UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

The Huxley Mobile: um aplicativo móvel multiplataforma para acesso ao juiz online The Huxley

Trabalho de Conclusão de Curso

Mateus dos Santos Silva

São Cristóvão – Sergipe

2022

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Mateus dos Santos Silva

The Huxley Mobile: um aplicativo móvel multiplataforma para acesso ao juiz online The Huxley

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Alberto Costa Neto

São Cristóvão – Sergipe

2022

Este trabalho é dedicado aos meus professores, familiares e amigos.

Resumo

O Moodley é um aplicativo móvel que oferece apoio à aprendizagem aos usuários utilizando um ambiente virtual de aprendizagem (AVA), o Moodle. O Moodley também permite aos usuários visualizarem problemas e questionários, escrever código-fonte na linguagem de programação escolhida e enviar soluções avaliadas por meio do The Huxley, um juiz *online*, que também pode ser considerado um ambiente virtual de aprendizagem. O aplicativo Moodley oferece apenas uma parcela das funcionalidades que existem no The Huxley, e está disponível apenas na plataforma Android. Assim, para ampliar as funcionalidades disponíveis para o perfil aluno do The Huxley e viabilizar a disponibilidade nas duas principais plataformas de dispositivos móveis (Android e iOS), foi desenvolvido o aplicativo The Huxley Mobile, uma evolução do aplicativo Moodley, utilizando Flutter como *framework* para desenvolvimento multiplataforma. Ademais, o novo App oferece apenas funcionalidades do The Huxley, já que existe o App móvel oficial do Moodle, o qual tem atendido muito bem as necessidades dos usuários, além disso, outros ambientes virtuais de aprendizagem cresceram em participação devido à digitalização ocorrida durante a pandemia de COVID-19.

Palavras-chave: moodley; the huxley; juiz *online*; aplicativo móvel;

Abstract

Moodley is a mobile application that provides learning to users using a virtual learning environment (VLE), Moodle. The Moodley also allows users to view problems and quizzes, write source code in their chosen programming language and submit solutions that are evaluated by The Huxley, an on-line judge, which can also be considered a virtual learning environment. The Moodley App offers only a portion of the functionality available in The Huxley and only available on the Android platform. Thus, to expand the functionalities available for the student role of The Huxley and enable availability on the two main mobile device platforms (Android and iOS), The Huxley Mobile application was developed, an evolution of the Moodley application, using Flutter as a framework for multiplatform development. In addition, the new App only offers The Huxley features, since there is the official Moodle App, which has served user needs very well. In addition, other virtual learning environments have grown in participation due to the digitization that occurred during the COVID-19 pandemic.

Keywords: moodley. the huxley. judge on-line. mobile app.

Lista de ilustrações

Sumário

1 Introdução

Segundo Barbosa, Oliveira e D'Carlo (2016), a adoção de aplicativos móveis no contexto educacional vem crescendo e abre espaço para o *mobile learning (m-learning)*, que envolve o uso do dispositivo móvel, sozinho ou em combinação com outras tecnologias de informação e comunicação (TICs) para promover o aprendizado.

Conforme descrito por Petit, Giménez e Roura (2012), algo bem estabelecido é a prática ser fundamental para aprender programação. Durante a aprendizagem, os professores atribuem problemas de programação aos alunos iniciantes para ajudá-los a adquirir essa habilidade. Esses problemas geralmente requerem o desenvolvimento de um algoritmo e a escrita de um programa que, dadas algumas entradas, produza as saídas correspondentes. A maioria desses problemas coloca sua ênfase na compreensão ao invés de conhecimento técnico, ou seja, pensando em como combinar algoritmicamente algumas instruções básicas e estruturas de dados que aparecem em praticamente qualquer linguagem de programação moderna. Mas, o processo de correção das soluções dos alunos é longo e está sempre sujeito a erros. Pelo fato da natureza dos problemas que os professores enfrentam serem computacionais, essa correção é candidata para uma avaliação automatizada.

De acordo com Petit, Giménez e Roura (2012), os sistemas de correção automática são conhecidos como juízes *online*. Os juízes *online* são sistemas geralmente baseados em um repositório de problemas de programação e em uma maneira de enviar soluções para esses problemas, de modo a obter um veredicto sobre seu comportamento quando executados em diferentes conjuntos de dados públicos e privados.

Zhao et al. (2018) afirma que, ao contrário de outros sistemas de aprendizagem *online*, em sua grande maioria, os juízes *online* são projetados para ter uma aprendizagem autodirigida sem a intervenção de professores.

O aplicativo Moodley é um exemplo de aplicativo educacional que permite acesso ao conteúdo de estudo por meio do Moodle e acesso aos problemas e questionários do The

Huxley. De acordo com Jesus (2018), seu maior diferencial é o apoio ao ensino-aprendizagem de programação através da capacidade de produzir mensagens de *feedback* que sejam facilmente compreendidas pelos usuários, norteando-os sobre os erros de sintaxe apresentados ao realizar uma submissão ao juiz *online*.

Contudo, o Moodley acabou não evoluindo com o avanço da tecnologia e dos padrões de usabilidade. Outro ponto negativo é que o Moodley possui somente disponibilidade na plataforma Android, já que foi desenvolvido nativamente.

Outro ponto importante é que diversos ambientes virtuais de aprendizagem cresceram devido à digitalização ocorrida durante a pandemia de covid-19, e um deles foi o aplicativo oficial do Moodle, que tem atendido muito bem às necessidades dos usuários.

Portanto, este trabalho consistiu no desenvolvimento do aplicativo The Huxley Mobile, oferecendo apenas as funcionalidades do The Huxley com novas funcionalidades disponíveis para o perfil do aluno. O desenvolvimento foi realizado utilizando Flutter como *framework* para desenvolvimento multiplataforma e, assim, viabilizando a disponibilidade nas duas principais plataformas de dispositivos móveis (Android e iOS). Ademais, é apresentada sua evolução visual e funcional com relação ao Moodley.

1.1 Justificativa

Neste trabalho foi utilizado o juiz *online* The Huxley pelo fato de ter a disponibilidade de uma API REST para conexão direta com a base de dados, permitindo acesso a diversas funcionalidades no sistema, e possuindo diversas ferramentas de análise de desempenho dos alunos, definição de exercícios, provas e verificação de plágio. Ademais, o The Huxley tem uma utilização relevante nas universidades do nordeste e do Brasil na totalidade.

1.2 Objetivos

1.2.1 Geral

Para corrigir os diversos problemas de usabilidade do Moodley e somente possuir disponibilidade na plataforma Android, esse trabalho teve como objetivo desenvolver um aplicativo móvel para o juiz *online* The Huxley. Assim, corrigindo os problemas existentes no Moodley e viabilizando a distribuição nas plataformas Android e iOS.

