dispositivo para outro) e os alimentos que podem ser ingeridos por eles. Essas informações alimentam a *Spinner* (*ComboBox*) de alimentos.

A Figura 4 mostra a arquitetura do aplicativo, onde é possível visualizar melhor a aplicação dos bancos de dados mencionados. O aplicativo tem o dispositivo móvel como principal membro de sua arquitetura e, além disso, ele interage com o banco de dados interno (*ObjectBox*). Tal interação tem o objetivo de armazenar informações dentro do telefone do usuário e com a Application Programming Interface (API) do aplicativo. Essa comunicação tem o propósito de possibilitar a conexão do dispositivo com um SGBD externo (sendo que a API atua como intermediária entre o SGBD e o dispositivo móvel).

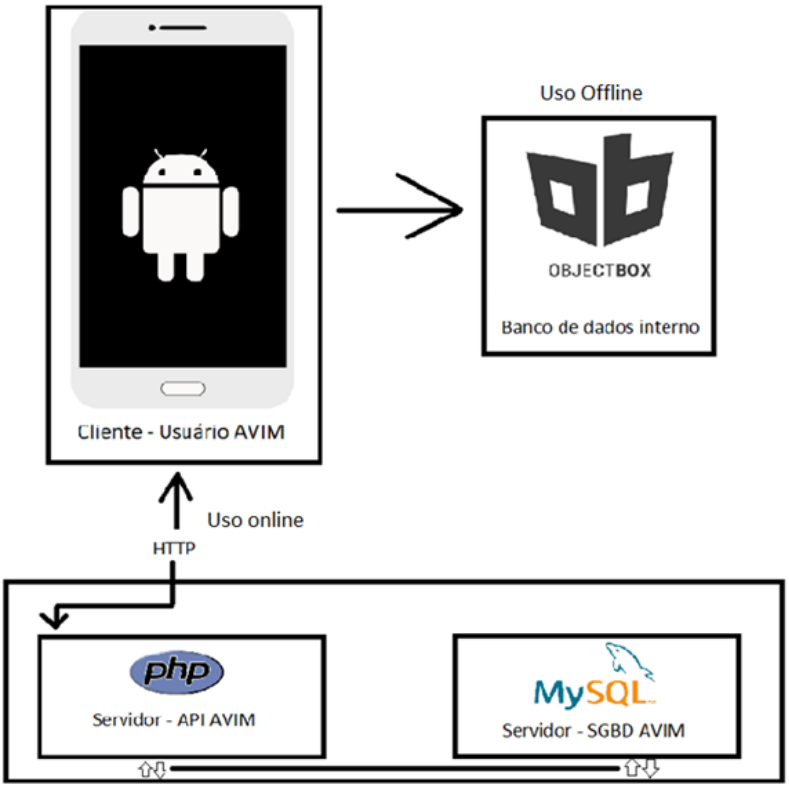

Figura 4. Arquitetura do Avim

4.3. Implementação CRUD de usuário

Dada a natureza simples do *ObjectBox*, esta etapa foi realizada sem grandes dificuldades. Por outro lado, para a integração entre a aplicação e o *MySQL*, foram desenvolvidas funções em PHP para realizar as operações solicitadas, sendo que o fluxo de dados ocorria através do formato JSON, com o tratamento de dados realizado dentro do *Android Studio*.

4.4. Criação das tabelas no banco de dados do cadastro de usuário e das características básicas do boneco

Esta etapa teve como base o diagrama entidade-relacionamento do projeto. Porém, com o avanço da parte lógica, novas necessidades começaram a surgir. Com isso, o diagrama de classes e o diagrama entidade-relacionamento foram alterados e, como consequência, o banco de dados também. Quanto às características básicas do boneco, a partir do momento em que foi decidido que seriam utilizadas imagens para representar o corpo e o rosto do usuário, foram criados os campos referentes a essas informações, sendo ambos do tipo inteiro.

