UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

FELIPPO STÉDILE

Aplicativo iOS para Motivar Usuários a Alcançarem seus Objetivos Baseado em Competições com Amigos

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

Orientador: Profa. Dra. Renata Galante

Porto Alegre
2024

“ Aquele que puder ser sábio, não lhe perdoamos que não o seja. “

— SÃO JOSEMARIA ESCRIVÁ

AGRADECIMENTOS

Primeiramente gostaria de agradecer àqueles que estão diretamente relacionados com o escopo deste trabalho, os professores que me inseriram com maestria no mundo da programação e que me fizeram aproveitar os últimos anos: Professora Dra. Renata Galante, Professor Dr. Alberto Egon Schaeffer Filho, Professor Dr. Lucas Mello Schnorr, Professor Mauricio Cagliari Tosin, Professor Dr. Alexandre da Silva Carissimi e Professor Dr. João Batista S. de Oliveira. Meus colegas Gabriel Iervolino Fernandes Couto, Filipe Ilunga Xindanhi e mentor Annderson Oreto por também terem me ensinado tanto ao longo dos anos e pelas amizades que pretendo manter para sempre. Quero também agradecer àqueles que me ajudaram indiretamente, começando pela minha família, meu pai, Paulo Carvalho Stédile, que me proporcionou estabilidade durante a vida inteira e nos últimos 5 anos para que eu pudesse me dedicar à faculdade. Minha mãe, Raquel Louise Thomé, pela verdadeira educação que escola e faculdade nenhuma provém, além é claro de todo cuidado e carinho. Minha irmã, Pauline Stédile, que me ajuda muito tanto com sua profissão tanto com sua amizade e amor. Finalmente, quero agradecer a Deus e a todos os Santos que intercedem por mim, em especial Sta. Mônica, São Josemaria, St. Thomás de Aquino, futuro São Carlo Acutis, e acima de todos: São José, Nossa Senhora e Jesus Cristo.

RESUMO

Com o passar do tempo, mais e mais aspectos de nossas vidas se tornam associados com aplicativos móveis. Além disso, há um constante aumento na demanda por soluções à procrastinação. As pessoas estão mais do que nunca procurando por motivação na era em que tantos produtos nos distraem de nossas tarefas diárias.

A correlação entre uso excessivo de Internet e problemas como depressão, procrastinação e ansiedade é surpreendentemente alta. Isso explica o porquê de terem tantas pessoas reclamando sobre como redes sociais impactaram negativamente sua produtividade e porque têm tantas pessoas diferentes atacando esse problema de maneiras variadas. Esse projeto pretende propor uma nova solução para a geração dos *apps* e da procrastinação.

Outro conceito que está em sua maior taxa de aparência em nosso dia a dia é a gamificação. Praticamente tudo tem uma solução digital gamificada hoje em dia já que isso tem se mostrado bastante eficiente a respeito de motivação dos usuários. Com este projeto, outro problema em nossas vidas é abordado com uma nova solução gamificada.

Este projeto descreve o processo de desenvolvimento por trás de uma aplicação *iOS*. Tendo o intuito de providenciar um ambiente para organização de tarefas, será também considerada a disponibilização de recursos presentes em diversos aplicativos já existentes, porém, sem custo mensal ou de entrada.

O projeto visa integrar aspectos online de compartilhamento e competição com amigos ao longo do dia junto às propriedades de organização da rotina para trazer gamificação e maior motivação ao usuário. Tudo isso focando nas tecnologias *Apple* para desenvolvedores e na distribuição pela *App Store* e *TestFlight*.

Com o projeto já publicado no *TestFlight* é possível colher feedback para futuras adições e correções no aplicativo, o tornando um produto escalável e passível de monetização.

Palavras-chave: Aplicativo. Competição. Objetivos. iOS. Tarefas. Disciplina.

iOS Application to Motivate Users to Reach Their Objectives Based on Competition With Friends

ABSTRACT

As time goes by, more and more aspects of our lives become associated with mobile applications. Besides that, there is a constant rize of demand for procrastination solutions. People are more than ever looking for motivation in the era where so many products distract us from our daily tasks.

The correlation between excessive Internet usage and problems such as depression, procrastination and anxiety is surprisingly high. This explains why there are so many people complaining about how social media has negatively impacted their productivity and why there are so many different people tackling this problem in a variety of ways. This project intends to propose a new solution for the generation of apps and procrastination.

Another concept that is in its highest rate of appeareance in our day to day basis is gamification. Practically everything has a gamified digital solution nowadays as it has shown to be very efficient regarding motivating users. With this project, another problem of our lives is addressed with a brand new gamified solution.

This project describes the process of development behind an iOS application. Intended to provide an environment for task organization, will also consider the availability os resources present in a variety of already existing applications, however, free of monthly or entry charge.

The project aims to integrate online aspects of sharing and competing with friends throughout the day along with the properties of routine organization to bring gamification and greater motivation to the user. All focusing on Apples' technologies for developers and App Store distribution.

As the project is already published on *TestFlight* it is possible to collect feddback and use it for future improvements and corrections in the app, turning it into a scalable and monetizable product.

Keywords: iOS, Application, Objectives, Competition, Tasks, Disciplina.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Existem vários aplicativos com o propósito de ajudar pessoas a se organizarem com suas tarefas e motivá-las (MILLEFEUILLE, 2023; TECHNOLOGIES, 2020; APALON, 2015; FABULOUS, 2013; UNORDERLY, 2021; ROUTINERY, 2020; TIIMO, 2019; SOFTWINGS, 2023; DAVETECH CO., 2018; ENERJOY, 2023; PALEBLUEDOT, 2023; SEEKRTECH CO., 2014). Estes disponibilizam uma plataforma onde os usuários anotam suas metas em um tipo de calendário, definem os momentos em que elas serão realizadas e assim conseguem organizar seus hábitos com mais facilidade.

Há também aplicativos que exploram a gamificação (SEEKRTECH CO., 2014; DAVETECH CO., 2018; MILLEFEUILLE, 2023), ou seja, dão ao usuário recompensas e fazem com que o usuário lide com suas tarefas como em um jogo. Estratégia que tem se demonstrado bastante eficiente mesmo fora dos âmbitos digitais e o mercado tem incorporado cada vez mais esse princípio em seus produtos (BALDISSERA, 2021).

Além disso, há também os aplicativos que colocam aspectos sociais, como por exemplo dinâmicas competitivas e cooperativas, para proporcionar um incentivo a maior engajamento e desempenho dos usuários (IRWIN et al., 2012). Exemplo disso é o Gymrats (AVOCADO APPS, 2019).

Centenas de milhares de usuários utilizam diariamente diversos aplicativos que contém algum desses aspectos. A presença das aplicações se encontra mais pressente do que nunca na vida das pessoas movimentando um mercado no mundo inteiro. Usuários confiam desde suas informações mais pessoais até suas decisões diárias a esses produtos.

Nos aplicativos de organização uma característica bastante recorrente é a possibilidade de criar tarefas personalizadas, colocando além do nome delas e dos dias que elas serão feitas (em período mensal e/ou semanal), um ícone ou uma cor que ajude a identificá-las e um horário no qual o usuário espera iniciar a tarefa.

Usualmente o horário é acompanhado por uma notificação que avisa o usuário que deve começar a respectiva atividade, o que não dependeria dele estar sempre dentro do aplicativo para se lembrar de seus objetivos. São milhares os aplicativos que compartilham destas características e prometem aos usuários uma plataforma de organização que facilite a realização de tais objetivos.

1.1 Problema

Por mais que os aplicativos supram de maneira eficiente a demanda por organização e melhor estruturação dos hábitos diários, há ainda a falta de um motivador eficiente para a realização de certos objetivos. Algumas aplicações que serão abordadas ao longo do artigo trazem uma solução para escopos mais específicos, como por exemplo para o público que busca motivação para consistência na academia, e público que busca motivação para aprender uma nova língua.

Mas os aplicativos que eficientemente trazem uma solução motivadora para o usuário realizar seus objetivos focam em pessoas que pertencem a um grupo fechado, deixando com que alguns públicos alvo não tenham sua demanda suprida.

Existem também outros públicos com nichos bem mais específicos para os quais não há aplicativo que aborde sua demanda. E além de tudo isso, alguns aplicativos só disponibilizam certas funcionalidades através de assinatura paga.

1.2 Objetivo

O intuito deste projeto é unir o aspecto de organização, que abrange qualquer pessoa com diferentes tarefas, objetivos e até mesmo compromissos para cumprir, com os aspectos de gamificação e de rede social de uma maneira a conectar pessoas para que motivem umas as outras em um aplicativo que atinja todos os tipos de usuários mais abrangentes e uma gama de tarefas maior.

Deve proporcionar um ambiente simples para uma organização sem distrações de suas tarefas e também contará com algumas características presentes na maioria dos *apps* já existentes para que cumpra demandas básicas dos usuários desse tipo de aplicativo. Mas além disso, possibilitar a conexão com amigos e o compartilhamento do progresso em objetivos do usuário. Assim, possibilitando uma competição que incentiva os usuários a completar mais tarefas do que seus amigos para receber uma recompensa ao fim do mês.

Das possíveis plataformas para as quais o aplicativo pode ser destinado, no primeiro instante e ao longo deste projeto será priorizada a *App Store* (Apple Developer, 2024a) e as tecnologias a disposição da conta de desenvolvedor *Apple* serão utilizadas por proporcionarem maior especialização e aprofundamento de estudos na área de interesse do autor. Contando que a aprovação na *App Store* pode ser um processo burocrático de diversas trocas de mensagem com a loja e que pode levar alguns meses, caso o aplica-

tivo passe demore para passar por esse filtro, será considerada a publicação no *TestFlight* (Apple Developer, 2024h) para fins deste projeto.

1.3 Metodologia

Para suprir o aspecto organizacional o usuário poderá adicionar tarefas a uma lista com seus respectivos dias a serem cumpridos e terá acesso a um calendário que expõe de maneira simples e intuitiva como essas tarefas estão espalhadas nos dias do mês e da semana.

Considerando a união dessa estrutura já estabelecida de aplicativos de organização ao aspecto de gamificação e interação social será integrada a possibilidade de se juntar a amigos em salas com suporte para enviar fotos de tarefas concluídas. Sem que este envio restrinja o usuário a apenas um tipo de objetivo como as soluções no mercado atualmente fazem. Utilizando estas mecânicas, é planejado criar um sistema de pontuação para engajar o usuário a seguir completando seus objetivos para receber recompensas no aplicativo e incentivar a competição com os amigos.

Será utilizado o software *XCode*, ambiente de desenvolvimento feito pela Apple onde a programação em *Swift* e integração com dispositivos *iOS* é facilitada. Será necessário um banco de dados para guardar informações de usuários e possibilitar as dinâmicas online e para isso será utilizado o *CloudKit* que se comunica diretamente com *Swift* sem necessidade de pacotes de terceiros.

1.4 Justificativa

A justificativa deste projeto reside na iniciativa de empreendimento e estudo para suprir uma demanda do mercado com uma estratégia diferente.

A implementação das ideias que aqui serão apresentadas e avaliadas se mostra com potencial de se tornar um produto capaz de atrair usuários e crescer no mercado tornando o projeto uma experiência de empreendimento da qual muitos frutos e habilidades valiosas para o mercado de trabalho serão colhidos.

O projeto é também uma oportunidade de entender melhor como funcionam os ambientes de programação e as ferramentas de desenvolvedor de uma das maiores empresas do mundo, *Apple*. E além disso adquirir contato, conhecimento e experiências na

área de *marketing*, monetização, *design*, *branding* e empreendimento que serão possíveis de serem abordadas a partir da conclusão deste aplicativo.