1.2.2 Específico

Visando atingir o objetivo de desenvolver um aplicativo móvel para o The Huxley, foram estabelecidos os seguintes objetivos específicos:

- Desenvolver um aplicativo para dispositivos móveis para o juiz *online* The Huxley que ofereça uma melhor experiência de usabilidade, uma nova proposta de design, com mais fluidez durante o uso e acesso *offline* de algumas funcionalidades. Além disso, que introduza uma nova forma de escolha de arquivos para submissão de código-fonte onde os resultados das mesmas sejam apresentados com mais detalhes;
- Publicar o aplicativo nas plataformas Android e iOS;
- Avaliar, através de uma pesquisa, o aplicativo desenvolvido junto aos usuários.

1.3 Organização do documento

Este trabalho está dividido em cinco capítulos. No primeiro, é apresentada a introdução, a contextualização, a justificativa, os objetivos e a organização do documento. O segundo aborda a fundamentação teórica. O terceiro contém os trabalhos relacionados. No quarto, aborda-se o desenvolvimento do aplicativo The Huxley Mobile, descrevendo os requisitos funcionais, não-funcionais, arquitetura, estratégia de integração e exemplos de requisitos desenvolvidos. No quinto, é apresentada a conclusão do trabalho.

2
Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos e fundamentos teóricos necessários ao embasamento do tema deste trabalho. Serão esclarecidas as definições de desenvolvimento híbrido na Seção 2.1 para poder desenvolver um aplicativo multiplataforma e arquitetura MVC na Seção 2.2 para explicar a arquitetura utilizada no desenvolvimento.

2.1 Desenvolvimento Multiplataforma

De acordo com Nunes (2021), no universo de desenvolvimento mobile existem dois grandes grupos de abordagens: a abordagem nativa e a multiplataforma. Aplicações nativas são aquelas desenvolvidas especificamente para uma plataforma, como Android ou iOS, cada uma possui suas linguagens de programação, plataformas e outras especificidades. Com isso, as aplicações multiplataforma são aplicativos desenvolvidos de forma que apenas um projeto é necessário para que o aplicativo funcione em várias plataformas, utilizando-se, assim, somente um ambiente de desenvolvimento e linguagem de programação.

Conforme Chebbi (2019), para desenvolver um aplicativo nativo utiliza-se uma linguagem específica para um determinada plataforma que só pode ser usada na plataforma em questão. Sendo assim, um App nativo iOS é desenvolvido conforme as características do sistema operacional e não pode ser instalado em um aparelho Android ou em qualquer outro sistema.

Apesar de que aplicativos nativos tenham a melhor usabilidade, segurança e desempenho em relação às funcionalidades da plataforma, isso acaba tendo um custo muito elevado quando se considera viabilizar a mesma aplicação em diversas plataformas. Santos (2022) afirma que atualmente, no mundo da tecnologia de desenvolvimento mobile, quando falamos de desenvolvimento multiplataforma, as tecnologias que logo vem à nossa mente são React Native e o Flutter. Assim, por serem duas tecnologias diferentes que possuem objetivos em comum (desenvolvimento de aplicações mobile), muitos debates giram em torno sobre qual realmente é

melhor.

2.1.1 Flutter

De acordo com Wu (2018), o Flutter é um *framework* multiplataforma baseado na linguagem Dart que visa o desenvolvimento de aplicações de alto desempenho. Flutter foi lançado em 2016 pelo Google. Não só os aplicativos Flutter podem ser executados no Android e iOS, mas também o Fuschia, o sistema operacional da próxima geração do Google que utiliza o Flutter como sua estrutura. O Flutter, diferente de várias tecnologias, ao invés de utilizar visualizações da *web*, ele renderiza todos os componentes de visualização usando seu próprio mecanismo de renderização de alto desempenho. Essa peculiaridade oferece a possibilidade de construir aplicativos de alto desempenho como os aplicativos nativos podem ser.

Flutter (2022) explica que o Flutter oferece o recurso de *hot reload* que ajuda o indivíduo a experimentar, criar *interfaces* de usuário de maneira rápida e fácil, adicionar recursos e corrigir bugs. O *hot reload* funciona injetando arquivos de código-fonte atualizados na máquina virtual Dart (VM) em execução. Depois que a VM atualiza as classes com as novas versões de campos e funções, a estrutura Flutter reconstrói automaticamente a árvore de *widgets*, permitindo que se visualize rapidamente os efeitos de suas alterações.

Macoratti (2022) comenta que os *widgets* são as partes da sua *interface* de usuário. Os textos são *widgets*. Botões são *widgets*. As caixas de seleção são *widgets*. Imagens são *widgets*. E a lista continua. Na verdade, tudo na *interface* do usuário é um *widget*. Até mesmo o aplicativo, é um *widget*.

Existem dois tipos principais de *Widgets*: *Stateless Widget* e *Stateful Widget*. A principal diferença é que um *Stateful Widget*, diferente do *Stateless Widget*, possui um "estado". Esse estado é considerado um componente interno do *Widget*.

Algumas das principais vantagens do Flutter são:

- Criação de *interface* flexível;
- Componentes leves;
- *Framework* reativo moderno;
- Suporte oficial da Google.

2.1.2 Escolha do *Framework* de Desenvolvimento

O *Framework* Flutter foi escolhido para ser utilizado no desenvolvimento da solução. O motivo da escolha se deu pelo conhecimento prévio do *Framework*, além dele atender a todas as funcionalidades previstas no projeto.

2.2 Arquitetura MVC no Flutter

Para Guedes (2022), MVC é um padrão de arquitetura de software. O MVC sugere uma maneira para você pensar na divisão de responsabilidades. O princípio básico do MVC é a divisão da aplicação em três camadas: a camada de interação do usuário (*view*), a camada de manipulação dos dados (*model*) e a camada de controle (*controller*). Com o MVC, é possível separar o código relativo à *interface* do usuário das regras de negócio. Cada uma das camadas apresenta geralmente as seguintes responsabilidades:

- *Model*: A responsabilidade dos *models* é representar objetos de negócio, como: aluno, turma, problema e outros. Também é responsável pelo acesso e manipulação dos dados na sua aplicação.
- *View*: A *view* é responsável pela *interface* que será apresentada, mostrando as informações do *model* para o usuário.
- *Controller*: É a camada de controle, responsável por ligar o *model* e a *view*. Nos *controllers* são implementadas as regras da aplicação e com isso fazendo com que os *models* possam ser repassados para as *views* e vice-versa.

