Logotipo, nome da empresa

Descrição gerada automaticamente

**UNIVERSIDADE DE UBERABA**

**SAMUEL LUCAS RODRIGUES DE OLIVEIRA E SILVA**

DESENVOLVIMENTO DE UM APLICATIVO PARA FORMAÇÃO DE HÁBITOS BASEADO EM NEUROCIÊNCIA COMPORTAMENTAL

UBERLÂNDIA – MG 2024

**SAMUEL LUCAS RODRIGUES DE OLIVEIRA E SILVA**

DESENVOLVIMENTO DE UM APLICATIVO PARA FORMAÇÃO DE HÁBITOS BASEADO EM NEUROCIÊNCIA COMPORTAMENTAL

Trabalho apresentado à Universidade de Uberaba, como parte das exigências à conclusão do componente Trabalho de Conclusão de Curso, da 10ª etapa, do curso de graduação em Engenharia da Computação, da UNIUBE, Campus

Uberlândia.

Orientador: Prof. Me. Clênio Eduardo da Silva

UBERLÂNDIA – MG 2024

**SAMUEL LUCAS RODRIGUES DE OLIVEIRA E SILVA**

DESENVOLVIMENTO DE UM APLICATIVO PARA FORMAÇÃO DE HÁBITOS BASEADO EM NEUROCIÊNCIA COMPORTAMENTAL

Trabalho apresentado à Universidade de Uberaba, como parte das exigências à conclusão do componente Trabalho de Conclusão de Curso, da 10ª etapa, do curso de graduação em Engenharia Elétrica, da UNIUBE, Campus Uberlândia.

Aprovado em \_\_ de junho de \_\_\_\_.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Me. Clênio Eduardo da Silva Universidade de Uberaba

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Me. Stéfano Schwenck Borges Vale Vita Universidade de Uberaba

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Me. Júlio Almeida Borges

Universidade de Uberaba

DESENVOLVIMENTO DE UM APLICATIVO PARA FORMAÇÃO DE HÁBITOS BASEADO EM NEUROCIÊNCIA COMPORTAMENTAL

Samuel Lucas Rodrigues de Oliveira e Silva  
 samuellucasrdg@gmail.com

Clênio Eduardo da Silva clenio.silva@uniube.br

# RESUMO

Atualmente, as pessoas têm dificuldades em manter e criar hábitos saudáveis. Isso pode ser explicado pela adição do um novo ambiente na vida das pessoas: a *internet*. Devido a facilidade do acesso ao ambiente digital, o usuário médio desse pode utilizar a internet por até 6 horas diárias, e os conteúdos consumidos nesse refletem nos hábitos diários e na rotina do usuário, seja de maneira positiva ou negativa. Assim, esse trabalho tem como objetivo utilizar o ambiente digital e criar uma aplicação *mobile*, com o intuito de auxiliar o usuário na formação de novos hábitos positivos, utilizando de técnicas baseadas na neurociência comportamental.

**Palavras-chave:** Dispositivos móveis, aplicativo, hábitos.

# ABSTRACT

Currently, people struggle to maintain and create healthy habits. This can be explained by the addition of a new environment in people's lives: the internet. Due to the ease of access to the digital environment, the average user can spend up to 6 hours daily on the internet, and the content consumed reflects on the user's daily habits and routine, either positively or negatively. Thus, this work aims to utilize the digital environment and create a mobile application with the purpose of assisting users in forming new positive habits, using techniques based on behavioral neuroscience.

**Keywords:** Mobile devices, application, habits.

# INTRODUÇÃO

De acordo com o relatório da Global Web Index (2017), usuários da internet estimaram passar cerca de seis horas e meia online diariamente. Esse tempo inclui o aumento do uso de telefones celulares para acessar a internet, que passou de 1 hora e 17 minutos para 2 horas e 30 minutos. Só no Brasil, em 2020, o país registrou 234,07 milhões de acessos móveis, um aumento de 3,26% em relação a 2019, conforme o Relatório do Acompanhamento do Setor de Telecomunicações da Anatel. Isso demonstra que o uso de celulares é algo habitual e que, se utilizado da maneira correta pode auxiliar em diversas áreas da vida do brasileiro, como o tema proposto nesse projeto: a formação de hábitos.

Hábitos, de acordo com Mendelsohn (2019) são moldadores da rotina, processos cognitivos e ações do dia a dia. Além disso, esses também são responsáveis por uma proporção significante do nosso dia a dia, pois quando esses formam a percepção de um gatilho em um determinado ambiente, como um doce encima de uma mesa, realizam uma ação relacionada a tal, como comer o doce. Ademais, esses podem ser criados ao tomar uma ação ou receber um motivador após um gatilho, assim como demonstrado em um experimento laboratorial com primatas (Schultz et al, 1993). Dessa maneira, em tese, é possível criar um gatilho virtual, que fará com que o usuário de tome execute uma ação desejada, consequentemente, formando um hábito.