1.5 Organização do Texto

O restante do texto está organizado da seguinte forma. O Capítulo 2 traz pesquisas e conceitos importantes para o problema o qual o projeto pretende solucionar, apresentando algumas das mais relevantes definições e contextualizações envolvendo tanto o problema quanto a solução e sua eficiência. O Capítulo 3 traz um *benchmark* de aplicativos que abordam os mesmos tópicos e utilizam da mesma estratégia, dissecando cada *feature* e aspecto que mais se repete no mercado, e com isso, discutindo as ferramentas disponíveis para aplicar tais mecânicas no projeto. O Capítulo 4 entra em detalhes sobre o processo de desenvolvimento, as ferramentas utilizadas para alcançar cada *milestone* e algumas informações necessárias para entender o funcionamento do aplicativo. O Capítulo 5, por vez, busca trazer um relatório referente a disponibilização do aplicativo ao público, junto com *feedbacks* de usuários e possíveis aprimoramentos para o futuro. Por fim, o Capítulo 6 traz considerações finais sobre o projeto realizado e os aprendizados ao longo de seu desenvolvimento.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo o enfoque central é a apresentação de pesquisas que apontam a situação atual humana referente a motivação, uso de tecnologia e de aplicativos móveis. Também a busca pela compreensão de técnicas de motivação e de como a experiência do usuários com o aplicativo impacta o incentivo, a persistência e longevidade do uso do produto como uma solução ao problema.

A Seção 2.1 aborda pesquisas científicas feitas a respeito de procrastinação, estatísticas, quais as pessoas mais afetadas e quais são suas principais causas. A Seção 2.2 entra no estudo da gamificação como uma estratégia para manter as pessoas engajados em suas tarefas e prover um nível de recompensa que as motive. E a Seção 2.3 foca em analisar a influência que outras pessoas têm em um indivíduo na sua produtividade para trazer também o aspecto social e as tecnologias *on-line* à solução de motivação.

2.1 Procrastinação

O estudo de fenômenos psicológicos dos seres humanos faz parte das áreas de pesquisa mais complexas, sensíveis e presentes em nossas vidas. Ainda há um universo de informações a serem descobertas e os pesquisadores caminham a largos passos em direção a elas. Muitos artigos foram escritos nos anos recentes sobre esses assuntos e alguns destes serão abordados e discutidos a seguir. O objetivo do estudo que será feito é descobrir como está o cenário atual da sociedade referente a produtividade, procrastinação, motivação e disciplina.

O Doutor Joseph Ferrari conclui de seu artigo (FERRARI, 2020) uma mensagem de apoio emocional para as pessoas que sofrem com a procrastinação, o subtítulo "Todos não estão fazendo"além de confortar pessoas que se sentem mal por estarem procrastinando, também mostra que é um problema muito comum. O estudo também mostra que ao redor do mundo inteiro as taxas de procrastinação crônica são em torno de 20 por cento. E isso considerando uma definição mais acadêmica do termo, ou seja, não incluindo pessoas que simplesmente têm a tendência de deixar as coisas pra depois, e certamente não contabilizou pessoas que fazem isso com tarefas difíceis. Aí está toda uma esfera de pessoas que podem cair em uma definição mais informal do termo procrastinação e que também precisam de maior motivação para realizarem suas tarefas.

Algumas estatísticas sobre procrastinação foram coletadas (LAYNE, 2024). De

acordo com a pesquisa metade dos estudantes do colegial procrastinam suas tarefas. "Procrastinação é um comportamento comum, com alta prevalência entre adultos e estudantes do colegial. Medo de falhar, duvidar de si mesmo, perfeccionismo, evasão de *feedback* são causas comum de procrastinação. Falta de motivação, objetivos pouco claros, se sentir oprimido e baixa confiança contribuem para a procrastinação. Procrastinação tem efeitos negativos no bem estar mental, performance acadêmica, resultados financeiros, e até mesmo saúde física. Tecnologia e outros fatores contribuem para esse comportamento, impactando indivíduos e negócios."São essas as principais conclusões da pesquisa.

Mas não para por ai, a porcentagem de estudantes e adultos que procrastinam chega a 50 e 25 respectivamente, mas isso se refere a procrastinação crônica, quando falamos de procrastinação no geral a porcentagem dos alunos sobe para 95! As tarefas que são procrastinada variam, desde comportamentos vinculado a saúde até tarefas de estudo, mas 74 por cento dos graduandos afirmam que distrações na Internet fomentam sua procrastinação. Também é mencionado que a falta de um indicador de recompensa a curto prazo é outro causador da procrastinação, e é nesse ponto que será abordada a solução gamificada futuramente.

Outras estatísticas interessantes podem ser encontradas, além de reafirmações dos números da pesquisa anterior. Não são apenas os estudantes que procrastinam, 88 por cento dos trabalhadores procrastinam mais de uma hora diariamente no trabalho, procrastinação custa 70 bilhões de dólares por ano para os Estados Unidos da América, apenas 30 por cento dos adultos não procrastinam ou o fazem raramente, adultos passam em média 3 horas e 38 minutos procrastinando diariamente, os motivos da procrastinação dos adultos são compostos por falta de motivação representando 41 por cento, falta de urgência com 24 por cento, falta de tempo representa 25 por cento, e falta de entendimento, 10 por cento (FLYNN, 2023).

Homens têm taxas maiores de procrastinação e as idades entre 14 e 29 anos são as que mais procrastinam. O motivo da procrastinação de 58 por cento dos estudantes são as redes sociais. E uma das estatísticas mais importantes que não foi mencionada ainda, é que 94 por cento das pessoas desgostam de sua procrastinação.

Em outro estudo (SHATZ, 2021) porcentagens muito parecidas se repetiram e além delas algumas outras estatísticas foram mencionadas, como alguns possíveis prejuízos da procrastinação: maior taxa de divorcio, redução salarial e outros. De acordo com a pesquisa as taxas de procrastinação parecem estar subindo, tendo que as pessoas passam muito mais tempo na Internet hoje em dia do que no passado.

Contudo, a procrastinação não é uma barreira sem contorno, existem algumas estratégia com as quais se pode lidar com tal problema, duas das estratégias para superar procrastinação serão abordadas ao longo do projeto. São elas: fazer uma lista de afazeres com o tempo de expiração das tarefas e ser recompensado ao final de uma tarefa (CHERRY, 2022).

Um fator que contribui para esse problema de procrastinação afetar tantas pessoas ao redor do mundo é o uso excessivo e o vício em telefones celulares. Estudos mostraram uma correlação significantemente positiva entre vício nos aparelhos e procrastinação (CHEN; LYU, 2024; ZHOU et al., 2024).

2.2 Gamificação

Jogos se tornam uma parte significante de diferentes culturas e promovem além de engajamento, motivação. Por esses motivos, algumas das mecânicas dos jogos passaram a ser utilizadas em outros contextos como por exemplo para a educação básica em ensino fundamental e elementar (IOANNOU, 2019; ZAINUDDIN, 2018; RACHELS, 2018) também no ensino para adultos (HUANG; HEW; LO, 2019; HUANG; HEW, 2018; BARATA et al., 2017) no trabalho (PASSALACQUA et al., 2019; PERRYER et al., 2016) para o engajamento e influência de consumidores (TOBON; RUIZ-ALBA; GARCíA-MADARIAGA, 2020; MORGANTI et al., 2017) também na área da saúde (ORJI; MOFFATT, 2018; MORGANTI et al., 2017) para promover mudança de comportamento (ZAINUDDIN et al., 2020) e mais importante para o contexto, para a promoção de motivação (HAMARI; KOIVISTO; SARSA, 2014; ALBERTAZZI; FERREIRA; FORCELLINI, 2019; KOIVISTO; HAMARI, 2019; DING; KIM; OREY, 2017).

Mas a fórmula mágica não é assim tão simples de ser atingida, para aproveitar da verdadeira eficiência da gamificação é necessário fazer uma intervenção eficiente com o modelo da gamificação, assim a estratégia atinge seu verdadeiro impacto (KRATH; SCHüRMANN; von Korflesch, 2021).

Uma forma eficiente de aplicar conceitos da gamificação é com competição entre usuários (BAYUK; ALTOBELLO, 2019). Isso é possível tanto para competições entre diferentes times quanto para competições internas com foco no mesmo objetivo e em uma mútua motivação para alcançá-lo (MORSCHHEUSER; HAMARI; MAEDCHE, 2019). Tudo isso faz com que a presença de elementos de competitividade dos jogos em outros contextos de produção se tornem ótimos medidores de eficiência (DISSANAYAKE et al.,

2019).

2.3 Fator Social na Eficiência

Focando no conceito de motivação, que é esperado que este aplicativo a ser desenvolvido gere nas pessoas, é importante entender melhor sua definição. Weiner (WEINER, 1992) definiu o conceito como uma construção psicológica que faz referência às forças que iniciam, energizam, direcionam, e sustentam um comportamento, geralmente em direção de um objetivo ou recompensa específicos. Zeidner (ZEIDNER; BOEKAERTS; PINTRICH, 2000) trouxe uma definição de motivação como um motor o desejo interno de um indivíduo para engajar em uma atividade ou comportamento particular. É o processo psicológico que inicia, guia e mantém ações que visam um objetivo. Schunk (SCHUNK, 2004) descrevia o conceito como um processo através do qual ambições eram ativadas e mantidas, geralmente envolvendo direcionamento, excitação, e persistência. Guerin et al. (GUERIN et al., 2011) definiu motivação como um processo psicológico que dá propósito e direção a um comportamento, é o que causa a ação em uma pessoa, seja tomar aguá para reduzir a sede ou ler um livro para receber conhecimento.

Tendo em vista o quê é a motivação, a primeira pergunta a se fazer é: O que causa esse processo psicológico? E são várias as possíveis fontes de motivação.

Alguns estudos nos levam entender que parte da eficácia por trás da gamificação gerar ações partindo de intenções se dá pelos aspectos de comparação social e pela busca humana de atingir, ou até mesmo superar, as expectativas de seus próximos (AJZEN, 1985; RYAN; DECI, 2017).

Porém é necessário manter as pessoas envolvidas com um relacionamento positivo e amigável. Isso é relevante pois uma competição a nível de exaustão do relacionamento pode não proporcionar o aumento desejável de produtividade que outras estratégias que se propõem a manter os indivíduos motivados por constantes afirmações positivas e de um contexto mais cooperativo trazem (SAINSBURY; WALKER, 2012). Estas informações são úteis não apenas para aperfeiçoar a maneira de incentivo para os usuários mas também para a longevidade da estratégia, pois o estudo também mostra que por mais que a companhia seja de amigos e os objetivos sejam os mesmos, é possível que um relacionamento com menosprezo no âmbito de comparação social reduza a persistência dos envolvidos ao invés de ampliá-la.

Outros estudos concordam com essas conclusões que apontam os possíveis pro-

blemas de competição quando mal utilizada (REEVE, 2023), mas seguem apontando a competição saudável como uma forma bastante eficaz de motivação.

Um estudo que envolve mais do que apenas motivação, mostra o efeito da competição em atenção, memória, performance, tempo de reação e algumas outras habilidades (DIMENICHI; TRICOMI, 2015). Este estudo é interessante por mostra que homens e mulheres são afetados em diferentes níveis de magnitude. Os prejuízos trazidos pela competição nos contextos estudados nesta pesquisa em questão são atribuídos a nervosismo e ao aumento do nível de ansiedade proporcionado pela tensão da competição.

O envolvimento do ego é um fator que explica como a motivação pode ser atingida por meio da competição, tanto pelos participantes não desejarem se ver falhando em algo que gostariam de conquistar, quanto pela ideia do juízo alheio negativo. Com isso podem vir benefícios em performance mas algumas possibilidades que prejudicam os competidores dependendo do contexto. Isso ocorre quando o objeto da competição possibilita a trapaça, *doping*, objetificação, agressão e, no geral, falta de desportivismo e *fairplay*.

3 REVISÃO BIBLIOGRÁFICA

Nesse capítulo serão abordados, na seção 3.1, o *benchmark* feito considerando outros aplicativos e soluções para o problema que será abordado, suas funcionalidades, a maneira com que elas são executadas e *feedback* de seus usuários. E na seção 3.2 serão exploradas as tecnologias que vão ser utilizadas e que possivelmente serão úteis para a confecção deste projeto.

3.1 Funcionalidades

Foi feito um *benchmarking* com 14 aplicativos para investigar quais as principais características que compõem um aplicativo de tarefas bom. Deu-se prioridade aos aplicativos presentes na *App Store* considerando que estão de acordo com as tecnologias disponibilizadas pela *Apple* e estão dentro do escopo do projeto que está limitado ao sistema *iOS*, compatível com *IPhone, IPad, Mac e Apple Watch*.