4.5. Desenvolvimento de front-end básico da aplicação

A criação de telas na fase de desenvolvimento foi feita utilizando a funcionalidade "*Drag and Drop*" do *Android Studio*, o que tornou esta etapa bastante intuitiva. Apesar disso, nesse momento, as telas foram feitas apenas para representar o que elas viriam a ser no futuro, não tendo muito tempo investido. Para que a interface ficasse mais interessante, esta etapa viria a

ser replicada no futuro, mas com mais tempo destinado a ela. De forma a ilustrar a situação mencionada (telas de representação), a Figura 5 será apresentada abaixo.

Figura 5. Tela de Alimentação do Usuário em um primeiro momento

Nessa mesma etapa, foi desenvolvido um esboço de o que viria a se tornar o fluxo de telas do aplicativo, e este material serviu de apoio para que o código fosse desenvolvido de forma a conseguir simular a utilização do aplicativo já em suas fases preliminares. Esse fluxo de telas seguiu inalterado até o término do desenvolvimento do aplicativo, e está representado na Figura 6.

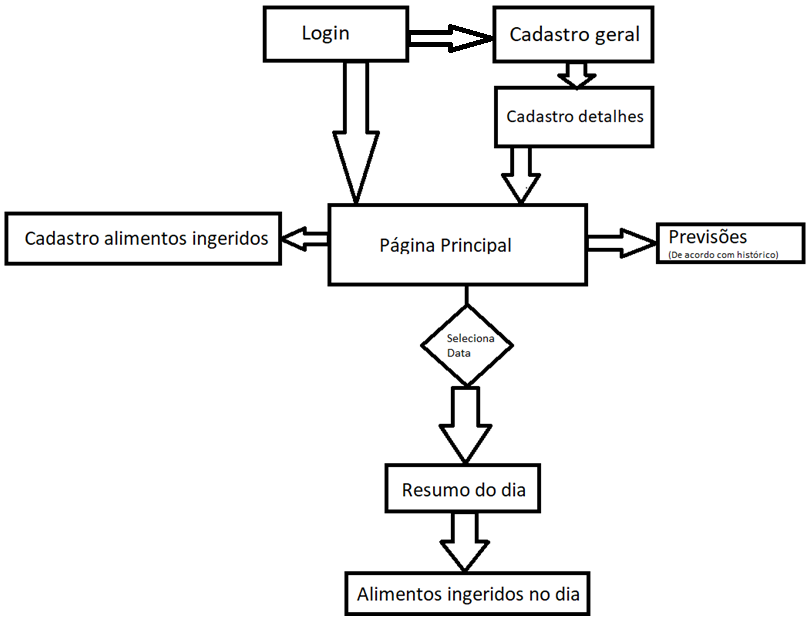

Figura 6. Fluxo de telas do aplicativo Avim

4.6. Implementação da criação do boneco

Inicialmente, a ideia era criar uma aplicação que trouxesse um personagem de alta qualidade e em 3D. Porém, com o avanço dos estudos, percebeu-se que essa implementação requisitaria um alto nível de conhecimento em um *framework* dedicado a isso, como a *Unity*. Considerando a realidade que ronda o desenvolvimento deste projeto, esta etapa vem sendo realizada com a utilização de imagens básicas, com uma relacionada ao rosto do boneco e outra ao corpo, tal como está representado na Figura 7. Para que sejam escolhidas as imagens corretas de rosto e de corpo a serem atribuídas ao boneco, criou-se um campo do tipo inteiro para cada informação na classe Usuário.