Perry (2018) explica que, como muitos padrões de projeto, o MVC visa separar "as áreas de trabalho". Com isso, separa as áreas de responsabilidade ao desenvolver os aplicativos. Com o MVC, isso significa separar a *interface* da regra de negócio que compõe o aplicativo. Essa dissociação no software permite codificação modular eficiente, reutilização de código e desenvolvimento paralelo.

3

Trabalhos relacionados

Este capítulo discutirá os trabalhos relacionados encontrados na revisão sistemática, visando contextualizar e esclarecer os pontos negativos e positivos encontrados.

Na pesquisa efetuada no Google Scholar, foram considerados apenas os trabalhos escritos nos idiomas inglês e português. Ademais, para manter a consistência entre os artigos selecionados, foram definidos critérios de inclusão e exclusão, visando deixar os trabalhos mais coerentes com o tema escolhido.

Critérios de inclusão definidos:

- O artigo deve propor o uso de um juiz *online*;
- O artigo deve ter a disponibilidade gratuita;
- O artigo deve ser disponível no idioma inglês ou português;
- O artigo deve ser acessível e completo.

Critérios de exclusão definidos:

- O artigo apresenta um juiz *online* indisponível;
- O juiz *online* apresentado no artigo está disponível na *internet*.

3.1 Juiz *Online*

Zhou et al. (2018) explica que um juiz *online* é um software para avaliar programas com teste de caixa preta (teste que verifica a saída dos dados usando entradas de vários tipos). O primeiro juiz *online*, UVa, surgiu em 1995. Desde o final do século passado, tornou-se um sistema de avaliação utilizado frequentemente em competições de programação.

Chaves et al. (2014) comenta que alguns juízes *online* adicionam funcionalidades didáticas para auxiliar alunos e professores no processo de ensino-aprendizagem. Alguns dos recursos educacionais são: suporte ao gerenciamento de conteúdo e ferramenta de discussão.

Assim, utilizando os juízes *online* como métodos de avaliação de alta eficiência, justos e objetivos, é possível tornar os resultados da pontuação mais confiáveis e reduzir a carga de trabalho dos educadores.

Atualmente existem diversos juízes *online*, alguns deles são o CodeChef, UVa *Online* Judge, Beecrowd e o The Huxley.

3.1.1 UVa *Online* Judge

Simpson (2017) define que o UVa é um juiz *online* automatizado para problemas de programação hospedado pela Universidade de Valladolid, na Espanha. Ele contém mais de 4300 problemas e está crescendo a cada dia. As soluções para os problemas podem ser submetidas nas linguagens C, C ++, Java, Pascal e Python. Nele também é possível hospedar competições. Nesse ambiente de competições, o usuário tem um tempo limitado para resolver um pequeno conjunto de problemas.

3.1.2 CodeChef

CodeChef (2021) afirma que o CodeChef começou em 2009 como um programa educacional para a comunidade de programação. Hoje, CodeChef é uma das maiores e mais populares plataformas de programação competitiva global, preferida por estudantes e programadores profissionais. Com o apoio de sua enorme comunidade de solucionadores de problemas, a CodeChef promove o aprendizado contínuo e competições de programação amigáveis para seus usuários. Atualmente a CodeChef contém suporte para mais de 55 linguagens de programação.

CodeChef (2021) ainda acrescenta que, inicialmente, o CodeChef foi criado como uma plataforma para ajudar os programadores a se destacarem no mundo dos algoritmos, da programação de computadores e das competições de programação. O CodeChef acabou crescendo e ganhando uma nova forma de oferecer visibilidade aos usuários, todos os meses são realizadas competições organizadas pela CodeChef, nas quais são oferecidos prêmios e brindes como recompensas para os vencedores. Além disso, a plataforma também está aberta a toda a comunidade de programação, especialmente instituições acadêmicas e grupos de alunos que podem, também, sediar suas próprias competições. Um diferencial do CodeChef são as parcerias. A CodeChef proporciona parceria com escolas, faculdades e grupos de estudantes para criar encontros, sessões de orientação e *workshops* de programação locais da CodeChef.

3.1.3 Beecrowd

Segundo Beecrowd (2022), o Beecrowd é um juiz *online* focado nas necessidades dos professores e dos alunos dos cursos de graduação de computação. Anteriormente nomeado URI Online Judge, em 24 de outubro de 2021, entrou em uma nova fase sendo renomeado para Beecrowd.

De acordo com BEZ (2014), a construção do URI Online Judge foi completamente centrada nas necessidades dos professores e, principalmente, nas necessidades dos alunos. Esta união guiou o desenvolvimento do projeto e permitiu definir os objetivos centrais, entre eles:

- Oferecer um ambiente agradável, didaticamente organizado e com acesso 24 horas que corrigisse automaticamente todos os programas a ele submetidos em tempo real;
- Oferecer uma ferramenta que corrigisse as soluções dos alunos em tempo real, indicando, visualmente, possíveis erros;
- Disponibilizar o ambiente em português e inglês;
- Disponibilizar recursos para que professores possam criar disciplinas com alunos, listas de exercícios e acompanhar a sua evolução;
- Disponibilizar problemas categorizados por assunto e nível de dificuldade, evitando, assim, que estudantes que estão iniciando se frustrem ao tentar resolver problemas que exigem mais conhecimento, prática e técnicas;
- Oferecer um ambiente onde os estudantes possam interagir trocando experiências relacionadas às técnicas usadas para solucionar problemas;
- Possibilitar aos iniciantes em programação obter o auxílio de estudantes mais experientes;

3.1.4 The Huxley

Segundo Paes et al. (2013), o The Huxley é um juiz *online* que permite aos usuários enviar código-fonte em várias linguagens de programação. Em particular, o The Huxley tem um cunho educacional onde os educadores que usam esta plataforma têm mais ferramentas de análise para o desempenho dos alunos, incluindo o número de problemas resolvidos, a porcentagem de acertos/erros, as categorias de problemas com mais erros, detecção de plágio e os erros específicos de cada aluno. Com isso, professores começam a resolver os problemas especiais de aprendizagem dos alunos de uma maneira mais eficaz. Todo o trabalho de definição de exercícios e provas, correção dos mesmos, verificação de plágio, definição de notas e publicação dos resultados pode ser feito automaticamente através do The Huxley. Professores e alunos têm visões distintas, ambas baseadas no conceito de *dashboard* que permite uma visão global do *status* dos exercícios, avaliações e conteúdos da página inicial.