3.1.1 Tempo

Há dois tipos de estratégias tomadas a respeito do momento que o usuário pretende iniciar uma tarefa pelos maiores aplicativos. Os aplicativos Fabulous, Routinery, Tiimo e HabitTracker colocam um timer na tarefa, fazendo com que o aplicativo mantenha o foco do usuário naquela função específica (FABULOUS, 2013; ROUTINERY, 2020; TIIMO, 2019; DAVETECH CO., 2018).

Há aplicativos que enviam notificações enquanto outras funcionalidades do celular são utilizadas durante o tempo de uma tarefa (ROUTINERY, 2020). Isso causa feedbacks divididos nos usuários. Alguns deles consideram isso um ponto positivo e mencionam o quanto isso ajuda na questão do Transtorno de Deficit de Atenção e Hiperatividade (TDAH) [1]. Já outros gostariam de poder ter mais liberdade utilizando o aplicativo e com pausas livremente tomadas durante as tarefas.

Existem os aplicativos que utilizam o método pomodoro de realização de estudos (ROUTINERY, 2020). O que consiste em vários temporizadores de 25 minutos, normalmente, que seriam utilizados para a realização da tarefa, intercalados com descansos de 5

[1] "O TDAH se caracteriza por uma combinação de dois tipos de sintomas: desatenção e hiperatividade-impulsividade"segundo a Associação Brasileira do Deficit de Atenção

minutos.

3.1.2 Coleta de Dados Extras

Vários aplicativos, mesmo que sendo a minoria, já têm de início algumas tarefas pré estabelecidas (FABULOUS, 2013; TIIMO, 2019), dentre elas se encontram comumente: tomar água, acordar cedo, exercício físico, meditação e estudo. E para o monitoramento de alguns desses hábitos, são utilizadas as tecnologias healthKit (Apple Developer, 2024d) e watchKit (Apple Developer, 2024j) disponibilizadas pela Apple para coletar dados como calorias gastas e distância percorrida. Porém carregam um pré requisito para entregar essas funcionalidades que é visto negativamente pelos usuários: a necessidade da conexão com a Internet.

Considerando que o usuário poderá criar essas tarefas e que dependendo do uso que ele queira ter do aplicativo pode não fazer sentido a presença delas, foi optado por não utilizar dessas integrações.

3.1.3 Aparência

Um ponto que frequentemente é levantado nas avaliações positivas de usuários é a estética dos aplicativos, que são elogiadas quando visam um ar mais calmo e minimalista, às vezes acompanhado de música e sons ambiente (MILLEFEUILLE, 2023; FABULOUS, 2013; TIIMO, 2019). Isto pode ser devido ao vínculo com meditação que esses aplicativos têm, onde o usuário que busca esse tipo de rotina se identifica com um estilo voltado a introspecção e leveza. Para aplicativos sem essa estilização é esperado que tenham suporte para *Dark Mode*.

3.1.4 Dispositivos

A maioria dos aplicativos têm suporte para Ipad, mas são poucas as avaliações onde os usuários mencionam a utilização do aplicativo na plataforma em questão. Uma porcentagem menor tem suporte para Apple Watch, que normalmente é utilizado para medições de exercícios, notificar o usuário do começo da tarefa e mostrar o temporizador quando contido no aplicativo.

3.1.5 Recompensa Visual

Um conceito que é foco principal em alguns dos aplicativos, mesmo sendo uma minoria pois neste caso é o que define a identidade deles, é a manutenção de um ambiente virtual baseada no desempenho do usuário. Eden - Rotina Diária e Hábitos (MILLEFEUILLE, 2023) e Forest - Mantenha o Foco (SEEKRTECH CO., 2014) são dois dos maiores exemplos dessa estratégia, ambos utilizando de um jardim, onde a vida e quantidade das plantas dependem do cumprimento das tarefas por parte do usuário.

3.1.6 Histórico de Tarefas

Diferentes métodos são utilizados para mostrar ao usuário o histórico de seu desempenho, sendo o mais bem avaliado entre eles um sistema parecido com o do Github, com cores marcadas nos dias em que as tarefas foram cumpridas. Nem todos têm a funcionalidade do histórico, mas é um diferencial grande nos que a possuem (PALIASHCHUK, ; TECHNOLOGIES, 2020; DAVETECH CO., 2018; ENERJOY, 2023; PALEBLUEDOT, 2023) pois podem fazer um resumo de como o usuário se saiu na semana, no mês e no ano.

3.1.7 Funcionalidades Extras

Existem outras funcionalidades que são mais dificilmente encontradas nos aplicativos como por exemplo a de motivar o usuário a largar hábitos ruins (DAVETECH CO., 2018).

Há também os aplicativos que medem os hábitos de maneira não binária, ou seja, uma porcentagem e outras métricas para acompanhar o progresso de uma única tarefa (TECHNOLOGIES, 2020; UNORDERLY, 2021; DAVETECH CO., 2018; PALEBLUEDOT, 2023).

É possível fazer o acompanhamento das emoções do usuário alimentada pelo próprio usuário e depois dispostas em um calendário com um histórico (DAVETECH CO., 2018; ENERJOY, 2023).

Widgets agregam bastante valor como lembretes e maneira mais rápida de acessar as tarefas (DAVETECH CO., 2018).

Sincronização de dados possibilitando que sejam acessados de outros dispositivos (UNORDERLY, 2021; DAVETECH CO., 2018), principalmente para quando o usuário troca de dispositivo.

Frases motivacionais diárias (APALON, 2015; ROUTINERY, 2020; TIIMO, 2019; DAVETECH CO., 2018) pode ser interessante para alguns usuários.

O aplicativo Lembretes de Caderno Digital (SOFTWINGS, 2023) disponibiliza uma inteligência artificial que gera imagens simples para que o usuário as utilize como ícone da tarefa, função que geraria um custo para a manutenção do aplicativo. Há um aplicativo, Habit Tracker (DAVETECH CO., 2018) com funcionalidades que começam a se assemelhar mais com aquelas propostas nesse projeto.

3.1.8 Interação Online

Habit tracker conta com um ambiente de comunidade, onde o usuário pode acompanhar seus amigos e ver quais tarefas eles estão cumprindo tanto de maneira passiva como em um modo competitivo, que é onde entra a idéia deste projeto. Um aplicativo que tem foco nisso e se sai muito bem, porém tem um foco mais específico em um tipo de tarefa é o Gymrats (AVOCADO APPS, 2019). Nele as pessoas se juntam com seus amigos e disputam entre si para ver ao final de períodos quem teve mais tempo de treino e atividade física realizada. No sistema do Gymrats os usuários enviam fotos de suas atividades onde os outros podem reagir e deixar comentários.

3.2 Tecnologias

Para tirar esse projeto do papel será necessário conhecimento das tecnologias da *Apple*. São duas as principais opções de linguagens nas quais um projeto como esse pode ser viabilizado: *UIkit* e *SwiftUI* (Apple Developer, 2024i; Apple Developer, 2024g). Como *UIKit* é uma linguagem mais antiga, de um nível mais baixo, e de mais difícil acesso para quem inicia no ambiente *Apple*, a utilização de *SwiftUI* será priorizada, ou seja, *UIKit* só será utilizado quando e se for necessário a utilização de um recurso indisponível no seu sucessor *SwiftUI*, que também torna mais fácil o alinhamento com *guidelines* da interface (Apple Developer, 2024e).

A funcionalidade mais básica e essencial, que seria ter as tarefas salvas, poderá

ser implementada com Core Data e *CloudKit*, um deles proporcionando que o usuário salve dados localmente (Apple Developer, 2024c), e o outro possibilitando que o usuário salve na nuvem (Apple Developer, 2024b). Com o login do *ICloud* poderia ser feita a sincronização entre outros dispositivos assim como facilitará as primeiras etapas de cadastro no aplicativo para o usuário (Apple Developer, 2024f).

Capturar informações do usuário vinculadas à saúde como passos, calorias gastas e distância percorrida já é feito pelo sistema caso o usuário autorize, e esses dados, caso sejam usados no projeto, podem ser acessados pelo *HealthKit* (Apple Developer, 2024d). Esse *framework* pode ser utilizado junto ao *Apple Watch* para mais informações e melhor controle do progresso do usuário (Apple Developer, 2024j), porém como as tarefa são bastante abrangentes, de livre criação do usuário, e os dados do *HealthKit* são específicos, é provável que essa funcionalidade fique de fora neste projeto por mais que aplicativos abordados no *benchmark* como o Tiimo e Fabulous façam proveito dessas ferramentas.

Um sistema de salas com códigos privados é o que é comumente mais usado nesses meios, e provavelmente será a solução para colocar todos amigos em um mesmo ambiente onde podem visualizar uns aos outros. As salas seriam salvas no banco de dado do *CloudKit* assim como os usuários.

A plataforma do *TestFlight* disponibilizada pela *Apple* como uma etapa da publicação de um aplicativo será utilizada para testes de usuários. Com isso será possível ter *feedbacks* das partes visuais e interativas do aplicativo, como também das funcionalidades das mecânicas online e que utilizem do banco de dados.

4 DESENVOLVIMENTO DO PROJETO

Neste capítulo são explicadas as lógicas e o processo de desenvolvimento por trás do aplicativo. Assim como os problemas que surgiram ao longo do projeto e quais as estratégias usadas para solucioná-los. Será dissecado o processo completo, desde as partes mais próximas do usuário como o design, interações e a *user experience*, até as tecnologias as quais o usuário final está alheio durante a utilização do aplicativo, e também a motivação por trás de algumas decisões de implementação.

A Seção 4.1 é aprofundada a lógica de comunicação entre telas abordando o tópico da arquitetura escolhida e uma introdução às linguagens *Swift* e *SwiftUI* descrevendo estruturas de dados utilizadas para possibilitar o fluxo de informação e geração de estruturas pelo desenvolvedor. Na Seção 4.2 apresenta uma visão geral de como são implementadas e o resultado das partes que compõem a interface do usuário do aplicativo, assim como uma introdução ao ambiente de desenvolvimento do *XCode*. A Seção 4.3 explica a lógica por trás de algumas funções que facilitam e possibilitam muitas das *features* disponibilizadas pelo aplicativo. Finalmente, a Seção 4.4 revela as estruturas e predicados na nuvem que compõem todo o funcionamento e dinâmica das salas onde o usuário pode se juntar a amigos e compartilhar suas tarefas concluídas, e assim competirem e motivarem uns aos outros.

4.1 Arquitetura e Estrutura

Nesta seção será explicada a arquitetura escolhida, as propriedade presentes em *SwiftUI* que são chave para a boa implementação da mesma, e a estrutura básica em alto nível de como são implementadas as *Views* em *SwiftUI*.

4.1.1 Model-View-ViewModel

A arquitetura *Model View ViewModel* é um padrão que propõe a separação das partes gráficas (*views*), lógicas (*ViewModel*) e de *back-end* (*Model*).

Consiste em ter na *Model* as estruturas que representam dados utilizados pelo usuário. A *View* é responsável apenas por aquilo que é mostrado na tela graficamente como por exemplo textos, cores, botões, elementos de listas. E a *ViewModel* age como

uma controladora da *View*, ou seja, as partes lógicas como navegação, ordenação de listas, condicionais, cálculos matemáticos e armazenamento de dados utilizados pela *View* se encontram na *ViewModel*

Essa arquitetura se comunica muito bem com a linguagem utilizada, *SwiftUI*, e isso se dá principalmente por culpa dos propertyWrappers [1].

Dos vários *property wrappers* disponíveis, os mais úteis e imprescindíveis para essa comunicação são:

- **@Binding** Guarda uma referência para uma variável, necessariamente. Alterar o *binding* localmente também muda o dado remoto.
- **@Environment** Permite a leitura de dados do sistema, como esquema de cores, opções de acessibilidade, e mais importante, podem ser adicionados objetos customizáveis aqui.
- **@EnvironmentObject** Lê um objeto que foi colocado no *environment*.
- **@ObservedObject** Refere uma instância de uma classe externa que se conforma com o protocolo *ObservableObject* [2].
- **@Published** É utilizado em propriedades dentro de um *ObeservableObject*, comunica o *SwiftUI* que deve recarregar telas que utilizam essa propriedade quando elas forem alteradas.
- **@State** Permite a manipulação local de valores individuais fazendo com que estes se repliquem na *View*
- **@StateObject** Parecido com *ObservedObject* mas permite a criação de novas instancias de classes *ObservableObject* localmente.

