UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

GABRIEL HENRIQUE PAPACIDRO DA SILVA

APLICATIVO MOBILE EDUCACIONAL PARA AVALIAÇÃO REMOTA E PRESENCIAL

Sorocaba
2024

GABRIEL HENRIQUE PAPACIDRO DA SILVA

APLICATIVO MOBILE EDUCACIONAL PARA AVALIAÇÃO REMOTA E PRESENCIAL

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Ivando Severino Diniz

Sorocaba

2024

S586a

Silva, Gabriel Henrique Papacidro da
Aplicativo mobile educacional para avaliação remota e presencial / Gabriel Henrique Papacidro da Silva. -- Sorocaba, 2024
60 p. : tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba
Orientador: Ivando Severino Diniz

1. Delphi (Linguagem de programação de computador). 2. Aplicativos móveis. 3. Software educacional. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

APLICATIVO MOBILE EDUCACIONAL PARA AVALIAÇÃO REMOTA E PRESENCIAL

GABRIEL HENRIQUE PAPACIDRO DA SILVA

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÂO**

Prof. Dr. Everson Martins
Coordenador

BANCA EXAMINADORA*:*

Prof. Dr. IVANDO SEVERINO DINIZ
UNESP- Campus de Sorocaba

Prof. Dr. EVERSON MARTINS
UNESP – Campus de Sorocaba

Prof. Eng. Mestrando DANILO RUY GOMES
UNESP – Campus de Sorocaba

Junho de 2024

AGRADECIMENTOS

Agradeço a minha família pelo suporte constante durante todas as fases da graduação. Agradeço aos professores, técnicos e funcionários da Unesp Sorocaba por todos os ensinamentos, apoio e manutenção de um ambiente propício para o aprendizado. Agradeço ao Prof. Dr. Ivando Severino Diniz pelos ensinamentos e ajuda durante o desenvolvimento do projeto. Agradeço também a todos os amigos e colegas que estiveram ao meu lado durante essa jornada, dividindo os bons e maus momentos durante todo o curso.

RESUMO

O projeto apresentado é baseado no desenvolvimento de aplicações utilizando de ferramentas populares no mercado internacional como Delphi, SQL e MySQL para a criação de sistemas multiplataformas, utilizando de método de desenvolvimento rápido baseado em programação de alto nível baseado em eventos e designs visuais. O objetivo do projeto é a criação de um aplicativo educacional para o auxílio de avaliação em sala de aula, através da criação de questionários de múltipla escolha, com correção automática. Devido a rápida evolução tecnológica na área de softwares, e o grande impacto causado pela pandemia de COVID-19, a utilização de ferramentas digitais nas instituições de ensino teve um grande aumento, criando a necessidade de novos profissionais e sistemas para atender as diversas áreas do ensino. Para o desenvolvimento, foi utilizando o Delphi 11 “Community Edition” em conjunto com o RAD Studio para a criação da aplicação principal, auxiliado por códigos PHP e SQL para realizar a comunicação e controle das informações no banco de dados MySQL, que atua como servidor central para as diversas requisições e armazenamentos presentes no fluxo. O sistema é baseado no envio de questionários criados por usuários, e a busca destes questionários seguidos pelo envio das respostas, para criar um fluxo de avaliação rápida e eficiente que pode ser utilizado em sala de aula, auxiliando o docente e os alunos. Ao final do desenvolvimento, o sistema atendeu os requisitos propostos, porém sua implementação em escala maior causaria um aumento de custos, além da necessidade do desenvolvimento de funcionalidades extras para suprir a maior demanda.

Palavras-chave: Delphi; RAD Studio; aplicativo; desenvolvimento rápido; MySQL; SQL; sistema educacional; software; PHP.

ABSTRACT

The Project is based on the development of applications using popular tools found in the international market, like Delphi, SQL, and MySQL to create a multi-platform system, with fast development methods, based on high level, event based programming and visual design. The main objective is the creation of an educational application, to assist the assessment in the classroom, by creating multiple choice quizzes with automatic revision. Due to the quick technological evolution in the software area, and the great impact caused by the COVID-19 pandemic, the use of digital tools in the educational institutions had a great increase, creating the need for new professionals and systems to attend the diverse areas of instruction. For the development, Delphi 11 “Community Edition” was used with RAD Studio to create the main application, assisted by PHP and SQL codes to communicate and control the information in the MySQL database that acts as the central server to the requests e storage present in the program flow. The system is based on the creation of quizzes made by the users, and search of those quizzes, followed by the send of the answers, to create a quick and effective assessment flux to be used in the classroom, assisting the teacher and the students. By the end of the development, the system complied with requirements, by its implementation in a bigger scale would result in a raised cost, beside the need of the development of new functionalities to attend the higher demands.