Além disso, da mesma maneira que hábitos são formados e executados após receberem um motivador, esses podem perder sua força de execução ao receberem uma recompensa menor, por meio da desvalorização da recompensa (Mendelsohn, 2019). Para ratos, a desvalorização da recompensa pode ser feito ao reduzir a quantidade de ração que é relacionada ao puxar de uma alavanca, já para Humanos, essa pode ser feita ao trazer à luz o valor da não execução da ação, por exemplo: explicar à uma pessoa que não fumar evita o desenvolvimento de doenças em curto prazo e evita morte prematura. Assim, isso pode ser feito de maneira automática, por meio de um modelo de linguagem de grande escala e apresentado ao usuário.

Dessa maneira, este projeto visa utilizar do impacto que os celulares têm no dia a dia do brasileiro e desenvolver uma aplicação mobile que auxilia na formação de hábitos positivos, utilizando técnicas de gamificação, criação de um ambiente social para suporte, criação de gatilhos para auxiliar a formação do hábito e criação de um motivador para a desvalorização da recompensa, por meio de um modelo de linguagem de grande escala, com base em estudos de neurociência comportamental. Os objetivos específicos incluem a avaliação de estudos sobre hábitos e o desenvolvimento de APIs para a aplicação móvel.

# FUNDAMENTAÇÃO TEÓRICA

Nesta sessão será explorado de forma completa os conceitos supracitados, trazendo embasamento teórico e técnico dos recursos explorados para desenvolvimento do projeto. Tais como: formação de hábitos, estruturas de projetos, bancos de dados, modelos de linguagem de grande escala e desenvolvimento de código com linguagens: *Dart*, *Java* e *Python*.

## Ações voluntárias

Ações voluntárias são distintas, e definidas por teoristas do aprendizado, em duas variedades: ações orientada a um objetivo e hábitos (BOUTON, 2021).

### Ação orientada a um objetivo

Uma ação pode ser descrita como um comportamento instrumental, com a função de levar uma pessoa a um objetivo, dado que essa esteja mentalmente engajada durante o processo da ação, ou seja, com a representação do objetivo em memória (BOUTON, 2021).

Este tipo de ação é o que permite uma pessoa a tomar decisões que estão fora de sua rotina e que não irão trazer nenhum tipo de benefício a curto prazo. Normalmente essas ações têm uma baixa carga dopaminérgica e demandam um grande esforço mental para serem concluídas. Algumas dessas ações são, por exemplo: uma pessoa que não tinha o hábito de estudar, começa a estudar com o objetivo de passar em um concurso; uma pessoa que não tinha o hábito de ir à academia, começa a ir à academia e se alimentar melhor com o objetivo de se tornar mais saudável.

### Hábito

Um hábito pode ser caracterizado como uma rotina, comportamento ou processo cognitivo que começa espontaneamente, mas é repetido de maneira automática como resultado de uma experiência própria. Ademais, hábitos são dependentes de contextos, pois eles se tornam mais fortes por meio de repetição e associações a dicas do ambiente, de maneira que esses se tornam dependentes de pistas relevantes (MENDELSOHN, 2019).

Além disso, sabe-se que um hábito não requer muito valor cognitivo, visto que esses são performados de maneira rápida e automática, além disso eles tendem a ser inflexíveis. Dessa maneira, hábitos servem um propósito crítico em tornar nosso comportamento mais eficiente, reduzindo a carga mental em decisões de menor importância, o que faz com que tomadas de decisões mais importantes tenham mais energia mental disponível (MENDELSOHN, 2019).

Por fim, sabe-se que o hábito é um comportamento que pode, eventualmente, levar uma pessoa a um objetivo, mas é performado de maneira automática, sem que se tenha um objetivo em “mente”. Uma ação orientada a objetivo pode se tornar um hábito com repetição extensiva e prática (BOUTON, 2021).

### Dopamina

De acordo com Cleveland Clinic[[1]](#footnote-1), a dopamina é um neurotransmissor produzido no cérebro que tem a função de agir como o “centro de recompensa” e tem participação ativa em diversas funções corporais, como memória, movimento, motivação e humor.

Na formação dos hábitos, a dopamina tem uma função importante, pois esse hormônio é o responsável pela motivação necessária para a tomada das decisões de ações orientadas a hábitos. O reconhecimento da recompensa ao executar a ação é essencial para que a dopamina seja liberada antes da ação nas próximas vezes. Um hábito é formado pela repetição da ação, que por sua vez também aumenta a quantidade de dopamina liberada previamente à ação (SCHULTZ et al, 1993).