(HUDSON, 2024)

O *property wrapper @EnvironmentObject* é o único que quebra as regras da arquitetura *MVVM* e será utilizado para a persistência dos dados da classe *UserManager*. Assim serão necessários menos envios de dados entre telas e todas elas terão acessos ao usuário local e algumas funções declaradas nesse objeto.

[1] Para uma explicação mais detalhada do funcionamento por trás dos *property wrappers*, acessar *Apple Documentation* (APPLE, 2024a)

[2] O protocolo *ObservableObject* faz com que as referências para as classes que conformam com ele sejam notificadas quando houverem alterações nos dados da classe.

4.1.2 Views

A estrutura primária que transforma *Swift* em *SwiftUI* é a *View*. Uma *View* é uma estrutura que representa parte da interface do aplicativo. As *Views* são amplamente configuráveis e geralmente são uniões de outras *Views* (APPLE, 2024e).

Por exemplo, é possível criar uma *View* com um texto vazio na estrutura *Text* (que também é uma *View*) e alterá-lo através do apertar de um botão da estrutura *Button* (a qual é outra *View*). Se a variável que guarda a *String* for um @*State*, por exemplo, será possível ver a alteração acontecendo na interface.

Para cada estrutura *View* podem ser adicionados atributos e modificadores dependendo dos protocolos com as quais ela se conforma. Então, para um estrutura de texto, podem ser adicionados modificadores de fonte, tamanho, limite de linhas e outros. Para estruturas de imagens podem ser adicionados modificadores que invertem sua cor, escalem a foto, adicionem efeitos como *blur* e *shadow*, assim por diante. Esse padrão de declaração de estruturas e atributos está exemplificado no Aloritmo 1.

Algoritmo 1 View

```
var body: some View {
    VStack() {
        Text("")
            .fontWeight(.bold)
        Button(action: {
            functionChangeText()
        }, label: {
            ViewToPress()
        }
    }
}
```

Todos estes componentes utilizam de um sistema de pontos, que correspondem a uma certa quantidade de *pixels* dependendo do dispositivo. Assim, um design pode ser portado sem burocracia alguma para dispositivos de tamanhos e de resoluções das mais variadas com diferenças geralmente imperceptíveis.

4.1.3 Pilhas

No exemplo anterior, o botão e o texto ficam dispostos na tela da maneira que o programador ou designer preferir, basta colocá-los dentro de uma pilha que corresponder à demanda. São três as pilhas disponíveis para esse fim: vertical (*VStack*) indo de cima para baixo, horizontal (*HStack*) iniciando pela esquerda, e referente a profundidade (*ZStack*) onde o primeiro elemento é o mais "distante"do usuário em profundidade.

Além da disposição dos componentes *Views* serem em pilha, as telas para as quais se pode navegar também utilizam da lógica de *stack*. Quando se pretende navegar para uma outra tela, esta geralmente sofre, "por baixo dos panos", um *push* na pilha. Para voltar, a lógica é a mesma de um *pop*.

4.1.4 Classes e Structs

Tendo em mente o comportamento das telas e como deve ser estruturada a arquitetura do projeto, são criadas estruturas do tipo *View* para cada visualização, componente ou elemento visual do aplicativo, alguns destes serão apresentados na seção seguinte. Essas estruturas possuem uma classe declarada como *StateObject* para ser sua *ViewModel* e conter todas as funções que mexem com os dados da *View* podendo atualizá-la automaticamente e mostrar as mudanças para o usuário sem necessidade de recarregar a página.

Além dessas há a necessidade também de uma classe para se conectar com o banco de dados e outra classe que será o administrador do usuários. A segunda é a *UserManager*, responsável por manusear os dados localmente e é acessível por ser um *Environment Object* por todas as outras classes, será abordada com mais profundidade na seção 4.3 Lógicas e Cálculos. E a outra, *CloudkitService*, que é um *Singleton* será abordada na seção 4.4 Nuvem.

4.2 Interface do Usuário

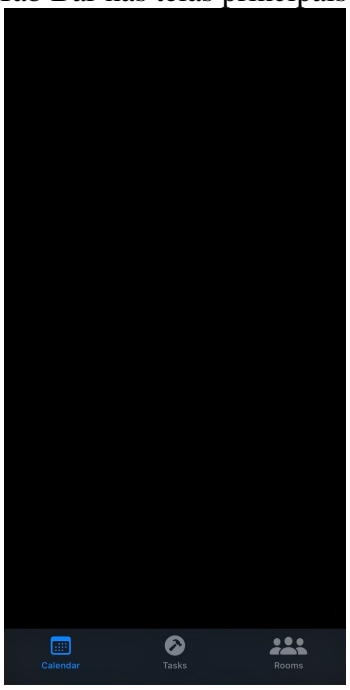
Nesta seção serão explorados os elementos e *Views* utilizadas na composição do projeto, mostrando alguns resultados visualmente.

4.2.1 Telas

São três as principais telas do aplicativo. Elas utilizam de uma outra maneira de navegação que é através de uma *tab bar*, a barra que se encontra na parte inferior das figuras 4.1 onde existem três botões, cada um levando para uma das seções do aplicativo.

Tab bar é um componente visual e de navegação para separar conteúdos diferente (APPLE, 2024d). É formada por uma quantidade definida pelo desenvolvedor de elementos compostos por descrição e um símbolo que servem para indicar ao usuário o contexto ao qual será levado no apertar do elemento em questão. O espaçamento entre cada elemento é automático e constante para cada distância entre elementos e limites da tela.

Figura 4.1: Tab Bar nas telas principais do aplicativo



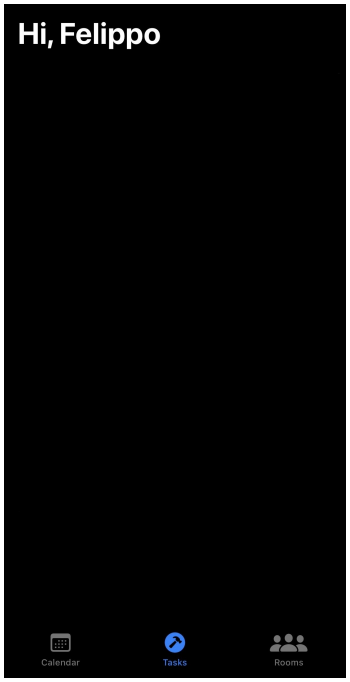
Fonte: Autor

Para cada elemento da *tab bar* foram escolhidos símbolos que representassem, respectivamente, um calendário, trabalho, e uma sala de amigos ou comunidade, presentes na biblioteca do *SF symbols*. Uma biblioteca disponibilizada pela *Apple* para que desenvolvedores tenham símbolos esteticamente compatíveis com a fonte customizada criada por eles, *San Franciso* (APPLE, 2024c).

Além da *tab bar*, as telas principais também possuem um letreiro com um texto que se comunica com o usuário ou passe uma informação sobre o contexto em que ele se encontra. Assim simulando, mas não implementando em nível de código, o visual

resultado de um *Navigation Title* em um *Navigation Bar*, exemplificado na Figura 4.2.

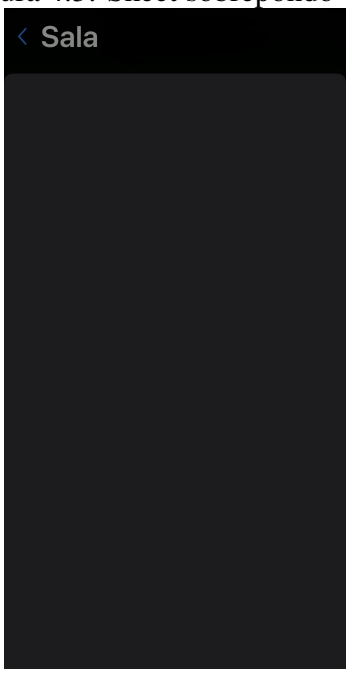
Figura 4.2: Título das Views no topo



Fonte: Autor

Há também outras maneiras de mostrar conteúdo para as telas que não são a base do aplicativo. A segunda técnica mais presente no projeto é a *sheet*. Utilizada nos contextos de criação e de consulta rápida de informações, é como uma planilha animada que parte do limite inferior da tela e se sobrepõem à *view*, mantendo ela no fundo e à vista do usuário. Exemplificada na Figura 4.3.

Figura 4.3: Sheet sobrepondo View



Fonte: Autor

4.2.2 XCode Preview

O ambiente de desenvolvimento *XCode* providencia uma ferramenta muito útil para a confecção rápida destes elementos visuais. Tanto a *tab bar*, quanto os títulos, botões, listas, *sheets* e até mesmo navegações, ou os componentes individuais da interface podem ser testados através da *preview* (APPLE, 2024b).

Através dela é possível escolher o dispositivo iOS para o qual o desenvolvedor deseja testar a interface e durante a codificação dos elementos visuais, são compilados em uma tela que simula o dispositivo escolhido como na Figura 4.4. Assim é possível ter *feedback* visual de como estão resultando as telas sem necessitar de pausas para compilar todo o projeto e carregar o simulador.

Figura 4.4: Preview da tela de tarefas

Fonte: Autor

A aplicação desta pre-visualização é opcional, para habilitar essa ferramenta basta escrever o código demonstrado no Algoritmo 2 enviando a *View* da tela sendo codificada.

Algoritmo 2 Preview em SwiftUI

```
#Preview {
    View()
}
```

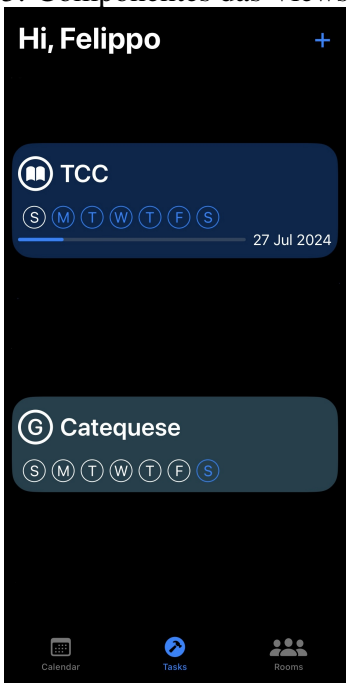
4.2.3 Task

O elemento essencial do projeto é o de tarefa. É através deste que toda a organização do usuário é possibilitada. Composto por um retângulo como primeiro elemento de uma *ZStack*, este é colorido baseado em um *enum* de tipos de tarefas, cada tarefa é atribuída a uma cor e um símbolo que muda o visual do elemento *TaskView*.

O símbolo, também retirado do *SF Symbols* é um dos dois componentes que o objeto *Task* necessita ter para ser criado. Seu símbolo e nome são dispostos lado a lado em uma *HStack* e essa pilha é posicionada acima dos componentes opcionais: dias da semana ou do mês nos quais a tarefa precisa ser cumprida, e o prazo final da tarefa junto com uma barra de progresso nativa do *SwiftUI*.

É possível visualizar exemplos desse componente na Figura 4.5. São colocados nessa disposição não apenas através da pilha vertical, mas também utilizando um iterador *ForEach*, que recebe uma estrutura iterável, como um vetor e para cada instância executa uma *View*. O campo do iterador foi preenchido com um vetor de tarefas, e no campo da *View* foi colocada a *TaskView* recebendo a tarefa a ser apresentada.

Figura 4.5: Componentes das Views de tarefas



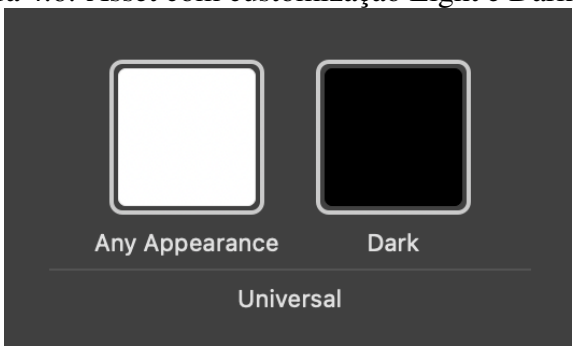
Fonte: Autor

4.2.4 Dark Mode

Os dispositivos móveis geralmente têm um suporte para uma configuração de tela que deixa todas as interfaces do sistema em um paleta de cores mais escura para usuários que preferirem assim. Para possibilitar que isso seja feito em um aplicativo em *SwiftUI* é preciso definir para cada *asset* utilizado seu correspondente do *dark mode*.