Paes et al. (2013) afirma que outra ferramenta disponível para educadores é a criação de avaliações. Na criação é definido um prazo válido para a submissão das respostas. Com isso, ela só fica disponível para os alunos na hora da prova. Após o tempo da prova ter expirado, a avaliação fica indisponível no The Huxley. Do ponto de vista dos alunos, há alguns mecanismos de incentivo que fazem com que os mesmos tenham motivação para acessar o sistema diariamente. Um deles é o Top Coders, que lista os alunos com maior quantidade de exercícios resolvidos.

4

Desenvolvimento do aplicativo The Huxley Mobile

Neste capítulo é apresentado o desenvolvimento do aplicativo The Huxley Mobile, comparando suas funcionalidades relacionando com o aplicativo Moodley. Ademais, os requisitos de software do aplicativo são apresentados.

4.1 Descrição da Proposta

Esse trabalho visa apresentar a evolução do aplicativo Moodley. Em sua nova versão, o aplicativo The Huxley Mobile (TH Mobile) foi totalmente reescrito utilizando o *Framework* Flutter e focando somente nas funcionalidades existentes no The Huxley. Sua *interface* foi atualizada e o escopo de funcionalidades incrementado quando comparado com o Moodley. O novo aplicativo inicialmente está disponível apenas no Google Play. A qualquer momento poderá ser disponibilizado na Apple Store, bastando efetuar o pagamento da assinatura anual. Dentre suas novas funcionalidades, o The Huxley Mobile permite ao usuário baixar tarefas, permitindo assim o uso de algumas funcionalidades mesmo sem conexão com a *internet*.

4.2 Requisitos Funcionais

Pressman (2009) define que requisitos funcionais (RF) são funcionalidades que determinam o comportamento de um sistema de software. Esse comportamento descreve o que deve ser feito, como deve ser feito e onde deve ser feito. Tais requisitos especificados devem ser completos, consistentes e claros. Os requisitos funcionais descrevem como cada ator pode se comportar ao utilizar o software.

Assim, o quadro 1 apresenta os requisitos funcionais identificados na aplicação. Com isso, os requisitos sinalizados com asterisco (*) são os novos requisitos adicionados aos demais já existentes no aplicativo Moodley.

Quadro 1 – Requisitos funcionais.

ID	Descrição	Classificação
RF01	Efetuar *login* com usuário e senha	Essencial
RF02	Efetuar logout	Essencial
RF03*	Efetuar cadastro de usuário	Essencial
RF04*	Recuperar senha com e-mail	Essencial
RF05	Buscar problemas	Essencial
RF06	Filtrar problemas por palavras-chave	Desejável
RF07	Filtrar problemas por dificuldade	Desejável
RF08	Filtrar problemas por tópicos	Desejável
RF09	Filtrar problemas por problema não resolvido	Desejável
RF10	Buscar tarefas	Essencial
RF11	Filtrar tarefas por turma	Desejável
RF12	Buscar turmas	Essencial
RF13	Adicionar nova turma	Essencial
RF14	Ordenar turmas por data de atualização mais recente	Desejável
RF15	Ordenar turmas por nome em ordem alfabética	Desejável
RF16	Visualizar problemas de uma tarefa	Essencial
RF17	Filtrar problemas de uma tarefa por *status* de andamento	Essencial
RF18	Visualizar descrição de um problema	Essencial
RF19	Submeter resposta de um problema	Essencial
RF20	Visualizar submissões	Desejável
RF21	Filtrar submissões por *status* de avaliação	Desejável
RF22*	Efetuar download de uma submissão	Desejável
RF23	Visualizar Top Coders	Essencial
RF24	Filtrar Top Coders por turma	Essencial
RF25*	Visualizar tarefas em andamento	Essencial
RF26*	Visualizar problemas pendentes	Essencial
RF27*	Visualizar tarefas *offline*	Desejável
RF28*	Visualizar problemas *offline*	Desejável
RF29*	Atualizar foto de perfil	Desejável

Fonte: Autor.

Quadro 2 – Definições de termos.

Termo	Definição
Problema	Conjunto de precondições estabelecidas, onde para cada entrada esperada existe uma saída conhecida.
Palavra-chave	Conjunto de caracteres ASCII.
Tópico	Característica definida que pode ser associada a um problema.
Tarefa	Conjunto de Problemas.
Turma	Um conjunto de usuários. Uma turma pode conter tarefas associadas.
Top Coders	Conjunto de usuários ordenado pela maior pontuação de resolução de problemas.

Fonte: Autor.

4.3 Requisitos Não Funcionais

Os Requisitos não funcionais (RNF) são requisitos que afetam as restrições do sistema. O quadro 3 apresenta os RNF que foram identificados na aplicação.

Quadro 3 – Requisitos não funcionais.

ID	Tipo	Descrição
RNF01	Compatibilidade	O sistema deve possibilitar compatibilidade com dispositivos móveis *Android e iOS*.
RNF02	Interoperabilidade	O sistema deve utilizar a API do Juiz Online The Huxley
RNF03	Produto	Todas as ferramentas utilizadas no desenvolvimento da aplicação deverão ser gratuitas.

Fonte: Autor.

4.4 Arquitetura

O aplicativo utiliza a arquitetura MVC que serve de referência para a organização dos módulos no projeto. A figura 1 apresenta o diagrama de pacotes contendo todos os pacotes/módulos existentes no projeto. Assim, usando essa estrutura de módulos, temos uma maior segregação de responsabilidade na aplicação. Os *models* são compartilhados no projeto, isso permite que qualquer módulo possa utilizar os *models*. Com os *models* disponíveis, podemos utilizá-los a partir de outros módulos. Cada módulo contém pelo menos uma *view* e um *controller*. A figura 2 mostra que também temos algumas outras pastas auxiliares opcionais que fazem parte desses módulos, como as pastas de *interfaces* sendo os contratos dos repositórios, *repositories* que permitem o isolamento do acesso aos dados e *widgets* que são os componentes da *view*.