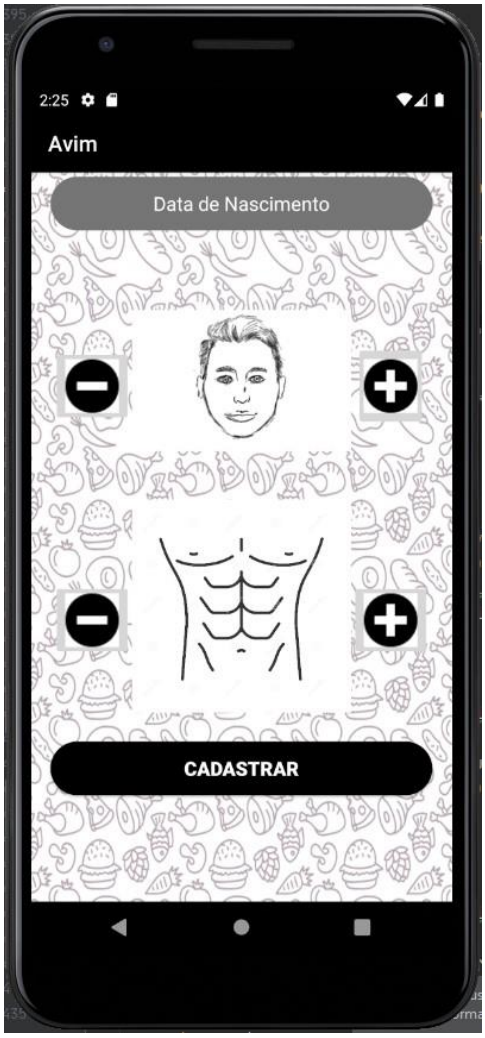

Figura 7. Tela de criação do boneco

A informação do rosto varia de acordo com o gosto do usuário, podendo ela ser selecionada no momento do cadastro ou atualizada posteriormente, enquanto a informação do corpo é, inicialmente, escolhida pelo usuário. Contudo, de acordo com os alimentos que o usuário cadastra no aplicativo e conforme os dias se passam, o número se altera. Esse ponto será tratado mais profundamente no tópico seguinte.

4.7. Implementação de lógicas relacionadas ao reflexo da alimentação do usuário no boneco (engordar, emagrecer)

Para materializar essa lógica, foi preciso implementar uma funcionalidade que calcula quantos dias de vida o boneco tem, e outra funcionalidade para calcular quantas calorias o usuário cadastrou no total, e, posteriormente, calcula-se a soma de calorias dividida pela quantidade de dias de vida do boneco.

Levando em conta que o boneco é composto por imagens básicas, os reflexos da alimentação do usuário no boneco ocorrem por meio de um índice que representa a quantidade de calorias ingeridas pelo usuário, que funciona de forma similar ao Índice de Massa Corporal (IMC) (MAHAN; ARLIN, 1995). A fórmula que gera esse índice está reproduzida na Figura 8. Quanto maior o resultado, mais calorias têm sido ingeridas diariamente em média, sendo que existem pontos de média caloria (como 1200, 1800, 2200, 2800) que, quando são atingidos,

alteram o tipo do corpo do boneco, o que, por sua vez, constitui a imagem a ser utilizada para representar o boneco.

```
public double mediaCal (String usuario) throws ParseException {
    //método que calcula quantas calorias foram ingeridas durante
    //o tempo de vida do personagem
    double caloriastotais = totalCalorias(usuario);
    //método para verificar tempo de vida do personagem
    int diasvida = diasDeVida(usuario);
    //divisão do total de calorias pelo tempo de vida
    double retorno = caloriastotais/diasvida;
    return retorno;
}
```

Figura 8. Fórmula para verificar quantas calorias foram ingeridas, em média, diariamente

4.8. Implementação de lógicas referentes à estimativa do que ocorrerá com o corpo do boneco a partir da alimentação

Para esta etapa, boa parte da lógica utilizada no passo anterior pôde ser reaproveitada, tendo em vista a característica da informação provida por ela, mas, para que a curadoria da qualidade da alimentação fosse mais precisa, não foram utilizadas apenas as calorias, mas também os demais macronutrientes (carboidratos, proteínas, gorduras) e, em sequência, verifica-se o equilíbrio e a quantidade da ingestão desses macronutrientes. Caso esse equilíbrio seja baixo ou a quantidade de nutrientes não seja adequada, uma funcionalidade subsequente indica essa ocorrência; por outro lado, se a alimentação estiver adequada, o usuário receberá a indicação de que ele tem se alimentado corretamente.