Já que cores são considerados *assets* é possível criar uma cor com seu hexadecimal ou rgb para o *light mode* e um *HEX* ou rgb para o *dark mode* como feito no exemplo da Figura 4.6.

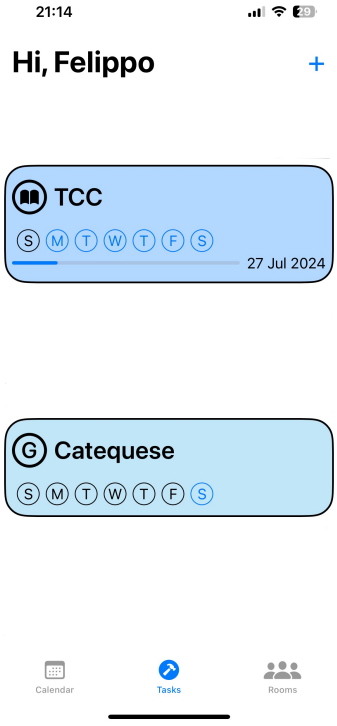
Figura 4.6: Asset com customização Light e Dark Mode



Fonte: Autor

Até agora todas os *screenshots* do aplicativo apresentados neste documento foram tirados no *dark mode*, enquanto a *preview* estava no modo padrão. Na Figura 4.7 estão os elementos das tarefas em *light mode*.

Figura 4.7: Componente das Views de tarefas no Light Mode



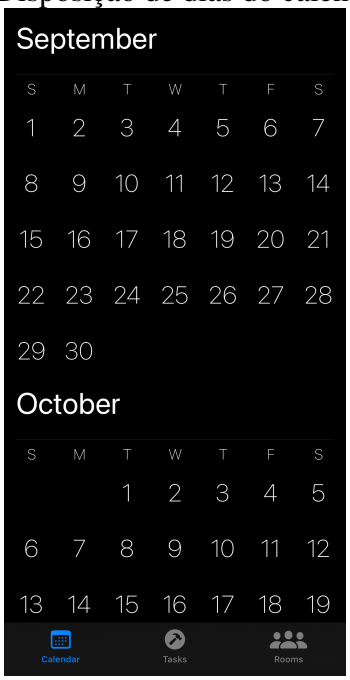
Fonte: Autor

4.2.5 Calendário

A *TabView* que apresenta o calendário utilizou de cálculos mais complexos para ser confeccionada, alguns serão abordados na próxima seção deste capítulo. Quanto a parte visual é interessante mencionar duas estruturas.

A primeira delas é nativa e utilizada para a disposição de elementos iteráveis assim como o *ForEach*, mas com a possibilidade de organização similar a uma matriz, o *LazyGrid*. Utilizando desta, é facilitada imensamente a criação de uma interface como a da Figura 4.8, que gera uma interface calendário como conhecemos usualmente.

Figura 4.8: Disposição de dias do calendário em grid



Fonte: Autor

A segunda foi criada para possibilitar que o usuário tivesse uma noção de quantas tarefas ele tem para completar no dia através do calendário. Uma *View* que recebe as tarefas do dia e gera um círculo de arcos coloridos cujas cores se baseiam na porcentagem de tarefas de um tipo que estão para serem concluídas no período em questão. Fazendo com que o calendário se renderize como na Figura 4.9

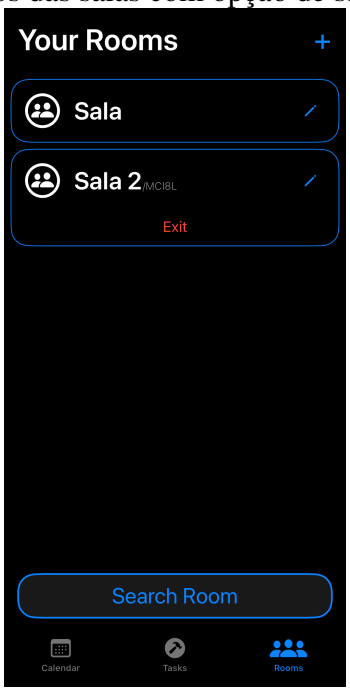
Figura 4.9: Círculos representando as tarefas em um dia

Fonte: Autor

4.2.6 Sala

São duas as telas vinculadas às salas, inicialmente a lista de seus componentes através dos quais se pode navegar para uma sala existente, criar uma nova, procurar uma sala nova com um código, visualizar código de entrada e sair da sala. Tudo isso demonstrado na Figura 4.10

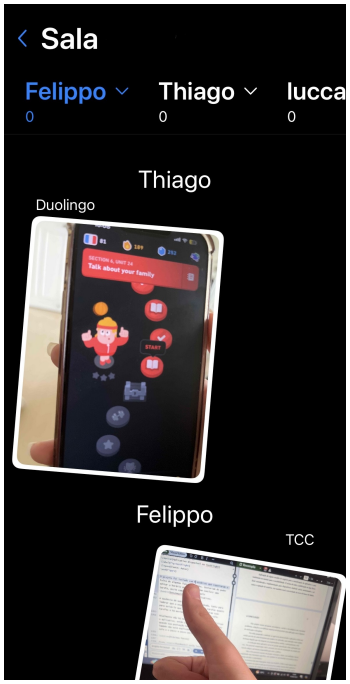
Figura 4.10: Elementos das salas com opção de saída e código de acesso



Fonte: Autor

Já no interior da sala foi criada uma seção horizontal com o nome dos participantes e seus respectivos pontos na competição. Abaixo, há uma *Scrollview*, *View* sem limitação de tamanho e que pode ser arrastada, com fotos das tarefas que cada usuário conclui no dia formando um *feed*. Esta tela é visualizável na Figura 4.11.

Figura 4.11: Fotos publicadas na sala com nomes das respectivas tarefas e usuários



Fonte: Autor

Clicando no nome de um usuário pode ser visualizada através de uma *sheet*, as tarefas que este importou na sala para fazerem parte da competição. E a mesma *View* é utilizada para usuário local, porém, com alterações para conseguir importar ou retirar tarefas da sala como amostra na Figura 4.12.

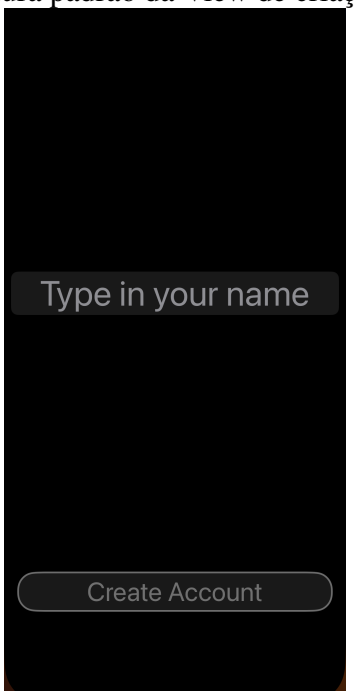
Figura 4.12: Visual de tarefas importadas e não importadas em Sheet

Fonte: Autor

4.2.7 Criação de Usuário, Tarefa e Sala

Também importante, mas mais simples, há a *View*, de criação de usuário que segue o modelo da criação de Sala exemplificada na Figura 4.13.

Figura 4.13: Estrutura padrão da View de criação de usuário e sala



Fonte: Autor

4.3 Lógicas e Cálculos

Nesta seção, serão abordados algoritmos que estão por traz do funcionamento das interfaces, coordenando listagens, datas, cálculos, pontos e criações de entidades mais complexas.

4.3.1 Extension Date

Com *SwiftUI* temos acesso a uma estrutura que representa a data, *Date*. Através dela é possível, por exemplo, saber o dia atual, definir uma verdadeira data para tarefas, até mesmo determinar horários. Porém, suas partes não são compostas de inteiros, ponto flutuante ou *doubles*, e isso faz com que os cálculos envolvendo datas necessitem de uma etapa de conversão a mais.

Por isso, e para facilitar a aquisição de outras informações, foi feito uma extensão no tipo data, que adiciona variáveis e funções à estrutura. Foi preciso criar funções na extensão para possibilitar a confecção do calendário como foi mostrado. É necessário saber, em inteiros: o dia da semana, quantidade de dias até domingo e quantidade de dias no mês, e os próximos 12 meses.

No Algoritmo 3 é possível verificar as funções que retornam um inteiro representando o dia da semana, um inteiro que representa a data inteira com finalidade de fácil comparação com outras datas, funções que retornam os meses até o fim do ano e os dias a serem dispostos em um mês, e por fim uma função que retorna um *booleano* a data seja a atual.

4.3.2 Distância das Tarefas

É possível deixar as tarefas do usuário em ordem na *TaskListView* fazendo com que o *Model* da tarefa se conforme com o protocolo *Sortable*. Para isso é necessário criar uma função que recebe dois modelos de tarefa, e através de uma regra determine qual deles deve ser priorizado no vetor.

O que torna a função mais complexa é o fato da tarefa ter várias datas em formatos diferentes, um sendo inteiros de 1 a 7 representando dias da semana, outro sendo inteiros de 1 a 31 representando os dias do mês, e apenas o prazo final da tarefa segue a estrutura *Date*.

Contudo, é necessário checar a quantidade de dias entre a data atual e o primeiro desses vários possíveis índices como feito no Algoritmo 4.

4.3.3 Cores nos Dias do Calendário

Para a tela do calendário ter o efeito das cores em cada dia, é necessário que cada elemento de dia dentro do *LazyGrid* de cada mês dentro do *ForEach* do ano saiba quais são as suas tarefas.

Foi criada uma função, descrita no Algoritmo 5, na classe *UserManager* que recebe uma data e retorna um vetor de *TaskModels*. Por essa classe ter acesso a todas as estruturas locais do usuário, a função é capaz de iterar sobre o vetor de tarefas e fazer um cálculo que compara os inteiros de semana e mês dentro da estrutura em questão com a

Algoritmo 3 Parte da Date Extension

```
var dayWeekDayInt: Int {
    var calendar: Calendar { Calendar.current }
    return (calendar.component(.weekday, from: self) -
    calendar.firstWeekday + 7) % 7
}
var YMD: Int {
    return self.yearInt * 10000 + self.monthInt * 100
    + self.dayInt
}
func monthsNextYear() -> [Date] {
    let currentDate = Date()
    let currentCalendar = Calendar.current
    let currentMonth = currentCalendar.component(.month,
    from: currentDate)
    let currentYear = currentCalendar.component(.year,
    from: currentDate)
    var months: [Date] = []
    for monthIndex in currentMonth...currentMonth + 11 {
    let adjustedMonthIndex = (monthIndex - 1) % 12 + 1
    // Wrap around to handle months beyond December
    let month = currentCalendar.date(from:
    DateComponents(year: monthIndex <= 12 ? currentYear :
    currentYear + 1, month: adjustedMonthIndex))
        if let month = month {
            months.append(month)
    }   }
return months
}
func calendarDisplayDays() -> [Date]{
    var days: [Date] = []
    for dayOffset in 0..<self.numberOfDaysInMonth() {
        let newDay = Calendar.current.date(byAdding:
        .day, value: dayOffset, to: startOfMonth())
        days.append(newDay!)
    }
    for dayOffset in 0.<startOfPreviousMonth()
    .numberOfDaysInMonth() {
    let newDay = Calendar.current.date(byAdding:
        .day, value: dayOffset, to: startOfPreviousMonth())
        days.append(newDay!)
    }
    return days.filter { $0 >= sundayBeforeStart()
    && $0 <= endOfMonth()}.sorted(by: <)
}
func isToday() -> Bool{
    return self.dayInt == Date.now.dayInt
    && self.monthInt == Date.now.monthInt
    && self.yearInt == Date.now.yearInt
}
```

Algoritmo 4 Função de ordenação de tarefas

```
static func < (lhs: TaskModel, rhs: TaskModel) -> Bool {
    return lhs.distance() < rhs.distance()
}
func distance() -> Int {
    var daysToHappen: Int
    if self.selectedPeriod == Period.monthly.description {
        if self.monthDays == [-1] {
            daysToHappen = Int.max - 1
        } else {
            if let gotDay = self.monthDays.first(where:
            {$0 >= Date.now.dayInt}) {
                //se há um dia entre hoje e o fim do mes
                daysToHappen = gotDay - Date.now.dayInt
            } else {
                daysToHappen = (Date.now.numberOfDaysInMonth()
                + self.monthDays[0]) - Date.now.dayInt
            }
        }
    } else {
        if self.weekDays == [-1] {
            daysToHappen = Int.max - 1
        } else {
            if let gotDay = self.weekDays.first(where:
            {$0 >= Date.now.dayWeekDayInt}) {
                //se há um dia entre hoje e o fim da semana
                daysToHappen = gotDay - Date.now.dayWeekDayInt
            } else {
                daysToHappen = (7 + self.weekDays[0])
                - Date.now.dayWeekDayInt
            }
        }
    }
    if self.dueDate > Date.distantPast {
        daysToHappen = min(self.dueDate.YMD - Date.now.YMD,
        daysToHappen)
        if self.dueDate.YMD < Date.now.YMD {
            daysToHappen = Int.max
            //Int max means the task has already
            been finished and there are no more updates
        //it will forever be "Completed",
        until the user deletes it
        }
    }
    return daysToHappen
}
```

data recebida.