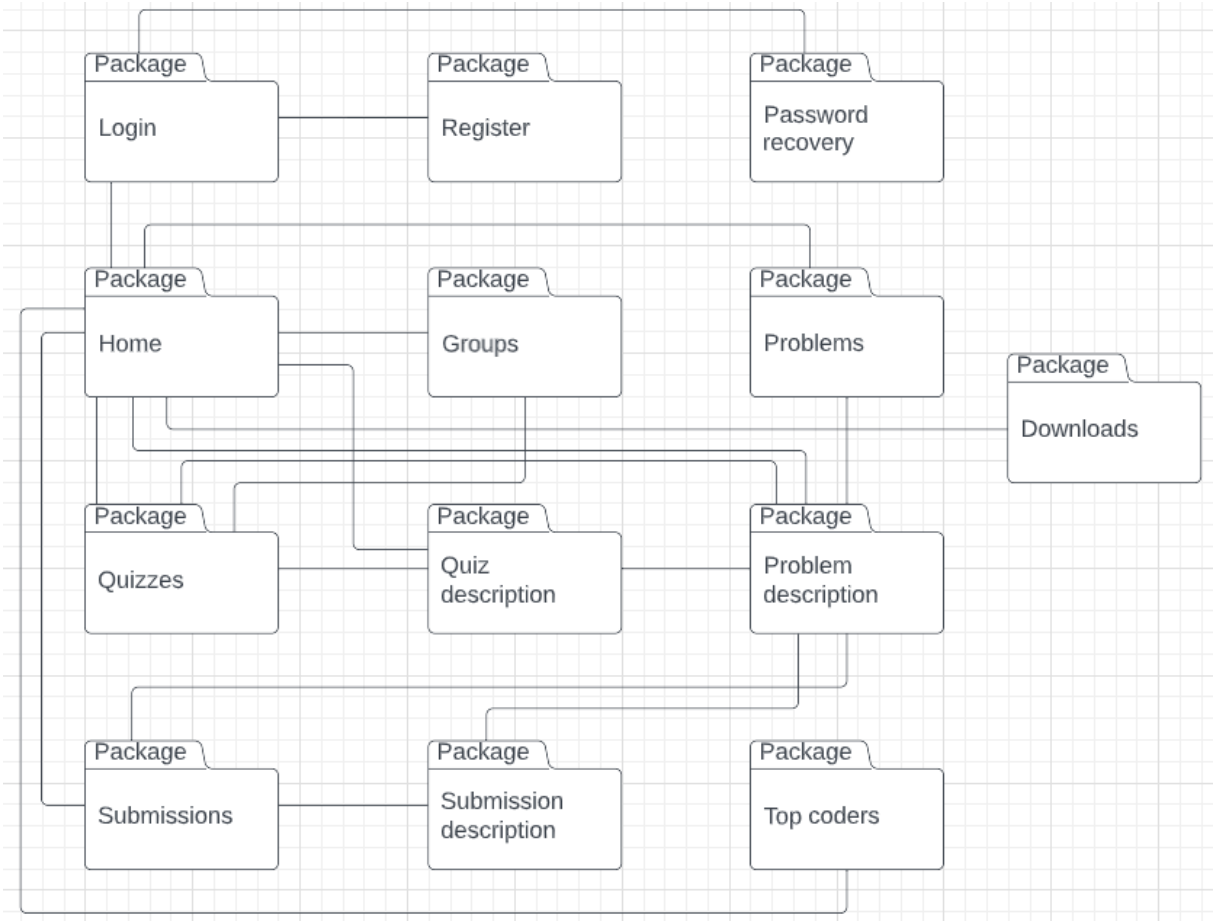

Figura 1 – Módulos do TH Mobile.

Fonte: Autor.

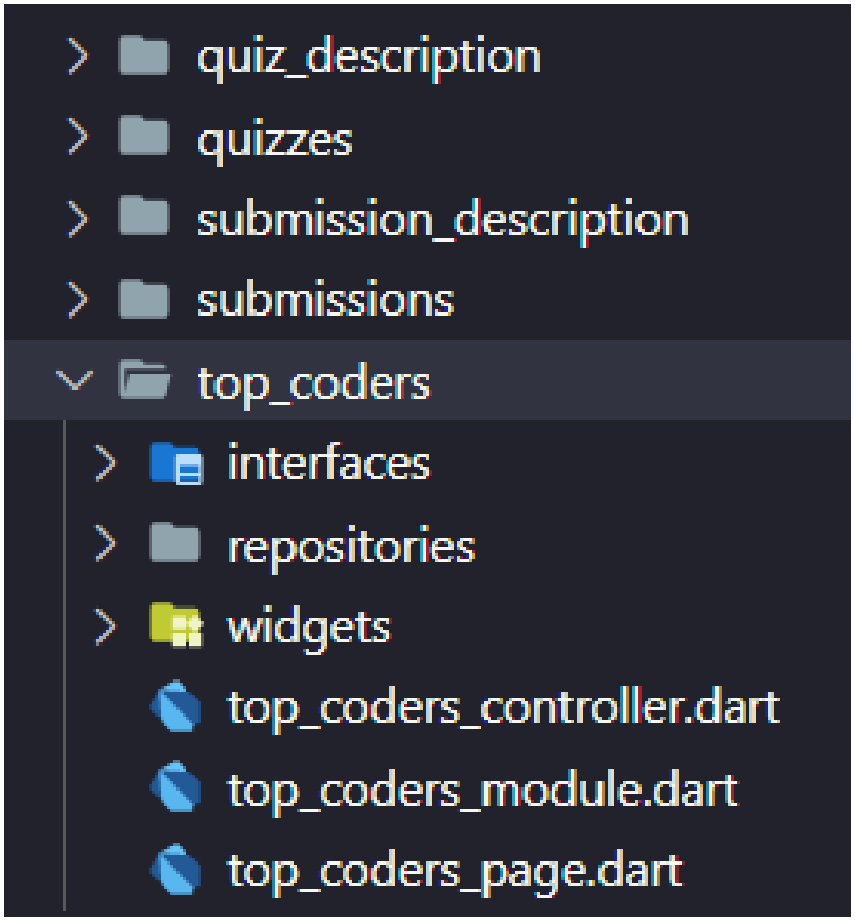

Figura 2 – Detalhes do módulo top coders do TH Mobile.

Fonte: Autor.

4.5 Estratégia de Integração com o The Huxley

O aplicativo The Huxley Mobile possui integração com o juiz *online* The Huxley, realizada através de uma API REST. O quadro 4 apresenta todos endereços *web* utilizados desta API.

Toda integração realizada seguiu o mesmo padrão de construção. Nela, um *endpoint* é utilizado para efetuar uma requisição para a API REST do The Huxley, por um cliente HTTP. Foi utilizado o pacote Dio, um cliente HTTP para Dart. Essa requisição pode conter diversas informações, sendo uma delas obrigatória, o verbo da requisição (GET, POST DELETE, PUT ou PATCH). Essa requisição, além de enviar, também pode receber informações, geralmente o envio e/ou obtenção de dados são no formato JSON, formato suportado no Dart com serialização e desserialização. Ademais, a figura 3 apresenta a comunicação descrita.

Quadro 4 – *Endpoints* utilizados nos requisitos.