## Linguagens de Programação

Uma linguagem de programação é um conjunto de regras e sintaxes que permite aos programadores escreverem instruções que um computador pode entender e executar. Essas linguagens são usadas para criar *software*, *aplicativos*, *scripts* e outros tipos de sistemas computacionais.Nesse projeto foram utilizadas as linguagens de programação: *Dart*, *Java* e *Python*.

### *Dart*

### *Dart* é a linguagem de programação onde o framework *Flutter* é escrito e compilado. Essa linguagem foi primeiro lançada em 14 de novembro de 2013 e teve sua segunda versão lançada em agosto de 2018. Dart foi popularizado principalmente após o lançamento do *Flutter*, atualmente seu *framework* principal.

### *Flutter* é um framework de código aberto desenvolvido pelo Google, com o objetivo de criar telas modernas, de compilação nativa e multi plataforma com apenas um código base. Esse foi lançado em maio de 2017 e no ano de 2022 teve sua terceira versão lançada.

O *Flutter* consiste tanto de uma linguagem de criação de interface de usuário, quanto de um motor de renderização. Dessa maneira, aplicativos *flutter* podem ser escrito em apenas uma base de código e esses podem ser renderizados para diversas plataformas.  
 Dessa maneira, o flutter será a linguagem utilizada nesse projeto para o desenvolvimento das telas *frontend[[2]](#footnote-2)*, visto que com essa é possível abranjer um maior número de dispositivos com apenas uma base de código.

### *Java*

*Java* é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems, e lançada em 1995. Projetada para ser simples, robusta e de uso geral, *Java* permite a criação de aplicativos portáteis e seguros, que podem ser executados em qualquer plataforma que possua uma Java Virtual Machine (JVM). Uma das características mais destacadas de *Java* é o seu conceito de "*Write Once, Run Anywhere*" (WORA)[[3]](#footnote-3), que garante a compatibilidade de código em diferentes sistemas operacionais sem a necessidade de modificações.

Além disso, Java é amplamente utilizada em desenvolvimento de aplicações empresariais, sistemas web, aplicativos móveis (particularmente para a plataforma Android), e grandes sistemas de *backend[[4]](#footnote-4)*, devido à sua alta performance, escalabilidade e um ecossistema rico de bibliotecas e frameworks. A linguagem também se destaca por seu gerenciamento automático de memória, através do coletor de lixo (*garbage collector*), que ajuda a prevenir vazamentos de memória e melhora a segurança e estabilidade das aplicações.

Dessa maneira, nesse projeto o desenvolvimento do *backend* será realizado em sua maioria em *Spring[[5]](#footnote-5), que é um* *framework* desenvolvido em *Java*. Pois essa é uma linguagem segura e resiliente.

### Python

*Python* é uma linguagem de programação de alto nível, interpretada e de propósito geral, desenvolvida por Guido van Rossum e lançada pela primeira vez em 1991. Caracteriza-se por sua sintaxe clara e concisa, que promove a legibilidade do código e a facilidade de manutenção.

Além disso, Python é amplamente utilizado em diversas áreas, como desenvolvimento web, automação de scripts, análise de dados, inteligência artificial e aprendizado de máquina, graças à sua vasta biblioteca padrão e ao rico ecossistema de bibliotecas e frameworks de terceiros. Ademais, a linguagem adota paradigmas de programação orientada a objetos, funcional e imperativa, oferecendo flexibilidade e poder aos desenvolvedores.

Assim, nesse projeto haverá o desenvolvimento de um serviço *backend* em *FastAPI[[6]](#footnote-6),* que é um framework desenvolvido em *Python*. Isso pois, além de ser uma linguagem resiliente para o desenvolvimento de aplicações *backend,* essa tem uma biblioteca ofertada pela OpenAI, que facilita o desenvolvimento de requisições para o modelo de linguagem de grande escala desenvolvido por eles.

## Modelo de Linguagem de Grande Escala

Os modelos de linguagem de grande escala, como o GPT-4 desenvolvido pela OpenAI, representam um avanço significativo na área de processamento de linguagem natural (PLN). Esses modelos são treinados em enormes quantidades de dados textuais, utilizando redes neurais profundas, especificamente arquiteturas de transformadores, para capturar nuances complexas da linguagem humana.

Ademais, esses modelos são capazes de gerar texto coerente, responder a perguntas, traduzir idiomas e até mesmo criar conteúdo original. Assim, nesse projeto, serão feitas requisições a API da OpenAI, para que sejam gerados conteúdos específicos para cada usuário da aplicação.

## Banco de Dados

Um banco de dados é um sistema organizado para armazenar, gerenciar e recuperar informações de forma eficiente e estruturada. Entre os diversos sistemas de gerenciamento de banco de dados (SGBD) disponíveis, o MySQL destaca-se como uma solução robusta e amplamente utilizada, especialmente em aplicações web.

Portanto, nesse projeto, o MySQL será utilizado para que as informações do usuário sejam salvas, pois esse é conhecido por sua alta performance, escalabilidade e compatibilidade com diversas plataformas e linguagens de programação.