Além disso, a classe também tem acesso a todas as postagens feitas pelo usuário no *feed* das salas, ou seja, quando uma tarefa é concluída em um dia, é possível ignorá-la e não inseri-la no vetor que será enviado ao calendário. Assim, o calendário só contêm tarefas ainda a serem realizadas.

4.3.4 Criação de Tarefa

A criação de uma tarefa necessita de mais atenção pois a lógica utilizada é contra-intuitiva. Não se pode utilizar a tarefa que está sendo representada na interface em que se tem a edição.

Isso ocorre pois na criação de uma estrutura localmente não se tem o *record* que representará essa estrutura no banco de dados. Então, se a criação for feita com a mesma estrutura utilizada na interface, após a criação, o usuário não tera poder de edição na estrutura guardada na nuvem.

Para que seja possível tanto a criação quanto a edição posterior, e o cancelamento de edições indesejadas, é necessário criar uma estrutura temporária de tarefa.

No caso da criação, é criada essa estrutura, a qual o usuário pode editar livremente, e após o apertar do botão de salvar é feito o envio para o banco de dados, que retorna, após a operação ser finalizada, o *CKRecord* (*Cloud Kit Record*) da tarefa enviada.

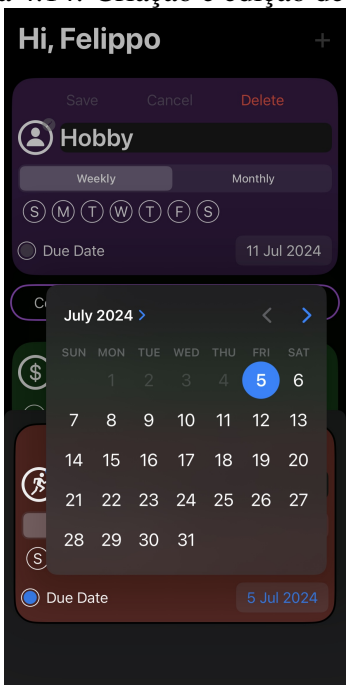
A cópia recebe esse *record* e a tela com a lista de tarefas recebe a estrutura após a cópia estar pronta com o *record*. São duas as maneiras de fazer esse dado estar presente na estrutura apresentada pela interface, uma é a maneira mencionada, e outra é com a utilização de um *delegate* na *escaping closure* (retorno do *CKRecord* ao fim do envio para o banco) para que a *ViewModel* da lista conheça a alteração feita na *ViewModel* da tarefa editada. Foi escolhido método da cópia pois segue mais precisamente as convenções da arquitetura, e também possibilita o cancelamento de edições indesejadas se o usuário quiser fazer alterações em outros momentos, já que a tarefa original serve como uma memória do estado anterior às edições.

A tarefa em modo edição e a tela de criação podem ser visualizadas na Figura 4.14.

Algoritmo 5 Função que descobre as tarefas em uma data

```
func findTasksOnDay(day: Date) -> [TaskModel]{
    var tasks: [TaskModel] = []
    for task in self.userTasks {
        if day.yearInt < task.dueDate.yearInt ||
        task.dueDate == .distantPast {
            if addMonthlyOrWeeklyTask(day: day, task: task){
                if !self.userCompletions.contains(where:
                {$0.taskId == task.id
                && $0.completionDate.YMD == day.YMD}) {
                    tasks.append(task)
                }
            }//se o ano for menor ou nao tem, coloca
        } else {
            if day.yearInt == task.dueDate.yearInt {
                if (day.YMD < task.dueDate.YMD) {
                    if addMonthlyOrWeeklyTask(day: day,
                    task: task) {
                    if !self.userCompletions.contains(where:
                        {$0.taskId == task.id
                    && $0.completionDate.YMD == day.YMD}){
                            tasks.append(task)
                        }
                    }
                } else {
                    if day.YMD == task.dueDate.YMD {
                    if !self.userCompletions.contains(where:
                    {$0.taskId == task.id &&
                    $0.completionDate.YMD == day.YMD}) {
                            tasks.append(task)
                        }
                        //se for o dueDate coloca
                    }
                }
            }
        }
    }
    return tasks
}
```

Figura 4.14: Criação e edição de tarefas



Fonte: Autor

Alguns campos são preenchidos com estruturas nativa em *iOS* disponibilizadas por *SwiftUI*. O campo de texto é editável em uma *TextField*, o campo do prazo utiliza de um calendário nativo com a estrutura *DatePicker*, e a opção de guardar as datas pelos dias da semana ou do mês é feita com a *Picker*.

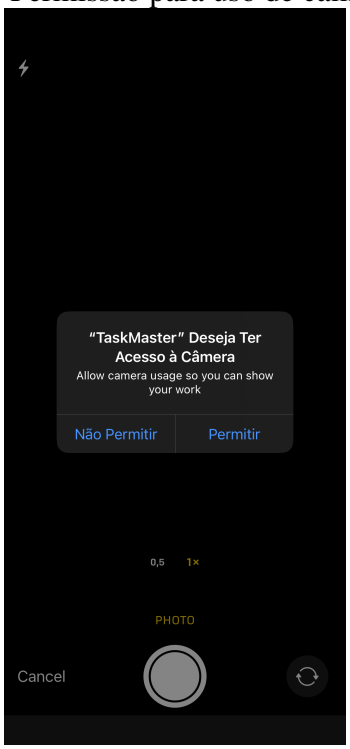
4.3.5 Tarefa Completa

No momento que o usuário completa uma tarefa, ele pode ir no aplicativo e marcar que ela foi feita. Mas para que a sala com os amigos receba essa tarefa, é necessário que ele envie uma foto que vá para o *feed*.

Para possibilitar o uso da câmera no aplicativo, é necessário incluir nas informações do projeto a requisição de utilização da câmera, e em código é necessário requisitar o uso no momento em que a câmera for aberta.

A funcionalidade da câmera se demonstra muito mais fácil de ser acessada através de *UIKit*, a linguagem antecessora a *Swift e SwiftUI*, então as lógicas por traz de sobrepor as telas do aplicativo com a tela nativa da câmera, guardar uma foto e deletar foram feitos em *UIKit*. A tela da câmera e o pedido de permissão de sua utilização podem ser vistos na Figura 4.15.

Figura 4.15: Permissão para uso de câmera e sua tela



Fonte: Autor

4.3.6 Pontuação

Toda vez que o usuário completa uma tarefa que está importada em uma sala em um dia que ela deve ser cumprida, ele recebe 3 pontos na respectiva sala caso tenha compartilhado a foto com seus amigos. A implementação destes condicionais e a iteração nos vetores de salas e tarefas podem ser visualizados no Algoritmo 6

4.4 Nuvem

Foram adicionadas as capacidades de interação com o *CloudKit* e conta do *iCloud* no projeto. Após as etapas de configurações de permissões na *App Store Connect* (plataforma de desenvolvedores) foi iniciada a implementação das funções de *create, read, update e delete*.

Algoritmo 6 Atribuição de pontos a uma sala

```
func addCompletion(taskCompletion: CompletionModel){
    if taskCompletion.displays.contains("Room") {
        if !self.userCompletions.contains(where:
        {$0.taskId == taskCompletion.taskId
        && $0.completionDate.YMD == Date.now.YMD})
        && self.userTasks.first(where:
        {$0.id == taskCompletion.taskId})!.distance()==0{
            userTasks.first(where: {$0.id ==
            taskCompletion.taskId})?.roomIDs
            .forEach { roomID in
                if let roomIndex=userRooms.firstIndex(where:
                {$0.id == roomID}){
                    DispatchQueue.main.async{
                    }
                }
            }
        }
    }
    self.userAppendCompletion(completionId: taskCompletion.id)
    DispatchQueue.main.async{
        if self.userCompletions.isEmpty {
            self.userCompletions = [taskCompletion]
        } else {
            self.userCompletions.append(taskCompletion)
        }
    }
}
```

4.4.1 CRUD

No projeto, há uma classe responsável por todas as comunicações com a nuvem, mais especificamente, com o *CloudKit*. Essa classe se chama *CloudkitService* e é uma variável estática no código, assim se comporta como um *Singleton*. Isso é feito para que as variáveis internas, como o status de conexão do usuário e suas funções de envio e download sejam acessíveis por todas as outras classes do projeto.

Primeira responsabilidade desta classe é verificar se o usuário já está registrado no banco de dados e isso é feito através de uma função que busca o identificador do usuário na nuvem e lança erros caso ele não tenha sido encontrado. Isso pode ser visualizado no Algoritmo 7.

Algoritmo 7 Fetch User Record ID

```
func getUserRecordId(completion:
@escaping ((Result<String, Error>) -> Void))
{
    self.container.fetchUserRecordID(completionHandler:
    { (recordId, error) in
        if let name = recordId?.recordName {
            self.currentUserID = name
            completion(.success(name))
        }
        else if let error = error {
            completion(.failure(error))
        }
    })
}
```

Além de encontrar o usuário, é necessário buscar todas as suas tarefas e salas para proporcionar a persistência dos dados. Levando isso em consideração, e também a utilidade de buscar os outros usuários na sala, com suas respectivas tarefas concluídas e pontuações, é necessário que o serviço tenha um função de *read / download / fetch*. Esta foi implementada de diversas maneiras com a sobrecarga de funções para possibilitar buscas mais ou menos específicas dependendo do contexto de onde forem chamadas, uma inicialização que recebe um predicado pode ser vista no Algoritmo 8.

Porém, um *fetch* e um *getUser* só conseguem trazer algo do banco quando há algo lá, então, para a criação da tarefa e do usuário é necessário implementar um *create*, que também servirá para atualização dos dados já existentes quando forem editados. Também foi feita a sobrecarga dos métodos de *create* e uma implementação que representa sua

Algoritmo 8 CRUD: Read

```
func fetchData<T: Recordable>(predicate: NSPredicate, type:
T.Type, completion: @escaping (Result<T, Error>) -> Void) {
    let query: CKQuery = CKQuery(recordType:
    String(describing: T.self), predicate: predicate)
    self.fetchData(query: query, type: T.self) { result in
        switch result {
        case .success(let data):
            completion(.success(data))
        case .failure(let error):
            completion(.failure(error))
        }
    }
}
```

funcionalidade pode ser vista no Algoritmo 9.

Algoritmo 9 CRUD: Create

```
func saveData<T: Recordable>(data: T)
async throws -> CKRecord {
    let record = data.toRecord()
    do {
        let a =
        try await container.publicCloudDatabase.save(record)
        return a
    } catch {
        print("Houve um erro:", error.localizedDescription)
        throw error
    }
}
```

Para a implementação do *update*, é utilizada uma função de *save* diferente daquela disposta no Algoritmo 9 após uma verificação da existência do dado no banco. É possível ver como isso é implementado em código no Algoritmo 10.