Key words: Delphi; RAD Studio; application; fast development; MySQL; SQL; educational system; software; PHP.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O avanço tecnológico e a educação, sempre andaram lado a lado para melhorar os métodos de ensino, buscando aumentar o engajamento dos estudantes e docentes dentro e fora da sala de aula. Essa combinação sofreu um grande crescimento devido a pandemia de COVID-19, que obrigou as instituições de ensino a adaptarem os métodos aplicados em sala de aula para versões digitais, devido ao isolamento causado pela pandemia.

Houve um grande um crescimento de 90% nos downloads de aplicativos educacionais ente os três últimos meses de 2019 e os três primeiros meses de 2020 (SCHAFFHAUSER, 2020), e manteve um aumento de 17% entre 2020 e 2021, durante o período de “lockdown” e continuaram a crescer ano após ano, mesmo com o fim da pandemia e o retorno das salas de aulas presenciais (SHRESTHA, 2023).

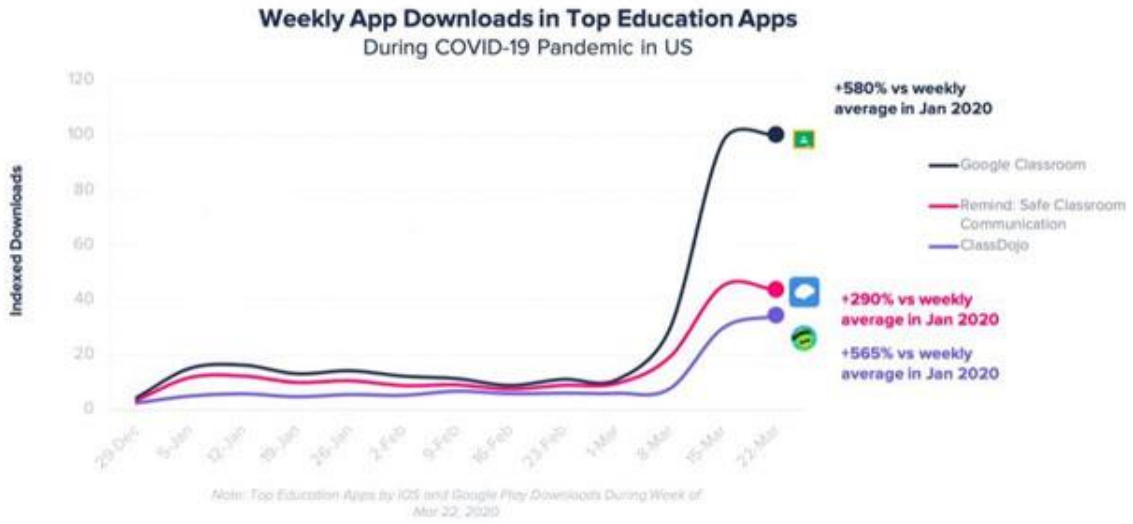
Figura 1– Crescimento das instalações de aplicativos educacionais. Jan2021 - Ago 2023

Fonte: Shrestha (2023).

O aumento na busca por aplicativos educacionais causou um aumento no número de startups de tecnologia focadas em educação, que apresentaram recordes de lucro e audiência durante 2021 (CNN, 2021). Apesar da variedade de opções presentes no mercado, as que apresentaram o maior ganho foram as plataformas grátis, controladas por grandes empresas de

tecnologia como o “Google Classroom”, que se mostrou a principal escolha entre as instituições de ensino, tanto do setor público como do setor particular.

Figura 2 – Downloads semanais de aplicativos educacionais durante a pandemia de Covid-19 nos EUA.



Fonte: App Annie ([2020?]) apud Schaffhauser (2020).