## Estrutura de Projetos

Na comunidade do desenvolvimento de software, discussões sobre conceitos e regras para garantia da qualidade do código são recorrentes e pertinentes. Durante a evolução da literatura da área de Engenharia de Software diversos conceitos se tornaram relevantes para o desenvolvimento.

### Código Limpo – *Clean Code*

O conceito de código limpo, descrito inicialmente por Robert C. Martin, criador e fundador da empresa Clean Coders e um dos 17 signatários originais do manifesto ágil. Esse termo foi citado e popularizado em sua obra Código Limpo: Habilidades práticas de Agile Software, no ano de 2008. Nela um conjunto de diretrizes e boas práticas eram definidas para que fossem aplicadas no dia a dia do desenvolvedor.

Assim, nesse projeto, princípios e diretrizes definidos na obra serão aplicados, como: nomenclaturas significativas, ou seja, nomes descritivos e sem abreviaturas; funções pequenas e objetivas; evitar código replicado, seguindo o padrão DRY (*Don’t repeat yourself*)[[7]](#footnote-7); simplicidade, assim como descrito em KISS (*Keep it Stupid Simple)[[8]](#footnote-8)* e YAGNI (*You aren’t gonna need it*)[[9]](#footnote-9); e separação de conceitos, mantendo classes relacionadas juntas e evitando a dependência entre módulos.

### *Bloc*

O padrão *Bloc* (*Business Logic Component)[[10]](#footnote-10)* é uma abordagem arquitetural utilizada no desenvolvimento de aplicações Flutter, que permite a separação entre a interface do usuário e a lógica de negócios. Inspirado nesse padrão, uma biblioteca, também chamada *Bloc*, foi criada com o intuito de gerenciar estado previsível em Dart. No *Bloc*, ações de usuários são registradas como eventos e são enviados para a camada de lógica da aplicação, essa que, por sua vez, processa esses eventos e emite estados para que a camada de interface do usuário consiga demonstrar o valor correto.

Dessa maneira, esse padrão foi o escolhido para o desenvolvimento do *frontend* do código, pois essa separação clara entre as responsabilidades contribui para um código mais modular e de fácil manutenção, assim como demonstrado na imagem 1.

**Figura 1 –** Arquitetura Bloc da aplicação *frontend*

A diagram of a program

Description automatically generated

Fonte: Autoria própria, 2024.

## *API*

De acordo com a AWS[[11]](#footnote-11) (Amazon Web Services), uma *API* (*Application Programming Interface)[[12]](#footnote-12)* é uma interface que disponibiliza o contrato de serviço entre duas aplicações, essas que são qualquer software de funções distintas. Esse contrato define como ambas as aplicações respondem a determinadas requisições.

Ademais, *API*s podem funcionar de quatro maneiras: SOAP, onde é seguido o protocolo de acesso a objetos simples; RPC, onde é seguido o padrão de chamadas de procedimento remoto; WebSocket, onde é oferecido suporte à comunicação bidirecional entre aplicativos cliente e o servidor; e REST, que são as *API*s mais populares e flexíveis do mercado atualmente.

Para esse projeto, o *backend*, que consta de *API*’s desenvolvidas em *Spring* e *FastAPI*, se comunicarão utilizando do padrão REST com o frontend e com API’s externas, como a *API* do modelo de linguagem de grande escala da OpenAI.

## Conteinerização

A conteinerização é uma tecnologia que permite empacotar uma aplicação e todas as suas dependências em um único contêiner, garantindo que ela possa ser executada de forma consistente em qualquer ambiente.

Os contêineres compartilham o mesmo *kernel[[13]](#footnote-13)* do sistema operacional, mas operam em ambientes isolados, oferecendo a vantagem de serem mais leves e eficientes em comparação com as máquinas virtuais. Esta abordagem promove uma maior portabilidade e escalabilidade das aplicações, facilitando o gerenciamento e a orquestração de múltiplos contêineres através de ferramentas como Docker Compose.

Assim, nesse projeto, para que seja possível a execução das aplicações em diversos ambientes, será utilizado do Docker para isolar o *backend* e do Docker Compose para orquestrar e gerenciar as múltiplas imagens.

## Padrões de Segurança

Os padrões de segurança de autenticação são práticas e protocolos estabelecidos para verificar a identidade de usuários e sistemas, garantindo acesso seguro a recursos e dados sensíveis. Entre os principais padrões, destacam-se o OAuth, que permite delegação segura de acesso através de tokens de autorização sem compartilhar credenciais; o OpenID Connect, que estende o OAuth adicionando autenticação baseada em identidade federada; e o SAML (Security Assertion Markup Language), utilizado para autenticação e autorização em ambientes corporativos através da troca de declarações de segurança.