Endpoint	Requisito
thehuxley.com/api/login	RF01
thehuxley.com/api/v1/topcoders	RF23 e RF24
thehuxley.com/api/v1/user	RF01
thehuxley.com/api/v1/user/avatar	RF29*
thehuxley.com/api/v1/user/quizzes	RF10 e RF11
thehuxley.com/api/v1/user/problems	RF19
thehuxley.com/api/v1/user/submissions	RF21
thehuxley.com/api/v1/user/groups	RF12, RF14 e RF15
thehuxley.com/api/v1/users	RF03*
thehuxley.com/api/v1/users/validate	RF03*
thehuxley.com/api/v1/users/recoveryPassword	RF04*
thehuxley.com/api/v1/groups/join	RF13
thehuxley.com/api/v1/groups/key	RF13
thehuxley.com/api/v1/submissions	RF20
thehuxley.com/api/v1/languages	RF19
thehuxley.com/api/v1/quizzes	RF16
thehuxley.com/api/v1/problems	RF05, RF06, RF07, RF08, RF09 e RF18
thehuxley.com/api/v1/topics	RF08

Fonte: Autor.

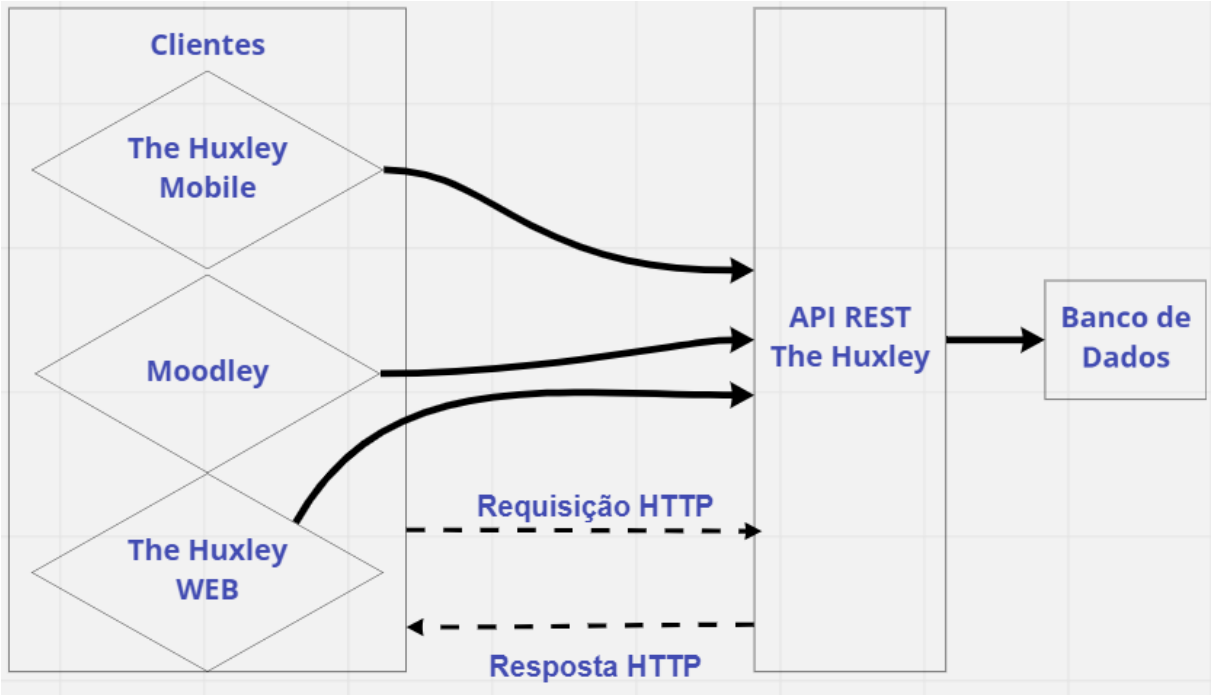

Figura 3 – Fluxo de integração no TH Mobile.

Fonte: Autor.

Na figura 4 é apresentado um exemplo de integração utilizando o endereço thehuxley.com/api/v1/user. No exemplo, é feita uma requisição com o verbo GET no endereço escolhido, e como resultado é obtido um JSON com todas as informações do usuário, como mostra a figura 5.

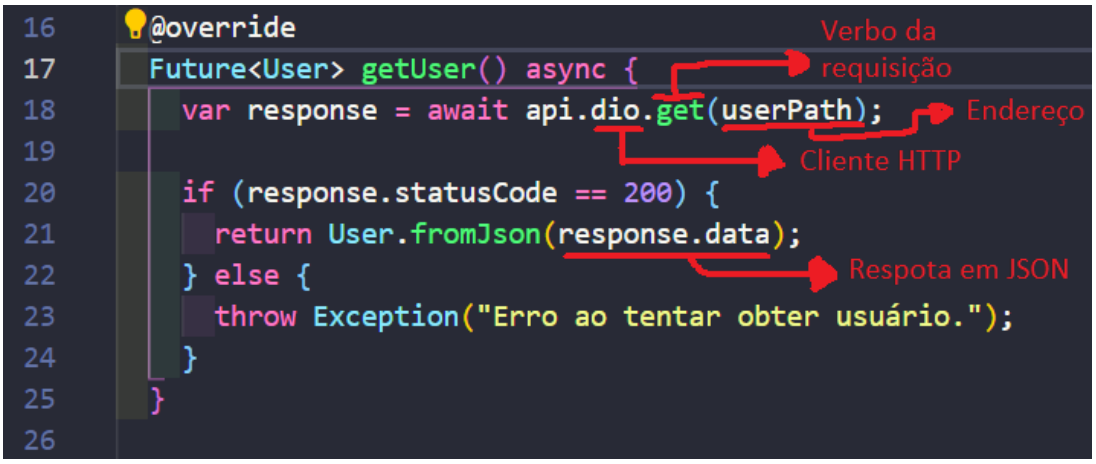

Figura 4 – Requisição de obtenção de informações do usuário no TH Mobile.

Fonte: Autor.

Figura 5 – Resposta de requisição de obtenção de informações do usuário no TH Mobile.

Fonte: Autor.

4.6 Exemplos de Requisitos Desenvolvidos

A seguir, tem-se a apresentação de algumas implementações do The Huxley Mobile. Ademais, também é exibida sua versão anterior, o Moodley, para servir de parâmetro de

comparação sobre a evolução funcional, visual e de usabilidade.

4.6.1 Visualizar Tarefas em Andamento

Os novos requisitos de visualização de tarefas e problemas adicionados deram uma nova forma do usuário acompanhar o andamento das tarefas pendentes. No acompanhamento do andamento de cada tarefa é apresentado o título, o resumo, o prazo restante para a tarefa acabar e a porcentagem de progresso. Na figura 6 temos a tela inicial do Moodley e na figura 7 temos a tela inicial do The Huxley Mobile.