4.9. Revisão do banco de dados (criação de novos campos referentes às características da alimentação do boneco)

De acordo com o andamento do projeto, constatou-se a necessidade de armazenar ou gerar mais informações (seja por ser algo fundamental para a lógica a ser desenvolvida ou por outros pontos técnicos, tal como o fato do banco de dados interno não ser relacional), e, para que elas pudessem ser geradas, mais dados viriam a ser captados. Com isso, houve diferenças notáveis entre o modelo entidade-relacionamento inicialmente desenvolvido e o modelo existente hoje, assim como nos campos responsáveis pela geração da informação de tempo de vida do boneco e na necessidade do aumento da classe, que é a intersecção entre o usuário e o alimento, abastecendo o banco de dados interno e possibilitando uma funcionalidade mais dinâmica da aplicação.

4.10. Aprimoramento do front-end da aplicação

Esta etapa foi um ponto-chave para muitas ocorrências do projeto, tanto que foi responsável pela mudança da aplicação de *Flutter* para *Android Studio*. A execução desta etapa foi catalisada por dois fatores, que são:

- A quantidade de informações disponíveis e ainda funcionais para *Android Studio* que estão disponíveis na internet.

- O desenvolvimento da disciplina de Programação de Dispositivos Móveis no Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP).

Ao começar o aprendizado em *Android Studio*, a ferramenta que possibilita "puxar" os atributos visuais a serem inseridos na tela é de grande utilidade, principalmente para se ter uma ideia do que será apresentado nas telas. No entanto, esse recurso não faz grande diferença quando se tem a intenção de criar uma tela efetivamente organizada. Quando essa atividade se iniciou no desenvolvimento do TCC, existia a intenção de fazer telas de qualidade.

Para que essa qualidade fosse alcançada, foi necessário, principalmente, o entendimento de *RelativeLayouts* e *LinearLayouts* no *Android Studio*, além de aprendizados referentes à customização de cores e formatos dos atributos visuais a serem inseridos e das *Constraints* que podem ser colocadas nos atributos visuais, de forma a auxiliar principalmente na disposição deles na tela. Na Figura 9, apresenta-se a tela de alimentação do usuário na versão final do aplicativo. É possível compará-la com a Figura 5, que possuía a mesma tela, mas foi desenvolvida anteriormente a esta etapa.