E por fim, para finalizar que a classe implemente tudo o que é esperado de um serviço de conexão com o banco de dados, foi criado o *delete*, o mais simples dos métodos como visível no Algoritmo 11.

Algoritmo 10 CRUD: Update

```
func update(data: Recordable) async throws {
        guard let record = data.record else {
            let error =
            NSError(domain: "record não encotrado", code: 0)
            throw error
        }

        record.setValues(data: data)
        try await saveData(data: record)
}
```

Algoritmo 11 CRUD: Delete

```
private func delete(data: CKRecord) async throws {
    let recordID = data.recordID
    try await container.publicCloudDatabase.deleteRecord
    (withID: recordID)
}
```

4.4.2 Usuários

O modelo de *User* é composto por seu *CKRecord*, por um identificador único, o nome escolhido pelo usuário, um vetor com os identificadores de suas tarefas, um vetor com os identificadores de suas publicações, um vetor com os identificadores das salas as quais o usuário participa, a última data em que houve uma atualização no usuário, e um vetor com a pontuação do usuário em cada sala.

A decisão de fazer a lógica de pontuação através de um vetor de pontos dentro do modelo do usuário, onde a pontuação deste está guardada no mesmo índice de sua respectiva sala, se deu para que não houvessem alterações em estruturas por diferentes usuários.

O *CloudKit* não oferece funções na nuvem, para tratar de problemas de concorrência seria necessário a utilização de uma API, o que sairia do escopo de utilização das tecnologias *Apple*. Então, cada usuário é responsável pelas suas estruturas que só podem ser lidas pelos outros acessos na nuvem mas não editadas.

O prejuízo causado por essa decisão é na lógica e consistência de manter as salas e suas pontuações sempre pareadas. É necessário um cuidado redobrado se forem feitas

futuras modificações em um destes aspectos (salas e pontuações).

4.4.3 Tarefas

As tarefas são compostas também por um *CKRecord*, um identificador único e um nome, mas além disso, o usuário também pode escolher o tipo de tarefa, dias em que ela deve ser realizada e prazo final.

Conhecendo o tipo da tarefa através de uma *String* que traduz diretamente para um *enum* criado, é possível extrair também o ícone e cor através de funções do próprio *enum*.

Há um campo que determina se os dias guardados da tarefa representam dias da semana ou dias do mês, assim o campo de dias em inteiro não acaba sendo ambíguo, por exemplo, 7 representando sábado ou dia 7 do mês.

No mesmo raciocínio de evitar problemas de concorrência, as salas nas quais a tarefa é importada também têm seus identificadores guardados em um vetor do *TaskModel*. Assim como guardar pontuação na sala, ter os índices das tarefas na sala faria com que cada importação ou exclusão de tarefa trouxesse risco de ser desconsiderado caso outro usuário também fizesse um *update* no mesmo *room*.

4.4.4 Publicação do Feed

As publicações possuem um campo que demarca se ela foi enviada para os quartos ou não, esse campo é em *String* para possibilitar no futuro outros destinos para as publicações. Além disso e dos que estão em todas estruturas (*CKRecord e ID*) há também o identificador da tarefa respectiva a publicação, e a data em que foi feita.

Assim, é possível filtrar na sala apenas por tarefas que foram concluídas no último dia, ou no último mês caso seja feita essa alteração para mostrar um *Feed* mais antigo.

4.4.5 Sala

A sala é a única estrutura apta a ser altera por usuários que não a criaram. Com a possibilidade de múltiplas tentativas de atualização simultâneas, ela sofre o risco de inconsistência ou perda de dados. Considerando isso, e o fato de que *Cloud Kit* não

possui funções na nuvem que evitam estes problemas, as estruturas foram todas moldadas para que os dados utilizados na sala estivessem salvos em outros modelos.

Os campos que o *RoomModel* possui são *CKRecord*, id da sala, id do criador, nome, senha da sala e código de acesso. Como mencionado anteriormente, invés de ter os identificadores dos usuários e tarefas importadas, é o id da sala que está salvo nos usuários e tarefas.

E a sala só tem ativada na *App Store Connect* a capacidade de edição por outros usuários para dar suporte a saída do criador sem a exclusão da mesma na nuvem. Então na saída de um membro, é feita uma requisição que retorna os membros presentes. Se o retorno for apenas o usuário local, então este aciona o *delete* do *CRUD*.

5 PUBLICAÇÃO

Nesse capítulo são feitas as considerações sobre o produto publicado. Na seção 5.1 é abordada a publicação do aplicativo e os *feedbacks* dos usuários que utilizaram até o presente momento. E na seção 5.2 são expostas as ideias para melhorar o aplicativo e traçar os próximos passos do produto e das estratégias para estabelecê-lo no mercado.

5.1 Feedback de usuários

A publicação foi feita no *TestFlight* como demonstrado na Figura 5.1. Usuários de iPhone com o *link* e com o modo de desenvolvedor ativado no celular têm acesso ao aplicativo.

Figura 5.1: Aplicativo disponível no TestFlight

Fonte: Autor

O projeto foi testado com 6 usuários que reportaram a falta de algumas funcionalidades. Um usuário disse "Senti falta da opção de colocar um horário pra concluir minha *task*, poderia ter um campo para escolher uma hora e um minuto na edição da tarefa."com isso é possível notar que os usuários esperam ainda mais conformidade com outros aplicativos já existentes pois é uma das características trazidas no *benchmark*, ou seja, para se manter com um produto competitivo, não é uma opção deixar de lado uma funcionalidade

como essa do horário, por mais que o projeto tenha o diferencial das salas.

Disse outro *tester* que "Gostaria de ter uma notificação que me avisasse a hora de concluir uma tarefa pois tenho muito problema em me lembrar das minhas tarefas e é mais por isso que utilizo de aplicativos de organização". Com isso se torna mais explícita outra utilidade do horário e também que mostra a importância de notificações, tanto para lembrar que o usuário deve fazer suas tarefas quanto para avisá-lo que seus amigos já realizaram suas tarefas e há movimento no *feed* da sala.

Atualmente não há nenhuma explicação de como utilizar o aplicativo, então alguns usuários ficaram confusos quando sua pontuação havia zerado ao fim do mês. Isso é possível extrair através do comentário "Gostei bastante do *app* mas faço o comentário pois encontrei um *bug*, alguma coisa aconteceu, não sei exatamente o que eu fiz, mas minha pontuação e a dos meus amigos zerou, e eu nem havia atualizado o aplicativo". Também não está explicado como funciona o *feed* da sala e o passo a passo para completar as tarefas, que é outra coisa presente nos comentários: "Só fiquei com dúvida em uma questão, na tela de conclusão, eu não sei se eu ganho pontos na sala quando não envio foto, ou se ganho quando tiro foto mandando nas salas apenas, e se uma pessoa pode completar múltiplas vezes para ganhar mais pontos, isso é algo que descobri testando apenas, seria bom ter algo explicando isso explicitamente, fora isso, muito bom."

Houveram outros *feedbacks* com mais comentários, usuários comentando sobre tamanho de botões, cores, posicionamentos, *bugs* que encontraram e várias ideias para serem utilizadas em futuras atualizações do aplicativo.

5.2 Próximos Passos

Esses *feedbacks* serão levados em consideração nos próximos passos do projeto. O plano consiste em implementar um *OnBoarding* para explicar ao usuário as funcionalidades e os caminhos dentro da aplicação, também considerar a possibilidade de adicionar uma API ou trocar o banco de dados utilizado para simplificar a lógica que previne problemas de concorrência.

Adição de horário e notificação são imprescindíveis para alcançar o objetivo do projeto que é de verdadeiramente trazer uma solução em aplicação móvel que motive usuários sem distrações a realizarem suas tarefas e alcançarem seus objetivos.

Um *redesign* considerando um *branding*, renomeação do aplicativo, uma identidade visual melhor definida e todas esses aspectos que remetem ao apelo visual e captação

de usuários pela aparência do aplicativo serem desenvolvidas por um profissional da área.

Dar inicio ao marketing e captar mais usuários de diferentes ciclos sociais para ter mais retorno sobre possíveis *features* e ideias que possam agregar ao projeto. Criar uma página no *Instagram* para compartilhar fotos do aplicativo, com a possibilidade de criar anúncios.

Há também a consideração de ter implementar a funcionalidade de seguir outros usuários, com isso cada um ter acesso à todo o progresso feito por usuários seguidos sem estar em uma sala com eles. Claro, considerando publicações com opção de serem compartilhadas com todos ou apenas com as salas.

Se fosse implementado, isto daria um aspecto mais forte indo para a linha de rede social, fazendo com que houvesse uma página de usuário com suas publicações e uma outra tela na *tab bar* que levaria o usuário para o *feed* com todos os seus seguidores.

Aplicação de algum modelo de negócio para a monetização do aplicativo e reinvestimento no mesmo para a escalabilidade. E acima de tudo, a publicação na *App Store*. Será considerada a implementação para dispositivos *Android*, assim aumentando o alcance e captação de usuários, isso também trará a necessidade de troca de banco de dados.

6 CONCLUSÃO

Este projeto trouxe pesquisas psicológicas, sociais e tecnológicas pra encontrar problemas, soluções e as melhores maneiras de desenvolver uma aplicação com o objetivo de motivação e disciplina. Foi então preparado o ambiente no qual seria desenvolvido o aplicativo e feito um estudo por trás de cada ferramenta que poderia ser utilizadas, para entender como melhor desfrutar daquilo que está disposto aos programadores *mobile*, e principalmente, *iOS*.

A confecção deste aplicativo serviu de grande aprendizado sobre aplicações móveis. Foi possível compreender todo o processo por trás de estruturação do projeto por trás do código e até mesmo disponibilização para o público.

Não só foi possível concluir um projeto orientado a objeto com múltiplas classes respeitando uma hierarquia específica, mas também foi aprofundada essa prática com o contexto de telas e controladores. A conversa entre duas linguagens, neste caso *SwiftUI* e *UIKit* é outro aspecto interessante de projetos em maior escala que foi possível ter contato através deste aplicativo.

Todo o processo de estudo por traz da implementação do banco de dados nativo, utilização de perfis de permissões na *App Store Connect*, considerações e testes por outras soluções com importações de pacotes externos ao ambiente de desenvolvimento do *XCode*, tudo serviu de aprendizado para a melhor visualização de como são possibilitados os maiores e mais utilizados softwares de uso pessoal e diário ao redor do mundo.

Estudos de mercado e *benchmarks* para descobrir quais são as soluções já existentes, sua eficiência, a consideração de *feedbacks* de usuários de outros aplicativos e toda a etapa de pesquisa que antecedeu a programação serve também como uma experiência muito importante que agrega para qualquer área voltada ao desenvolvimento de produtos e ao empreendimento.

Com isso se tem uma ideia clara dos próximos passos a serem tomados no projeto. São várias as possíveis funcionalidades a agregar como implementação de notificações em um horário específico para cada tarefa, refinação da interface visual e identidade do aplicativo, criação de um *feed* público com a possibilidade de seguir amigos, retrospectivas mensais para deixar claro o progresso do usuário em seus objetivos, *widgets* para relembrar os usuários constantemente de realizarem suas tarefas, e mais várias ideias que ainda estão por vir dos usuários que futuramente utilizarão o aplicativo.

REFERÊNCIAS

AJZEN, I. From intentions to actions: A theory of planned behavior. In: KUHL, J.; BECKMANN, J. (Ed.). **Action control: From cognition to behavior**. [S.l.]: Springer, 1985. p. 11–39.

ALBERTAZZI, D.; FERREIRA, M.; FORCELLINI, F. A wide view on gamification. **Technology, Knowledge and Learning**, v. 24, n. 2, p. 191–202, 2019.

APALON, A. **Hábitos e tarefas - Productive**. 2015. <https://apps.apple.com/br/app/h%C3%A1bitos-e-tarefas-productive/id983826477>. Accessed: 2023-12-05.

APPLE, D. **Apple Documentation**. 2024. Disponível na Internet: <https://developer.apple.com/documentation>.