Figura 6 – Tela inicial no Moodley.

Figura 7 – Tela inicial no TH Mobile.

Fonte: Autor.

Na figura 8 é ilustrado o código-fonte para exibição das tarefas em aberto. Já na figura 9, é ilustrado o código-fonte responsável por estabelecer a conexão com a API do The Huxley através do método "_quizRepository", onde é possível obter as tarefas filtradas pela data inicial e final através do método "openedQuizzes" disponível na classe "Quiz".

```
Container(
  height: 210,
  width: 256,
  child: StreamBuilder<List<Quiz>>(
      stream: store.quizzes,
      initialData: [],
      builder: (context, snapshot) {
        var quizzes = snapshot.data!;
        return QuizzesListViewWidget(
            quizzes: quizzes); // QuizzesListViewWidget
      }), // StreamBuilder
), // Container
```

Figura 8 – Exibição das tarefas presente no arquivo “home_page.dart”.

```
Future<void> _getQuizzes() async {
  var quizzes = await _quizRepository.getQuizzes();
  var openedQuizzes = Quiz.openedQuizzes(quizzes);

  _quizzesController.add(openedQuizzes);
}
```

Figura 9 – Obtenção das tarefas presente no arquivo “home_controller.dart”.

Fonte: Autor.

Na Figura 10 é ilustrado o código-fonte para exibir os problemas em aberto. Nesse trecho do código, faz-se uso de um "StreamBuilder", o qual é responsável por receber os eventos para poderem ser exibidos conforme seu *status*. Na Figura 11, é apresentada a codificação capaz de obter os problemas das tarefas previamente obtidas. Ademais, os problemas são filtrados por *status* de "Não resolvido" e, assim, é enviado um evento para o "StreamBuilder" com os problemas disponíveis.

```
StreamBuilder<List<Quiz>>(
    stream: store.quizzes,
    initialData: [],
    builder: (context, snapshot) {
      var quizzes = snapshot.data!;
      return ProblemsViewWidget(
          quizzes: quizzes); // ProblemsViewWidget
    }), // StreamBuilder
```

Figura 10 – Exibição dos problemas presente no arquivo "home_page.dart".

```
Future<void> _getProblems() async {
  var quizzes = _quizzesController.value;
  var user = _userController.value;

  for (int i = 0; i < quizzes.length; i++) {
    var quiz = quizzes[i];

    var problems = await _quizRepository.getQuizProblems(user.id, quiz.id);

    problems = Problem.unsolvedProblems(problems);

    quizzes[i].problems = problems;
  }
  _quizzesController.add(quizzes);
}
```

Figura 11 – Obtenção dos problemas presente no arquivo "home_controller.dart".

Fonte: Autor.

4.6.2 Visualizar Problemas de uma Tarefa

Ao clicar em uma tarefa, o usuário é direcionado para a tela de visualização. Nela o usuário encontra o título da tarefa, descrição, prazo de entrega restante, data de início, data de término e os problemas associados para a tarefa, como podemos observar na figura 13. Ainda na tela de visualização de tarefa, foi inserido um filtro de *status* dos problemas (pendentes, iniciados, finalizados). Já no Moodley temos algumas informações semelhantes, mas não temos o filtro de *status* dos problemas, como mostra a figura 12.

Figura 12 – Tela de tarefa no Moodley.

Figura 13 – Tela de tarefa no TH Mobile.

Fonte: Autor.

Nas figuras14 ,15 e 16 temos o código-fonte da exibição dos detalhes da tarefa, os filtros e os problemas. O "CardQuizInfoWidget" contém as informações relacionadas à tarefa e as ações de efetuar e excluir *download*. Mais embaixo, temos o código-fonte da exibição dos filtros de "Não resolvidos", "Tentados" e "Resolvidos", nele são utilizados "ToggleButtons" com uma lista

de valores booleanos para identificar quais filtros estão ativos no momento. Ademais, temos a exibição da lista dos problemas com o “TileProblemsListWidget”.

```
LayoutWidget(
  child: CardQuizInfoWidget(
    quiz: store.quiz,
    downloadQuiz: offline ? null : store.downloadQuiz,
    deleteQuiz: offline ? null : store.deleteQuiz,
    quizExists: offline ? null : store.quizExists,
    offline: offline,
    navigation: false,
  ), // CardQuizInfoWidget
), // LayoutWidget
```

Figura 14 – Exibição de detalhes da tarefa presente no arquivo “quiz_description_page.dart”.

```
StreamBuilder<List<bool>>(
    stream: store.selected,
    initialData: [false, false, false],
    builder: (context, snapshot) {
      var isSelected = snapshot.data!;
      return Center(
          child: (ToggleButtonsWidget(
        isSelected: isSelected,
        onPressed: (int index) {
          isSelected[index] = !isSelected[index];
          store.changeSelected(isSelected);
        },
      ))); // ToggleButtonsWidget // Center
    }), // StreamBuilder
```

Figura 15 – Exibição de filtros da tarefa, arquivo “quiz_description_page.dart”.

```
StreamBuilder<List<Problem>>(
    stream: store.problems,
    initialData: [],
    builder: (context, snapshot) {
      var problems = snapshot.data!;
      return TileProblemsListViewWidget(problems: problems, offline: offline);
    }), // StreamBuilder
```

Fonte:

Figura 16 – Exibição de problemas da tarefa, arquivo “quiz_description_page.dart”.

Autor.

Já nas figuras 17 e 18 temos o código-fonte da regra de negócio onde os problemas podem ser filtrados e os detalhes e problemas da tarefa são obtidos. Na figura 17 temos o método "changeSelected" responsável por filtrar os problemas baseados nos filtros, nele temos as verificações de quais *status* estão ativos, com isso é verificado se os problemas contêm esse *status* e, finalmente, retornando eles para tela. Ademais, na figura 18 temos o código-fonte responsável por acessar a API do The Huxley e com o uso do repositório "_quizRepository" as informações da tarefa e os problemas desse tarefa são obtidas com os métodos "getQuiz" e "getQuizProblems".

```
changeSelected(List<bool> values) {
  bool pending = values[0];
  bool started = values[1];
  bool solved = values[2];
  List<Problem> problems = _problems.where((element) {
    if (!pending && !started && !solved) return true;

    if (pending && element.status == ProblemStatus.pending) {
      return true;
    }
    if (started && element.status == ProblemStatus.started) {
      return true;
    }
    if (solved && element.status == ProblemStatus.solved) {
      return true;
    }
    return false;
  }).toList();
```

Figura 17 – Alteração de filtros da tarefa, arquivo "quiz_description_controller.dart".

```
Future<void> _getQuiz(int userId, int quizId) async {
  var quiz = await _quizRepository.getQuiz(quizId);
  this.quiz = quiz;
}

Future<void> _getProblems(int userId, int quizId) async {
  var problems = await _quizRepository.getQuizProblems(userId, quizId);
  _problems = problems;
  _problemsController.add(problems);
}
```

Figura 18 – Obtenção de detalhes e problemas, arquivo "quiz_description_controller.dart".