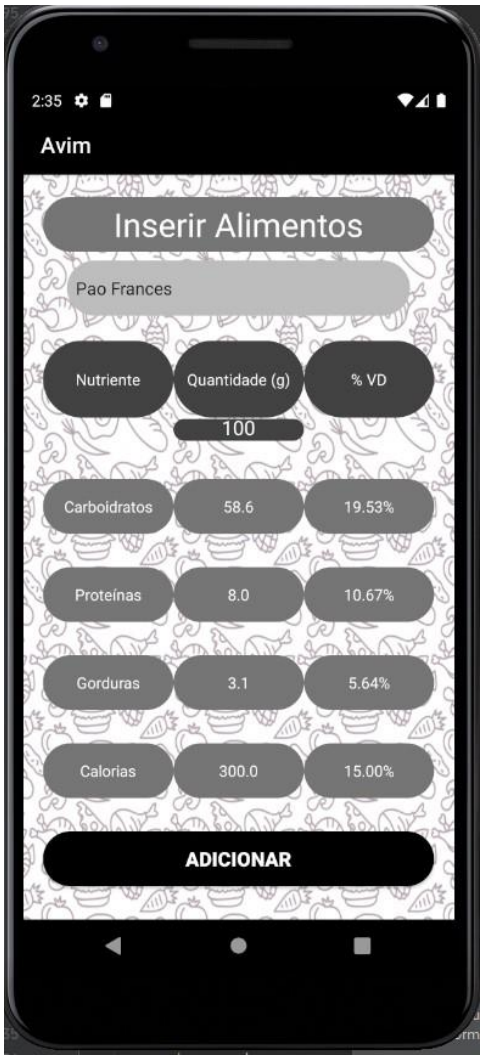

Figura 9. Tela de alimentação do usuário na versão final do aplicativo

4.11. Realização de testes e ajustes (se necessário)

Para a realização desta etapa, utilizou-se o emulador presente no *Android Studio*, e foram realizados testes funcionais de fluxo básico para todas as telas, seguidos da validação dos valores apresentados em tela. Todos os dados se mostraram consistentes e a aplicação foi capaz de armazená-los corretamente ao ser fechada, demonstrando o correto funcionamento do *ObjectBox*. Nenhum ajuste se mostrou necessário dentro desta etapa, tendo em vista que as correções realizadas durante outras etapas do processo de desenvolvimento se mostraram consistentes.

5. Conclusão

Neste artigo, demonstrou-se o processo de desenvolvimento de um aplicativo para auxiliar na dieta alimentar de um usuário em conjunto com as consequências disso representadas em um personagem. O aplicativo foi desenvolvido para a plataforma *Android*, com uso de *webservice* implantado em PHP, e o uso do banco de dados *MySQL*.

O aplicativo foi implementado conforme expectativa e requisitos apresentados neste artigo, tornando possível, para o usuário final, realizar o seu cadastro, cadastrar os nutrientes que ingeriu e verificar os efeitos colaterais disso. Durante o desenvolvimento, notou-se que a implementação de todas as funcionalidades desejadas para o aplicativo levaria muito mais tempo do que poderia ser, de fato, empregado. Isso se deve a diversos fatores, tal como a experiência do desenvolvedor em relação à programação de dispositivos móveis e às restrições de horário existentes durante o andamento deste trabalho. Quando observadas as percepções descritas nos parágrafos anteriores, é notável quanta experiência foi adquirida durante o desenvolvimento do Avim, tanto técnica quanto profissional.

Os itens que eram considerados como todo o processo de desenvolvimento do trabalho durante as fases iniciais do TCC se mostraram apenas parte dele, pois foi preciso complementar informações e torná-las consistentes. Devido a isso, o tempo que teve que ser empregado no desenvolvimento do trabalho se amplificou, sendo que, ainda assim, existem outros requisitos não funcionais que não foram especificados, mas que constavam dentro das projeções do que viria a ser implementado.

Dentre os trabalhos futuros para o aplicativo proposto estão o aprimoramento da qualidade dos personagens, a possibilidade de compartilhar o desenvolvimento do personagem em redes sociais, o desenvolvimento da aplicação para multiplataformas, uma forma de sincronizar os dados locais do aplicativo de forma a transferir os dados para outros dispositivos e uma forma de disponibilizar o menu de alimentos no modo offline.

Tendo em vista o mercado de trabalho atual, os conhecimentos em desenvolvimento de aplicação móvel se mostram um diferencial valioso, dado que o interesse dos usuários por essa opção se mostra uma realidade, e nem sempre os desenvolvedores saem do meio acadêmico preparados para o desenvolvimento de aplicações *mobile* eficientes, ricas em recursos e confiáveis para o usuário final. Cada vez mais se mostra primordial o uso consciente dos recursos disponibilizados pela tecnologia no dia a dia, uma vez que o aplicativo não substitui em nenhum momento o suporte e o monitoramento de nutricionistas capacitados.

Referências

ANDROID Studio provides the fastest tools for building apps on every type of Android device. *In*: **Android Developers**. [*S.l.*], 2022. Disponível em: https://developer.android.com/studio. Acesso em: 10 jan. 2022.

BIDIU, R. Mobile: Native Apps, Web Apps, and Hybrid Apps. *In*: **Nielsen Norman Group**. [*S.l.*], 14 set. 2016. Disponível em: https://www.nngroup.com/articles/mobile-native-apps/. Acesso em: 15 jul. 2021.