APPLE, D. **Previewing your app's interface in Xcode**. 2024. Disponível na Internet: <https://developer.apple.com/documentation/xcode/previewing-your-apps-interface-in-xcode>.

APPLE, D. **SF Symbols**. 2024. Disponível na Internet: <https://developer.apple.com/sf-symbols/>.

APPLE, D. **Tab Bar**. 2024. Disponível na Internet: <https://developer.apple.com/design/human-interface-guidelines/tab-bars>.

APPLE, D. **View**. 2024. Disponível na Internet: <https://developer.apple.com/documentation/swiftui/view>.

Apple Developer. **App Store**. 2024. <https://developer.apple.com/app-store/>. Accessed: 2023-12-01.

Apple Developer. **CloudKit**. 2024. <https://developer.apple.com/documentation/cloudkit/>. Accessed: 2023-12-03.

Apple Developer. **Core Data**. 2024. <https://developer.apple.com/documentation/coredata/>. Accessed: 2023-12-03.

Apple Developer. **HealthKit**. 2024. <https://developer.apple.com/documentation/healthkit/>. Accessed: 2023-12-04.

Apple Developer. **Human Interface Guidelines**. 2024. <https://developer.apple.com/design/human-interface-guidelines>. Accessed: 2023-12-01.

Apple Developer. **iCloud**. 2024. <https://developer.apple.com/design/human-interface-guidelines/icloud/>. Accessed: 2023-12-03.

Apple Developer. **SwiftUI**. 2024. <https://developer.apple.com/documentation/swiftui/>. Accessed: 2023-12-01.

Apple Developer. **TestFlight**. 2024. <https://developer.apple.com/testflight/>. Accessed: 2023-12-01.

Apple Developer. **UIKit**. 2024. <https://developer.apple.com/documentation/uikit/>. Accessed: 2023-12-01.

Apple Developer. **WatchKit**. 2024. <https://developer.apple.com/documentation/watchkit/>. Accessed: 2023-12-04.

AVOCADO APPS, L. **GymRats · Desafio fitness**. 2019. <https://apps.apple.com/br/app/gymrats-desafio-fitness/id1453444814>. Accessed: 2023-12-06.

BALDISSERA, O. O que é gamificação e como ela aumenta o engajamento. **PósPucPRDigital**, 2021.

BARATA, G. et al. Studying student differentiation in gamified education: A long-term study. **Computers in Human Behavior**, v. 71, p. 550–585, 2017.

BAYUK, J.; ALTOBELLO, S. Can gamification improve financial behavior? the moderating role of app expertise. **International Journal of Bank Marketing**, v. 37, n. 4, p. 951–975, 2019.

CHEN, G.; LYU, C. The relationship between smartphone addiction and procrastination among students: A systematic review and meta-analysis. **Personality and Individual Differences**, v. 224, p. 112652, 2024. ISSN 0191-8869. Disponível na Internet: <https://www.sciencedirect.com/science/article/pii/S0191886924001120>.

CHERRY, M. K. What is procrastination? putting off tasks we don't enjoy is common, despite the consequences. 11 2022.

DAVETECH CO., L. **Habit Tracker**. 2018. <https://apps.apple.com/br/app/habit-tracker/id1438388363?platform=iphone>. Accessed: 2023-12-06.

DIMENICHI, B. C.; TRICOMI, E. The power of competition: Effects of social motivation on attention, sustained physical effort, and learning. **Frontiers in Psychology**, v. 6, p. 1282, September 1 2015.

DING, L.; KIM, C.; OREY, M. Studies of student engagement in gamified online discussions. **Computers & Education**, v. 115, p. 126–142, 2017.

DISSANAYAKE, I. et al. Competition matters! self-efficacy, effort, and performance in crowdsourcing teams. **Information and Management**, v. 56, n. 8, p. 103–158, 2019.

ENERJOY, P. L. **Me+ Daily Routine Planner**. 2023. <https://apps.apple.com/br/app/me-daily-routine-planner/id1596403446>. Accessed: 2023-12-06.

FABULOUS. **Fabulous: Habit Tracker, Saude**. 2013. <https://apps.apple.com/br/app/fabulous-habit-tracker-saude/id1203637303>. Accessed: 2023-12-05.

FERRARI, P. D. J. **The Prevalence of Procrastination Everyone isn't doing it.** 2020. Acesso em: 25 de Junho de 2024. Disponível na Internet: <https://www.psychologytoday.com/us/blog/still-procrastinating/202010/the-prevalence-procrastination>.

FLYNN, J. **20 TELLING PROCRASTINATION STATISTICS [2023]: THE PREVALENCE OF PROCRASTINATION**. 2023. Acesso em: 25 de Junho de 2024. Disponível na Internet: <https://www.zippia.com/advice/procrastination-statistics/>.

GUERIN, D. W. et al. Childhood and adolescent antecedents of social skills and leadership potential in adulthood: Temperamental approach/withdrawal and extraversion. **The Leadership Quarterly**, v. 22, p. 482–494, 2011.

HAMARI, J.; KOIVISTO, J.; SARSA, H. Does gamification work? - a literature review of empirical studies on gamification. In: **47th Hawaii International Conference on System Sciences**. [S.l.: s.n.], 2014. p. 3025–3034.

HUANG, B.; HEW, K. Implementing a theory-driven gamification model in higher education flipped courses: Effects on out-of-class activity completion and quality of artifacts. **Computers & Education**, v. 125, p. 254–272, 2018.

HUANG, B.; HEW, K.; LO, C. Investigating the effects of gamification-enhanced flipped learning on undergraduate students' behavioral and cognitive engagement. **Interactive Learning Environments**, v. 27, n. 8, p. 1106–1126, 2019.

HUDSON, P. **All SwiftUI property wrappers explained and compared**. 2024. Disponível na Internet: <https://www.hackingwithswift.com/quick-start/swiftui/all-swiftui-property-wrappers-explained-and-compared>.

IOANNOU, A. A model of gameful design for learning using interactive tabletops: Enactment and evaluation in the socio-emotional education classroom. **Educational Technology Research and Development**, v. 67, n. 2, p. 277–302, 2019.

IRWIN, B. C. et al. Aerobic exercise is promoted when individual performance affects the group: A test of the kohler motivation gain effect. **Annals of Behavioral Medicine**, Springer, v. 44, n. 2, p. 151–159, 2012.

KOIVISTO, J.; HAMARI, J. The rise of motivational information systems: A review of gamification research. **International Journal of Information Management**, v. 45, p. 191–210, 2019.

KRATH, J.; SCHüRMANN, L.; von Korflesch, H. F. Revealing the theoretical basis of gamification: A systematic review and analysis of theory in research on gamification, serious games and game-based learning. **Computers in Human Behavior**, v. 125, p. 106963, 2021.

LAYNE, J. **Procrastination Statistics.** 2024. Acesso em: 25 de Junho de 2024. Disponível na Internet: <https://medium.com/statistic-hub/procrastination-statistics-1ef2ca5124fb>.

MILLEFEUILLE, A. **Eden - Rotina diaria & hábitos**. 2023. <https://apps.apple.com/br/app/eden-rotina-diaria-h%C3%A1bitos/id1589243137>. Accessed: 2023-12-05.

MORGANTI, L. et al. Gaming for earth: Serious games and gamification to engage consumers in pro-environmental behaviours for energy efficiency. **Energy Research and Social Science**, v. 29, p. 95–102, 2017.

MORSCHHEUSER, B.; HAMARI, J.; MAEDCHE, A. Cooperation or competition – when do people contribute more? a field experiment on gamification of crowdsourcing. **International Journal of Human-Computer Studies**, v. 127, p. 7–24, 2019.

ORJI, R.; MOFFATT, K. Persuasive technology for health and wellness: State-of-the-art and emerging trends. **Health Informatics Journal**, v. 24, n. 1, p. 66–91, 2018.

PALEBLUEDOT. **Easy Habit Tracker & Routine**. 2023. <https://apps.apple.com/br/app/easy-habit-tracker-routine/id1525432202>. Accessed: 2023-12-06.

PALIASHCHUK, Y. **Habitty - Lembretes Metas**. <https://apps.apple.com/br/app/habitty-lembretes-metas/id1228128916>. Accessed: 2023-12-05.

PASSALACQUA, M. et al. Playing in the backstore: Interface gamification increases warehousing workforce engagement. **Industrial Management and Data Systems**, 2019.

PERRYER, C. et al. Enhancing workplace motivation through gamification: Transferrable lessons from pedagogy. **International Journal of Management Education**, v. 14, n. 3, p. 327–335, 2016.

RACHELS, A. R.-S. J. The effects of a mobile gamification app on elementary students' spanish achievement and self-efficacy. **Computer Assisted Language Learning**, v. 31, n. 1–2, p. 72–89, 2018.

REEVE, J. Competition can enhance motivation—but typically undermines it. In: BONG, M.; REEVE, J.; KIM, S. il (Ed.). **Motivation Science: Controversies and Insights**. online edition. New York: Oxford Academic, 2023. Accessed 29 June 2024. Disponível na Internet: <https://doi.org/10.1093/oso/9780197662359.003.0028>.

ROUTINERY, C. **Routinery: Autocuidado/Rotina**. 2020. <https://apps.apple.com/br/app/routinery-autocuidado-rotina/id1450486923?platform=iphone>. Accessed: 2023-12-05.

RYAN, R.; DECI, E. **Self-determination theory. Basic psychological needs in motivation, development and wellness**. [S.l.]: The Guilford Press, 2017.

SAINSBURY, E.; WALKER, R. Motivation, learning and group work - the effect of friendship on collaboration. 01 2012.

SCHUNK, D. H. Achievement motivation in academics. In: **Encyclopedia of Applied Psychology**. [S.l.: s.n.], 2004. p. 35–40.

SEEKRTECH CO., L. **Forest - Mantenha o Foco**. 2014. <https://apps.apple.com/br/app/forest-mantenha-o-foco/id866450515>. Accessed: 2024-01-24.

SHATZ, P. I. **Procrastination Statistics: Interesting and Useful Statistics about Procrastination**. 2021. Acesso em: 25 de Junho de 2024. Disponível na Internet: <https://solvingprocrastination.com/procrastination-statistics/>.

SOFTWINGS, T. **Lembretes Da Caderno Digital**. 2023. <https://apps.apple.com/br/app/lembretes-da-caderno-digital/id1632352471?platform=iphone>. Accessed: 2023-12-06.

TECHNOLOGIES, L. C. **Rotina Diaria + Habitos**. 2020. <https://apps.apple.com/br/app/rotina-diaria-habitos/id1477345602>. Accessed: 2023-12-05.

TIIMO. **Tiimo - Planner visual diário**. 2019. <https://apps.apple.com/br/app/tiimo-planner-visual-di%C3%A1rio/id1480220328?platform=iphone>. Accessed: 2023-12-05.

TOBON, S.; RUIZ-ALBA, J.; GARCíA-MADARIAGA, J. Gamification and online consumer decisions: Is the game over? **Decision Support Systems**, v. 128, 2020.

UNORDERLY, G. **Structured - Planejador Diário**. 2021. <https://apps.apple.com/br/app/structured-planejador-di%C3%A1rio/id1499198946?platform=iphone>. Accessed: 2023-12-05.

WEINER, B. **Human Motivation: Metaphors, Theories, and Research**. [S.l.]: Sage Publications, Inc, 1992.

ZAINUDDIN, Z. Students' learning performance and perceived motivation in gamified flipped-class instruction. **Computers & Education**, v. 126, p. 75–88, 2018.

ZAINUDDIN, Z. et al. The impact of gamification on learning and instruction: A systematic review of empirical evidence. **Educational Research Review**, v. 30, 2020.

ZEIDNER, M.; BOEKAERTS, M.; PINTRICH, P. Self-regulation: Direction and challenges for future research. In: **Handbook of Self-Regulation**. [S.l.: s.n.], 2000. p. 749–768.

ZHOU, X. et al. The correlation between mobile phone addiction and procrastination in students: A meta-analysis. **Journal of Affective Disorders**, v. 346, p. 317–328, 2024. ISSN 0165-0327. Disponível na Internet: <https://www.sciencedirect.com/science/article/pii/S0165032723013794>.