Os dados apresentados mostram um crescimento no interesse do público pelas plataformas de ensino digitais, e considerando o retorno das aulas presenciais, o projeto em questão busca mesclar os ambientes, através de um aplicativo que possa ser utilizado em sala de aula ou como lição externa, para que os docentes possam compartilhar questionários de forma dinâmica com seus alunos, com retorno rápido sobre o desempenho e nota obtido, como forma de avaliar e testar os conhecimentos dos alunos ao longo do curso.

O projeto foi desenvolvido utilizando Delphi para a criação de um sistema intuitivo e de fácil utilização, desenvolvido através de programação baseada em eventos e design visual, para obter as funcionalidades de auxílio ao docente de forma simples e direta.

2 OBJETIVOS

O projeto tem como objetivo o desenvolvimento de um aplicativo para auxiliar o docente na avaliação dos alunos através de uma plataforma de compartilhamento e correção de exercícios. O processo de criação do sistema irá auxiliar no aprendizado sobre programação em Delphi, criação e manuseio de banco de dados e comunicação via PHP.

Para atender o objetivo do projeto, o sistema irá contar com as funcionalidades:

- Criação de lista de Exercícios múltipla escolha;
- Definição das alternativas corretas;
- Valor de cada questão;
- Abrir e Fechar o questionário no momento desejado;
- Compartilhamento de perguntas através de código;
- Acesso do questionário pelos outros usuários, quando aberto;
- Responder o questionário e enviar as respostas;
- Corrigir e atribuir notas automaticamente as respostas enviadas;
- Autor pode verificar quem respondeu o questionário e suas notas;
- Usuário pode verificar suas notas dos questionários que respondeu;

O aumento da integração da tecnologia na vida cotidiana permite também o uso dela em sala de aula para automatizar o processo de avaliação dos alunos. Portanto, o projeto visa criar uma maneira fácil de integrar os aparelhos móveis ao ensino diário em sala de aula.

3 REVISÃO BIBLIOGRÁFICA

Neste capítulo, serão revisadas as referências bibliográficas utilizadas na concepção e desenvolvimento do projeto.

O desenvolvimento do projeto busca aumentar a interatividade em sala de aula, segundo (BOTTENTUIT, 2020):

> A promoção de experiências mais significativas em sala de aula é uma exigência nos dias atuais, uma vez que os alunos irão atuar em um mercado cada vez mais exigente e que necessita de profissionais polivalentes, com competências necessárias, entre elas: cultura digital, domínio dos recursos tecnológicos, facilidade em comunicação, colaboração, resolução de problemas, entre outros.

Em seu artigo, Andrade, Araújo e Silveira (2015) citam que a aprendizagem móvel *(M-learning)* é considerada uma das principais tendências de aplicações tecnológicas na educação. Buscando acompanhar essa tendência, o aplicativo busca facilitar ao professor a distribuição e correção de questionários, como maneira de avaliar e acompanhar o desempenho dos estudantes, dentro e fora de sala. Segundo Sonego e Behar (2015) às tecnologias da informação e comunicação, e o *M-learning* geram oportunidades de inovação nas ações dos docentes de todas as áreas, possibilitando práticas pedagógicas que utilizem e explorem estas tecnologias dentro e fora do âmbito escolar.

Para o desenvolvimento, escolheu-se o Delphi, a fim de uma programação voltada ao design simples e interativo, tendo em vista que o Delphi é uma IDE que possui um ambiente de desenvolvimento visual, no qual muitas vezes, o ambiente de desenvolvimento pode ter um papel mais importante que a própria linguagem de programação (KALMYKOV, [2016?]). A IDE Delphi combina o desenvolvimento visual, criando classes automaticamente para os objetos adicionados de forma dinâmica no sistema.

O aplicativo tende a melhorar o fluxo em sala de aula, e permitir ao docente um maior manejo dos seus alunos. Em Leite (2014, p. 60), o autor cita:

> [...] as tecnologias de uso educativo [...] se tem convertido em um suporte fundamental para a instrução, beneficiando um universo cada vez mais amplo de pessoas. Esta associação entre tecnologia e educação não somente gera melhoras de caráter quantitativo (possibilidade de ensinar a mais estudantes), mas principalmente de ordem qualitativa (os educandos encontram na Internet novos recursos e possibilidades de enriquecer seu processo de aprendizagem).