Nesse projeto, será utilizada a autenticação com JSON Web Tokens (JWT), que permite a verificação de identidade e a troca de informações de maneira segura e compacta entre as partes, utilizando um token assinado digitalmente.

## Sistema de Controle de versão

Um sistema de controle de versão (VCS) é uma ferramenta essencial no desenvolvimento de software que permite aos desenvolvedores rastrear e gerenciar alterações no código-fonte ao longo do tempo, facilitando a colaboração e a manutenção do histórico de modificações.

O *Git[[14]](#footnote-14)*, um dos sistemas de controle de versão mais populares, oferece um modelo distribuído onde cada desenvolvedor possui uma cópia completa do repositório, permitindo operações rápidas e offline. GitHub, uma plataforma baseada em *Git*, expande essas funcionalidades ao fornecer um ambiente colaborativo na nuvem, onde equipes podem hospedar repositórios, revisar código, gerenciar projetos e integrar ferramentas de CI/CD (Integração Contínua/Entrega Contínua).

Nesse projeto, a plataforma GitHub será utilizada para a hospedagem do repositório da aplicação, no padrão *“monorepo”[[15]](#footnote-15),* para que seja garantida a consistência e segurança do versionamento.

# METODOLOGIA

Para o desenvolvimento desse projeto foi necessário fazer o levantamento dos requisitos da aplicação, definir a arquitetura de comunicação entre os serviços, definir o padrão da estrutura de arquivos dentro de cada serviço, definir relacionamentos no banco de dados e criar o *design* da aplicação que será utilizado na tela do usuário. Após isso, foi necessário o desenvolvimento de cada aplicação seguindo as melhores práticas definidas no mercado. Dessa maneira, a seguir serão apresentadas as etapas do desenvolvimento do projeto

* 1. **Levantamento de requisitos**

A etapa de levantamento de requisitos é o momento onde o desenvolvedor responsável pela aplicação, sendo, no caso desse projeto, o acadêmico, faz questionamentos sobre o objetivo desejado e como esse pode ser atingido de maneira enxuta e eficiente. No caso desse projeto, o objetivo é o desenvolvimento de um aplicativo para formação de hábitos baseado em conceitos da neurociência comportamental.

Para atingir tal objetivo, primeiro foi necessário definir em qual ambiente seria desenvolvida a aplicação que o usuário teria acesso. Para isso foi feita uma pesquisa sobre qual o sistema operacional com maior número de usuários atualmente, para que o aplicativo abrangesse o maior número de pessoas. Assim, o sistema operacional selecionado foi o *Android*, pois, de acordo com Lisboa (2022), esse sistema é utilizado por 44,66% dos dispositivos, no mercado, atualmente. Por fim, pesquisas sobre desenvolvimento mobile foram realizadas e notou-se que a linguagem de programação *Dart*, junto do framework do *Flutter*, seria uma boa escolha pois esse, além de abranger o sistema operacional *Android*, também abrangeria o sistema *IOS*.

Após definido o ambiente em que o aplicativo seria desenvolvido e executado, foi necessário definir quais seriam os recursos principais oferecidos pelo aplicativo. Para gerar a desvalorização da recompensa, seria necessário a geração do texto motivador de maneira automática, para tal seria preciso um modelo de linguagem de grande escala, e, como criar esse necessitaria de uma grande quantidade de dados e tempo, foi definido que o modelo utilizado seria o GPT 3.o oferecido pela API da empresa OpenAI.

Como hábitos são dependentes de contexto, foi proposto o desenvolvimento de um ambiente social, em que o usuário pode interagir com outros usuários e propor desafios entre si, assim gerando maior motivação para a execução do hábito. Além disso, para gerar maior engajamento, foi proposto a gamificação de áreas do aplicativo, pois essa estratégia, de acordo com Luis von Ahn (2023), faz com que o usuário retorne mais vezes à aplicação, o que, novamente, aumentaria as chances da formação de um hábito.

Por fim, foi proposto o desenvolvimento de um gatilho para recordar o usuário da execução do hábito. Assim, a solução proposta para esse foi a criação de uma notificação *push[[16]](#footnote-16)*, que diariamente lembraria o usuário de entrar no aplicativo, executar seu hábito e o registrar.

## Arquitetura dos serviços

Definir a arquitetura dos serviços, antes do desenvolvimento do código da aplicação, é importante para que o aplicativo seja desenvolvido de maneira enxuta e eficiente. Como esse projeto foi desenvolvido por apenas uma pessoa, foram feitos levantamentos sobre a complexidade requerida e, assim, a arquitetura principal para o *backend* da aplicação foi definida como monolito[[17]](#footnote-17), desenvolvido na linguagem *Java*, com um serviço secundário, desenvolvido na linguagem *Python*. Esses seriam responsáveis por prover os dados necessários para a aplicação *frontend*, desenvolvida em *Flutter*.