Fonte: Autor.

5 Conclusão

Este trabalho foi iniciado com uma revisão de mercado onde não foram encontrados artigos e aplicativos para serem discutidos, além daqueles que abordam o Moodley[1]. Assim, foi inicialmente apresentado um juiz *online* e suas particularidades em cada um dos sistemas de software já desenvolvidos.

O objetivo principal deste trabalho foi apresentar o aplicativo The Huxley Mobile, desenvolvido no *framework* Flutter com abordagem multiplataforma, utilizando-se como *backend* da API do The Huxley.

O The Huxley Mobile conseguiu alcançar seus objetivos principais: oferecer uma melhor experiência de usabilidade para os usuários com uma nova proposta de *design*, mais fluidez durante o uso, um melhor detalhamento de resultados das submissões e o uso *offline* de algumas funcionalidades. A nova forma de escolha de arquivos para submeter código-fonte, por meio da galeria, possibilita uma boa sinergia com aplicativos de edição de código-fonte com mais recursos, assim podendo obter a melhor experiência desde a escrita do código-fonte.

Por fim, foi disponibilizada uma versão com todas funcionalidades desenvolvidas até o momento. A versão Android foi publicada na Google Play[2] e a versão iOS a qualquer momento poderá ser disponibilizada na Apple Store, bastando efetuar o pagamento da assinatura anual.

5.1 Trabalhos Futuros

Como trabalhos futuros, busca-se a continuidade do desenvolvimento de um editor de código-fonte, a funcionalidade de executar o código com entradas de exemplo, sem submeter o código-fonte, com distribuição na App Store e validação da aplicação com os usuários.

[1] Moodley: <https://play.google.com/store/apps/details?id=br.ufs.avamovel.moodley>
[2] The Huxley Mobile: <https://play.google.com/store/apps/details?id=br.ufs.dcomp.thehuxleymobile>

Referências

BARBOSA, G.; OLIVEIRA, E.; D'CARLO, D. Usabilidade em aplicativos móveis educacionais: Um conjunto de heurísticas para avaliação. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 777. Citado na página 7.

BEECROWD. *Beecrowd*. 2022. Disponível em: <https://www.beecrowd.com.br/>. Citado na página 15.

BEZ, N. A. T. J. L. Uri online judge e a internacionalizaÇÃo da universidade. 2014. Disponível em: <http://www2.reitoria.uri.br/~vivencias/Numero_018/artigos/pdf/Artigo_21.pdf>. Citado na página 15.

CHAVES, J. O. M. et al. Mojo: uma ferramenta para integrar juízes online ao moodle no apoio ao ensino e aprendizagem de programação. *HOLOS*, Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, v. 5, p. 237–251, 2014. Citado na página 14.

CHEBBI, A. *Desenvolvimento de app: escolhendo a linguagem de programação*. 2019. Disponível em: <https://www.tecmundo.com.br/software/204629-desenvolvimento-app-escolhendo-linguagem-programacao.htm>. Citado na página 10.

CODECHEF. *CodeChef*. 2021. Disponível em: <https://www.codechef.com/aboutus?itm_medium=navmenu&itm_campaign=aboutus>. Citado na página 14.

FLUTTER. *Hot reload*. 2022. Disponível em: <https://docs.flutter.dev/development/tools/hot-reload>. Citado na página 11.

GUEDES, M. *O que é MVC?* 2022. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-mvc>. Citado na página 12.

JESUS, G. S. de. *Uma abordagem para auxiliar a correção de erros de programadores iniciantes*. 157 p. Tese (Doutorado), 2018. Citado na página 8.

MACORATTI, J. C. *Flutter - Exibindo SnackBars*. 2022. Disponível em: <https://www.macoratti.net/19/06/flut_widgt1.htm>. Citado na página 11.

NUNES, V. G. C. Avaliação de abordagens e definição de diretrizes para o desenvolvimento de aplicativos mobile: um relato de experiência utilizando aabordagem não nativa de desenvolvimento. Universidade Federal Rural do Semi-Árido, 2021. Citado na página 10.

PAES, R. de B. et al. Ferramenta para a avaliação de aprendizado de alunos em programação de computadores. In: *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*. [S.l.: s.n.], 2013. v. 2, n. 1. Citado 2 vezes nas páginas 15 e 16.

PERRY, G. *Flutter + MVC at Last!* 2018. Disponível em: <https://medium.com/follow-flutter/flutter-mvc-at-last-275a0dc1e730>. Citado na página 12.

PETIT, J.; GIMéNEZ, O.; ROURA, S. Jutge.org: An educational programming judge. In: *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2012. (SIGCSE '12), p. 445–450. ISBN 9781450310987. Disponível em: <https://doi-org.ez20.periodicos.capes.gov.br/10.1145/2157136.2157267>. Citado na página 7.

PRESSMAN, R. *Engenharia de Software - 7.ed.* McGraw Hill Brasil, 2009. ISBN 9788580550443. Disponível em: <https://books.google.com.br/books?id=y0rH9wuXe68C>. Citado na página 17.

SANTOS, T. *Flutter vs React Native – Qual é o melhor?* 2022. Disponível em: <https://blog.4linux.com.br/flutter-vs-react-native-qual-e-o-melhor/>. Citado na página 10.

SIMPSON, R. The appropriateness of competitive programming and the uva online judge in the classroom: Conference tutorial. *J. Comput. Sci. Coll.*, Consortium for Computing Sciences in Colleges, Evansville, IN, USA, v. 32, n. 4, p. 49, apr 2017. ISSN 1937-4771. Citado na página 14.

WU, W. React native vs flutter, cross-platforms mobile application frameworks. Metropolia Ammattikorkeakoulu, 2018. Citado na página 11.

ZHAO, W. X. et al. Automatically learning topics and difficulty levels of problems in online judge systems. *ACM Transactions on Information Systems (TOIS)*, ACM New York, NY, USA, v. 36, n. 3, p. 1–33, 2018. Citado na página 7.

ZHOU, W. et al. The framework of a new online judge system for programming education. In: *Proceedings of ACM Turing Celebration Conference-China*. [S.l.: s.n.], 2018. p. 9–14. Citado na página 13.