BUILD apps for any screen. *In*: **Flutter**. [*S.l.*], 2022. Disponível em: https://flutter.dev/. Acesso em: 10 jan. 2022.

CORTEZ, D. Por que manter-se magro é mais difícil do que perder peso. *In*: **UOL**. [*S.l.*], 11 dez. 2019. Disponível em: https://www.uol.com.br/vivabem/noticias/redacao/2019/12/11/por-que-manter-o-peso-perdido-e-mais-dificil-do-que-emagrecer.htm. Acesso em: 23 nov. 2021.

EQUIPE TOTVS. Arquitetura REST: Saiba o que é e seus diferenciais. *In*: **TOTVS**. [*S.l.*], 23 mar. 2020. Disponível em: https://www.totvs.com/blog/developers/rest/. Acesso em: 20 jan. 2022.

HIGH-PERFORMANCE NoSQL database with integrated Data Sync for decentralized Edge Computing. Get fast access to the data you need across Embedded Devices, IoT, and Mobile. *In*: **ObjectBox**. [*S.l.*], 2021. Disponível em: https://objectbox.io/. Acesso em: 10 jan. 2022.

INTRODUCING JSON. *In*: **JSON.org**. [*S.l.*], 2022. Disponível em: https://www.json.org/json-en.html. Acesso em: 10 jan. 2022.

MAHAN, L. K.; ARLIN, M. T. **Krause**: alimentos, nutrição e dietoterapia. 8. ed. São Paulo: Roca, 1995.

NUTT, D.; RAILTON, D. The Sims: Real Life as Genre. **Taylor & Francis Online**, Londres, v. 6, n. 4, p. 577-592, 24 jun. 2010. Disponível em: https://www.tandfonline.com/doi/abs/10.1080/1369118032000163268. Acesso em: 9 jul. 2021.

PEIRCE, N.; WADE, V. P. Personalised Learning for Casual Games: The'Language Trap'Online Language Learning Game. **ResearchGate**, [*s.l.*], maio 2011. Disponível em: https://www.researchgate.net/publication/228733556_Personalised_Learning_for_Casual_Games_The'Language_Trap'Online_Language_Learning_Game. Acesso em: 10 jan. 2022.

PIECHARTPIRATE. [OC] Mobile Operating System Market Share 2000 - 2021. *In*: **Reddit**. [*S.l.*], jun. 2021. Disponível em: https://www.reddit.com/r/dataisbeautiful/comments/nig4gv/oc_mobile_operating_system_market_share_2000_2021/. Acesso em: 16 nov. 2021.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

THE WORLD'S most popular open source database. *In*: **MySQL**. [*S.l.*], 2022. Disponível em: https://www.mysql.com/. Acesso em: 10 jan. 2022.

VIEIRA, T. O que é mockup? *In*: **Tecnoblog**. [*S.l.*], 22 set. 2020. Disponível em: https://tecnoblog.net/responde/o-que-e-mockup/. Acesso em: 20 jan. 2022.

WHAT is Scrum? *In*: **Scrum.org**. [*S.l.*], 2022. Disponível em: https://www.scrum.org/resources/what-is-scrum. Acesso em: 10 jan. 2022.

WHAT is SQLite? *In*: **SQLite**. [*S.l.*], 2022. Disponível em: https://www.sqlite.org/index.html. Acesso em: 10 jan. 2022.

WIGMORE, I. Mobile App. *In*: **TechTarget**. [*S.l.*], dez. 2013. Disponível em: https://whatis.techtarget.com/definition/mobile-app. Acesso em: 15 jul. 2021.

Documento Digitalizado Público

Artigo do TCC - Versão Final

Assunto: Artigo do TCC - Versão Final
Assinado por: Fernando Sambinelli
Tipo do Documento: Formulário Geral
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Fernando Sambinelli**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 23/03/2022 13:00:10.

Este documento foi armazenado no SUAP em 23/03/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 924488
Código de Autenticação: 722d9454c3