O monolito desenvolvido em *Java* seria responsável pela autenticação, funções do usuário, relacionamentos de amizades, criação de desafios, conexão com o serviço secundário e persistência de dados no banco de dados. Já o serviço secundário, desenvolvido em *Python*, seria responsável por chamadas na *API* do modelo de linguagem de grande escala do GPT 3.o, desenvolvido pela OpenAI, por meio de uma biblioteca provida por essa.

O *frontend* desenvolvido em *Flutter* faz a conexão com os serviços desenvolvidos no backend e utiliza os dados providos por esse para gerar telas interativas ao usuário. Abaixo, a figura 2 ilustra a arquitetura completa da aplicação:

**Figura 2:** Arquitetura da aplicação.

A diagram of a computer program

Description automatically generated

**Fonte:** Autoria própria, 2024.

## Estrutura de arquivos

Nessa seção serão discutidas as decisões tomadas para as estruturas de arquivos em cada serviço da aplicação.

* + 1. ***Frontend***

Para o desenvolvimento do aplicativo que será executado no celular do usuário, foi escolhido o padrão apresentado na documentação da biblioteca Bloc do *Flutter*, assim como apresentado na figura 3:

**Figura 3:**Exemplo de estrutura de projeto bloc.

A diagram of a data layer

Description automatically generated

**Fonte:** < <https://bloclibrary.dev/pt-br/tutorials/flutter-todos/>>. Acesso em 14 de jun. de 2024.

Esse padrão tem como objetivo separar os arquivos de código do projeto em três camadas: camada de recursos do usuário, camada de lógica de negócio e camada de dados; com o objetivo de manter o código desacoplado e de fácil manutenção.

**3.3.2** ***Backend***

Para o desenvolvimento dos serviços desenvolvidos no *backend* foi decidido que esses seguiriam o padrão de *controller, service* e *repository*. Assim como demonstrado na figura 4:

**Figura 4:** Padrão estrutural de Controller, Service e Repository.

A green and blue box with text

Description automatically generated with medium confidence

**Fonte:** Autoria própria, 2024.

Nesse padrão, a camada *controller* é responsável por expor *endpoints* para a chamada de requisições e lidar com cada requisição que ocorre em determinado ponto, essa camada conversa apenas com a camada de *service* para que não ocorram problemas de segurança ou problemas de acoplamento. A camada *service* é responsável por lidar com as lógicas da aplicação, sejam essas lógicas de criação ou deleção, validações e até mesmo lógicas de erro interno, essa camada utiliza de uma ou mais classes de dentro da camada *repository* para se buscar e adicionar informações de uma fonte externa. A camada *repository* é a camada responsável por se comunicar com fontes externas, sejam essas APIs ou o banco de dados.

* 1. **Banco de Dados**

Nesse projeto, como foi proposta a criação de um ambiente social para que diversos usuários interajam, a persistência dos dados não pode ser local, ou seja, no celular do usuário. Portanto, nesse projeto foi utilizado um banco de dados MySQL, que foi responsável pela persistência de todas as informações de um usuário, desde qual a motivação do usuário, até quais são os amigos desse.

Como o MySQL é um banco de dados relacional, antes de o utilizar, é necessário criar um diagrama de relacionamentos das tabelas para que, durante a fase de desenvolvimento, não haja conflitos de tabelas. O diagrama de relacionamentos é demonstrado na figura 5 abaixo.

**Figura 5:** Esquema do banco de dados do aplicativo.

A diagram of a user flow

Description automatically generated

**Fonte:** Autoria própria, 2024.

* 1. **Fluxos do aplicativo**

Nessa seção serão demonstrados os possíveis fluxos que um usuário pode tomar dentro a aplicação e como esses foram desenvolvidos.

* + 1. **Fluxo de autenticação**

Como no projeto foi proposta a criação de um ambiente social, em que diversos usuários interagem entre si, é necessária, também, a criação de uma forma do usuário se autenticar e se identificar dentro do aplicativo. Para tal, foi desenvolvido no monolito em *Java* um módulo de autenticação, em que, quando um usuário se registra ou se autentica, um token JWT é gerado e retornado ao cliente para que esse possa ter acesso aos *endpoints* criptografados. Este token é gerado de maneira automática, a identificação é feita por meio do e-mail do usuário que é salvo dentro do token e, para garantir a segurança e evitar fraudes desse, esse é assinado por meio de uma chave secreta que é armazenada dentro do arquivo de configuração do monolito.

Já no frontend, caso o usuário não tenha um token salvo em seu aparelho, esse verá uma tela inicial onde poderá se registrar ou se autenticar. Caso esse tenha um token salvo, ele será redirecionado à tela principal da aplicação, como demonstrado nas imagens abaixo.