4 DESCRIÇÃO TEÓRICA

Este capítulo tem como intuito explicar as tecnologias utilizadas no projeto. As tecnologias citadas serão os conceitos centrais a serem utilizados no desenvolvimento do projeto.

A programação teve início em meados da década de 1930 com o surgimento dos primeiros elétricos. Em 1948, Konrad Zuse publicou sua criação, a linguagem de programação Plankalkül. Foi na década de 1950 em que as primeiras linguagens modernas surgiram, como OFRTRAN, COBOL e ALGOL(PCIEVITCH, 2021).

Nos anos 90, a internet começou a surgir e mudou o rumo da programação. Nesta época surgiram as linguagens Java e Javascript, relacionadas com páginas na rede, também surgiram o visual basic e o Object Pascal. Com o desenvolvimento destas tecnologias veio também o PHP, linguagem importante para auxiliar o desenvolvimento de aplicativos WEB (PCIEVITCH, 2021).

4.1 DELPHI 11 “COMMUNITY EDITION”

Delphi é um compilador, uma IDE e uma linguagem de programação, produzido inicialmente pela Borland Software Corporation em 1995 e atualmente é produzido pela Embarcadero.

Baseada inicialmente em *C++* e *Object Pascal*, é uma plataforma altamente utilizada para o desenvolvimento de aplicações devido a felicidade de aprendizagem e de uso, principalmente em aplicações multicamadas, cliente/servidor e aplicações com banco de dados, por sua grande compatibilidade com as principais ferramentas de banco de dados do mercado (EMBARCADERO, [201-?b].).

Delphi é um sistema de desenvolvimento de software em alto nível com um conjunto de ferramentas integradas para prover um desenvolvimento rápido de aplicativos. Os conceitos de programação baseada em eventos e design visual são os pilares da ferramenta e reduzem significativamente o tempo de desenvolvimento (KALMYKOV, [2016?]).

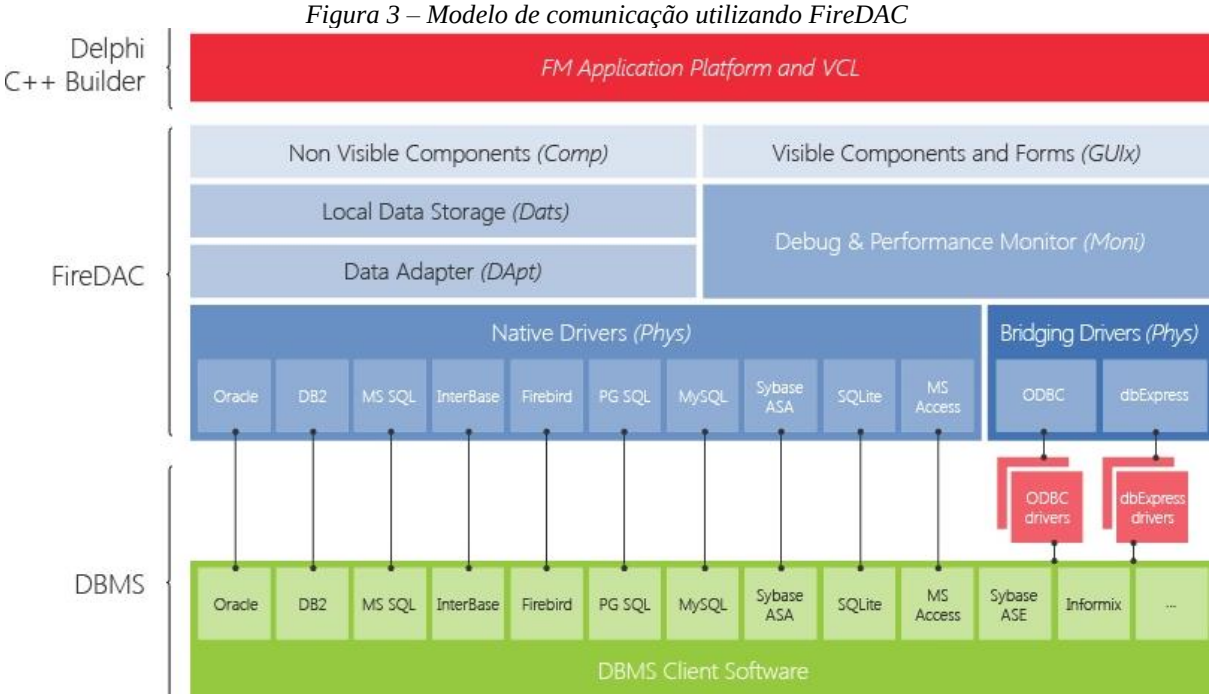
Hoje Delphi é utilizado em diversas aplicações de diversos segmentos tanto no lado cliente como no lado servidor em aplicações multicamadas e com compatibilidade para diversos bancos de dados. Uma de suas vantagens é a possibilidade de com um único código criar aplicativos para Android, IOS e Windows.