**Figura 6:** Usuário não autenticado. **Figura 7:** Tela inicial da aplicação.

**A screenshot of a phone screen

Description automatically generated A screenshot of a cell phone

Description automatically generated**

**Fonte:** Autoria própria, 2024. **Fonte:** Autoria própria, 2024.

* + 1. **Fluxo de registro de hábito**

Nesse projeto, foi definido o desenvolvimento de um motivador, seja para gerar uma ação orientada a um objetivo ou para desvalorizar a recompensa associada a um vício. Para tal, é necessário que o usuário registre qual é o motivador que ele deseja, esse será enviado para o monolito em *Java* que, por sua vez, fará uma requisição ao serviço secundário em *Python* e fará outra requisição à *API* do modelo de linguagem de grande escala GPT 3.5 da OpenAI. Ao receber a requisição, a API do modelo de linguagem de grande escala irá gerar uma frase diária que será salva no banco de dados e apresentada na tela do usuário assim como demonstrado na figura 7.

Caso o usuário não tenha registrado o motivador e esteja tentando utilizar do aplicativo, uma tela com um formulário será apresentada a esse para que possa fazer o registro de seu objetivo, assim como demonstra a figura abaixo.

**Figura 8:** Tela de registro de hábito.

A screenshot of a phone

Description automatically generated

**Fonte:** Autoria própria, 2024.

Além disso, para que haja maior engajamento do usuário no aplicativo e gere futuras visitas, o usuário pode registrar se esse teve sucesso em manter o hábito naquele dia. Esses registros são salvos em um banco de dados e salvos também na tela do usuário para que esse possa visualizar quantos dias em sequência ele teve sucesso, assim como demonstrado na figura 7.

* + 1. **Fluxo de Amigos**

Nesse projeto, foi definido que seria criado um ambiente social para suporte e motivação dos usuários, para tal foi criado então um sistema de amizade dentro do aplicativo. Nesse, um usuário pode enviar pedidos de amizade, aceitar ou recusar pedidos de amizade e desafiar amigos para desafios de formação de hábito. Para isso, o monolito é responsável pela verificação e validação de amizades e pela persistência desses no banco de dados. Já na tela do usuário, o *frontend* executa requisições ao monolito e gera a interface de acordo com as informações recebidas do servidor, assim como demonstrado nas figuras a seguir.

**Figura 9:** Tela de Amigos. **Figura 10:** Tela de Pedidos de Amizade.

A screenshot of a phone

Description automatically generated A black square with white dots

Description automatically generated with medium confidence

**Fonte:** Autoria própria, 2024. **Fonte:** Autoria própria, 2024.

* + 1. **Fluxo de Desafios**

Nesse projeto, foi definido que seria criado um ambiente social para suporte e motivação dos usuários, para tal, além do desenvolvimento do ambiente de amigos, foi desenvolvido um ambiente de desafios, onde usuários podem se desafiar a seguirem um protocolo de hábitos durante um tempo determinado de dias. Essa parte do projeto foi desenvolvida com o objetivo de ser *gamificada*, para engajar o usuário pela maior quantidade de tempo, assim como explicado por Luis Vohn Ahn (2023).

O monolito tratará de criar, deletar e manter os registros dos participantes de cada desafio e deverá enviar os dados processados ao *frontend*. Já o *frontend* deverá utilizar dessas informações para criar as interfaces para o usuário, como demonstrado nas figuras abaixo.

(Essa parte do projeto não está pronta, quando pronta colocar imagens aqui).

* + 1. **Fluxo da Notificação Push**

Esse projeto propôs a criação de um gatilho para que o usuário associe uma ação a esse. Para tal, foi criado um sistema de notificações push, onde em determinado horário do dia, o monolito envia uma notificação ao sistema do usuário com uma mensagem customizada, para que esse execute a ação do hábito.

(Essa parte não está pronta, quando pronta colcoar imagens)

* 1. **Conteinerização**

Por fim, para garantir que o backend da aplicação possa ser executado em qualquer máquina, esse foi conteinerizado em um imagens Docker. Para orquestrar as diversas imagens docker, tanto dos serviços monolito e secundário, quanto do banco de dados, foi utilizado o Docker compose.

Dessa maneira, fatores como sistema operacional ou versão da tecnologia presente no computador do sistema que executará o *backend* não afetarão a execução desse. O arquivo de execução do docker compose é demonstrado na figura a seguir.

**Figura 11:** Arquivo DockerCompose.

A screenshot of a computer program

Description automatically generated

**Fonte:** Autoria própria.

# RESULTADOS E DISCUSSÕES

# CONCLUSÃO

# REFERÊNCIAS

BOUTON, M. E. **Context, attention, and the switch between habit and goal-direction in behavior**. Learning & behavior, v. 49, n. 4, p. 349–362, 2021. Disponível em: https://doi.org/10.3758/s13420-021-0048f8-z. Acesso em: 09 jun. 2024.