Para o desenvolvimento foi utilizada a versão “Delphi 11 Community Edition”

4.2 FIREDAC E FIREMONKEY

FireDAC é uma biblioteca de acesso de dados universal utilizada pelo Delphi, para o desenvolvimento de aplicações para múltiplos dispositivos. Sua arquitetura permite a conexão de alta velocidade com diferentes bancos de dados e servidores de “back-end”(EMBARCADERO, [201-?a].).

Figura 3 – Modelo de comunicação utilizando FireDAC



Fonte: Embarcadero (*[201-?a]).*

FireMonkey é uma ferramenta de suporte ao FireDAC que tem como intuito, facilitar o desenvolvimento e conexão dos processos para sistemas criados para aplicações móveis, permitindo uma conexão direta destas aplicações aos bancos e ferramentas suportadas, sem a necessidade de camadas extras de processamento.

4.3 SISTEMAS GERENCIADORES DE BANCO DE DADOS RELACIONAL

Sistemas gerenciadores de banco de dados são sistemas capazes de alterar, editar, remover e inserir dados em bancos de dados, com o intuito de organizar e manipular os dados armazenados no banco, de maneira a fornecer o controle para o usuário, além de garantir segurança e consistência das informações contidas no sistema.

O principal sistema utilizado atualmente são os SGBDs relacionais. Os sistemas relacionais têm os dados organizados em tabelas relacionadas entre si através de colunas “chaves” que indicam o valor único de cada linha da tabela, e chaves estrangeiras que relacionam a tabela a outra tabela através de suas chaves primárias.

4.4 SQL

SQL significa “Structured Query Language” ou em português “Linguagem de Consulta Estruturada”. Foi criada em 1986 pela American National Standards Institute (ANSI) e é amplamente utilizada nos dias atuais por diferentes sistemas gerenciadores de bancos de dados.

SQL é uma linguagem de pesquisa declarativa utilizada para banco de dados. É a linguagem principal utilizada em gestão de bancos de dados e devido ao fato de ser uma linguagem declarativa, existe uma facilidade de aprendizado que faz com que ela seja amplamente usada (OLIVEIRA, 2023).

Apesar de seguir um padrão, cada gerenciador de banco tem suas particularidades em relação às funcionalidades presentes, podendo apresentar variações de sintaxe e de formato dos dados, além das ferramentas disponíveis.

4.5 PHP

PHP é um acrônimo para a sigla *“Hypertext Preprocessor”* e foi criada em 1995 pelo programador Rasmus Lerdorf. É uma linguagem de sintaxe simples, que mescla o código executado do lado do servidor com HTML, facilitando a criação de conteúdos dinâmicos (MILETTO, 2014).

PHP é uma linguagem de scripts, usada principalmente para aplicações do lado do servidor, capaz de gerar conteúdo dinâmico processado pelo lado do servidor no lado do cliente. É um software gratuito de código aberto que tem como função o desenvolvimento e implementação de serviços web velozes, através de estruturas dinâmicas que permitem uma programação mais ágil. Com ela é possível executar ações mais complexas como processar dados de um formulário através de uma página estática (HTML).

4.5.1 Método POST

As chamadas em PHP são feitas através do protocolo HTTP, utilizado para chamar páginas HTML, junto a chamada do endereço, no método POST, são enviados parâmetros que serão utilizados no código PHP para executar as ações, no formato: ‘Nome do Parâmetro’=’Valor’.

5 MATERIAIS E MÉTODOS

Nesta seção serão discutidos os softwares e as configurações feitas para a criação do aplicativo. Primeiramente, serão apresentados os softwares e os motivos para sua escolha. Após discutidos as ferramentas utilizadas, será apresentada a estrutura definida para o banco de dados que atuará como servidor central e armazenamento dos questionários. Por fim será comentado sobre a construção da aplicação em Delphi, junto com os códigos explicando a funcionalidade, e os scripts PHP que fazem a intermediação da aplicação com o banco de dados.