BRASIL. **Brasil tem mais de 234 milhões de acessos móveis em 2020**. Ministério das Comunicações, 2021. Disponível em: https://www.gov.br/pt-br/noticias/transito-e-transportes/2021/04/brasil-tem-mais-de-234-milhoes-de-acessos-moveis-em-2020. Acesso em: 9 jun. 2024.

GLOBAL WEB INDEX**. Digital vs Traditional Media Consumption: analyzing time devoted to online and traditional forms of media at a global level, as well as by age and across countries**. Insight Report, Q1 2017. Disponível em: https://www.gwi.com/hubfs/Digital\_vs\_Traditional\_Media\_Consumption.pdf. Acesso em: 09 jun. 2024.

LISBOA, Alveni. **Qual é o sistema operacional mais usado do mundo?** Editado por Douglas Ciriaco. Canaltech, 23 ago. 2022. Disponível em: <https://canaltech.com.br/software/qual-e-o-sistema-operacional-mais-usado-do-mundo-223507/>. Acesso em: 09 de jun. 2024.

MENDELSOHN, A. I. **Creatures of habit: the neuroscience of habit and purposeful behavior**. Biological Psychiatry, v. 85, n. 11, p. e49–e51, 2019. Disponível em: https://doi.org/10.1016/j.biopsych.2019.03.978. Acesso em: 9 jun. 2024.

OSTLUND, S. B.; BALLEINE, B. W. **On habits and addiction: an associative analysis of compulsive drug seeking**. Drug discovery today. Disease models, v. 5, n. 4, p. 235–245, 2008. Disponível em: https://doi.org/10.1016/j.ddmod.2009.07.004. Acesso em: 09 jun. 2024.

Schultz, W., Apicella, P., & Ljungberg, T. (1993). **Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task**. The Journal of neuroscience : the official journal of the Society for Neuroscience, 13(3), 900–913. <https://doi.org/10.1523/JNEUROSCI.13-03-00900.1993>.

VON AHN, Luis. **How to Make Learning as Addictive as Social Media**. TED, 2023. 17 min. Disponível em: <https://www.youtube.com/watch?v=P6FORpg0KVo&t=2s&ab\_channel=TED>. Acesso em: 09 de jun. 2024.

1. Disponível em < <https://my.clevelandclinic.org/health/articles/22581-dopamine>>. Acesso em: 10 jun. 2024. [↑](#footnote-ref-1)
2. *Frontend* é onde se encontra a parte visual da aplicação. Ou seja, o aplicativo que irá rodar no celular do usuário. [↑](#footnote-ref-2)
3. *Escreva uma vez, execute em qualquer lugar.* [↑](#footnote-ref-3)
4. *Backend* é onde se encontra a parte lógica da aplicação. Ou seja, os servidores e bancos de dados que são responsáveis pelo processamento de dados. [↑](#footnote-ref-4)
5. Disponível em < <https://spring.io/> >. Acesso em 10 de jun. de 2024. [↑](#footnote-ref-5)
6. Disponível em < <https://fastapi.tiangolo.com/> >. Acesso em 10 de jun. de 2024. [↑](#footnote-ref-6)
7. Não repita a si mesmo [↑](#footnote-ref-7)
8. Mantenha estupidamente simples [↑](#footnote-ref-8)
9. Você não vai precisar [↑](#footnote-ref-9)
10. Componente de Lógica de Negócios [↑](#footnote-ref-10)
11. Disponível em < [https://aws.amazon.com/pt/what-is/api/#:~:text=API%20significa%20Application%20Programming%20Interface,de%20servi%C3%A7o%20entre%20duas%20aplica%C3%A7%C3%B5es](https://aws.amazon.com/pt/what-is/api/#:~:text=API significa Application Programming Interface,de serviço entre duas aplicações). >. Acesso em: 10 de jun. de 2024 [↑](#footnote-ref-11)
12. Interface de Programação de Aplicação [↑](#footnote-ref-12)
13. Núcleo do sistema operacional que gerencia os recursos do hardware e fornece serviços essenciais para a execução de todos os outros componentes de software. [↑](#footnote-ref-13)
14. Disponível em < <https://www.git-scm.com/> >. Acesso em 13 de jun. de 2024 [↑](#footnote-ref-14)
15. Disponível em < <https://www.alura.com.br/artigos/monorepo-usa-lo-desenvolver-integrar-grandes-projetos#:~:text=O%20que%20s%C3%A3o%20monorepos%3F,diferente%20de%20uma%20aplica%C3%A7%C3%A3o%20monol%C3%ADtica>). >. Acesso em 13 de jun. de 2024. [↑](#footnote-ref-15)
16. Tecnologia responsável por alertas em aplicações mobile. [↑](#footnote-ref-16)
17. Um monolito é uma arquitetura de software integrada em uma única aplicação, onde todos os componentes e funcionalidades estão interconectados. [↑](#footnote-ref-17)