5.1 MYSQL

MySQL é um gerenciador de banco de dados relacional que utiliza a linguagem SQL. Atualmente, é uma das principais escolhas do mercado para gerenciadores de banco, devido à grande variedade de estratégias de controle e escalabilidade, podendo prover o projeto com um bom conjunto de ferramentas para a gestão de dados do lado do servidor. MySQL possui código aberto o que permite o desenvolvimento de funções e melhorias por terceiros aumentando ainda mais o conjunto de ferramentas disponíveis (ORACLE, [200-?]).

Figura 4 – SGBDs mais utilizados do mundo, dezembro de 2018

341 systems in ranking, December 2018

Rank			DBMS	Database Model	Score		
Dec 2018	Nov 2018	Dec 2017			Dec 2018	Nov 2018	Dec 2017
1.	1.	1.	Oracle	Relational DBMS	1283.22	-17.89	-58.32
2.	2.	2.	MySQL	Relational DBMS	1161.25	+1.36	-156.82
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1040.34	-11.21	-132.14
4.	4.	4.	PostgreSQL	Relational DBMS	460.64	+20.39	+75.21
5.	5.	5.	MongoDB	Document store	378.62	+9.14	+47.85
6.	6.	6.	IBM Db2	Relational DBMS	180.75	+0.87	-8.83
7.	7.	↑ 8.	Redis	Key-value store	146.83	+2.66	+23.59
8.	8.	↑ 10.	Elasticsearch	Search engine	144.70	+1.24	+24.92
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	139.51	+1.08	+13.63
10.	10.	↑ 11.	SQLite	Relational DBMS	123.02	+0.31	+7.82

Fonte: Longen (2024)

MySQL foi desenvolvido em 1994 por uma empresa Sueca chamada MySQL AB, em 2008 foi comprada pela empresa americana Sun Microsystems que obteve controle total do software, já em 2010, a empresa Sun Microsystems foi adquirida pela Oracle, que detém os direitos e desenvolvimento desde então (LONGEN, 2024).

As vantagens do MySQL são robustez da ferramenta, assim como o suporte nativo do RAD Studio para ele. Ele também atua com um modelo de Cliente-Servidor, permitindo que

dispositivos possam acessar o banco no servidor através de chamadas externas, como por exemplo, através de chamadas em PHP. Para o desenvolvimento do projeto foi utilizado a versão “10.4.28-MariaDB” do servidor e o cliente “libmysql - mysqlnd 8.2.4”

5.2 XAMPP

XAMPP é um ambiente de desenvolvimento PHP gratuito de código aberto contendo distribuição do Apache, MySQL, PHP e Perl. Já existente a mais de 10 anos, é uma ferramenta utilizada por diversos desenvolvedores, o que resulta em uma comunidade ativa, facilitando o aprendizado e solução de problemas (APACHE, [200-?]).

A distribuição do XAMPP faz parte do projeto “Apache Friends” que visa criar uma distribuição de fácil uso para criação de servidores web utilizando Apache.

Neste projeto foi utilizada a versão 3.3.0 (6 abril de 2021).

5.3 RAD STUDIO 11 “COMMUNITY EDITION”

RAD Studio é uma plataforma de desenvolvimento criada pela Embarcadero, que traz suporte para a criação de aplicações em Delphi e C++, contendo as principais bibliotecas de desenvolvimento para ambas as linguagens. O software também possui suporte para ferramentas de desenvolvimento das APIs para dispositivos móveis em Android (KALMYKOV, [2016?]). A versão utilizada do RAD Studio e do Delphi, é a versão 11 “Community Edition”, sendo esta a versão mais recente, com distribuição grátis para utilização em ensino, além de possuir compatibilidade com as versões mais recentes do Android e iOS.

5.4 NOTEPAD++

Notepad++ é um editor de texto e código gratuito que além de atuar como um bloco de notas, tem suporte de edição para diferentes linguagens de programação, incluindo HTML, CSS e PHP. Foi desenvolvido por Don HO, com intuito de criar uma ferramenta rápida, de fácil uso, para auxiliar a comunidade de desenvolvedores com recursos que agilizam o desenvolvimento de código, como salvamento automático, tabulação de acordo com a linguagem utilizada, localização e substituição de código, entre outros (HO, [2011?]).

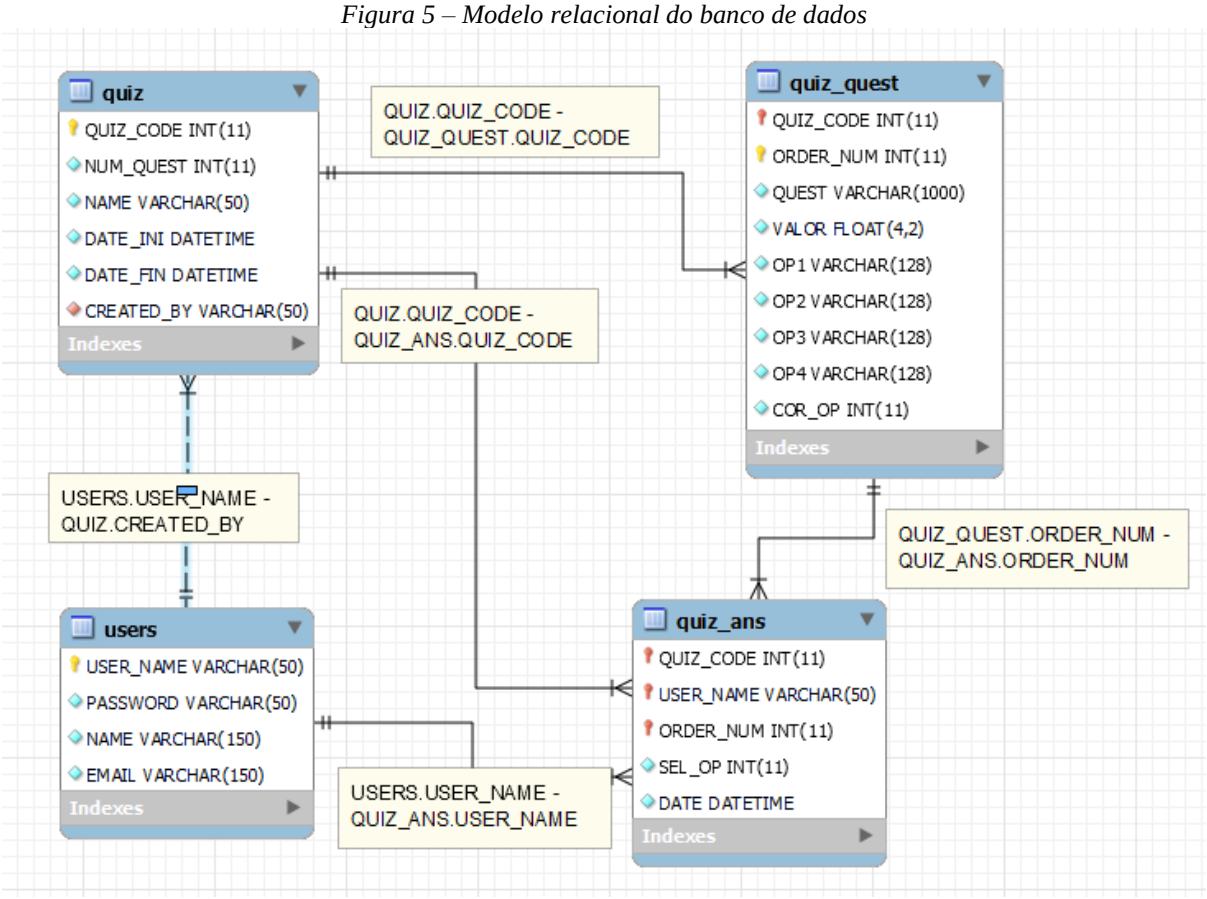
5.5 BANCO DE DADOS

O Banco de dados central do sistema utiliza MySQL, um gerenciador eficiente e com um conjunto de ferramentas adequado para o sistema. O banco é hospedado em servidor local utilizando XAMPP com aplicação APACHE para que possa ser acessado por outros dispositivos presentes na rede. A hospedagem local garante que apenas dispositivos conectados à rede possam acessar a funcionalidade, garantindo uma segurança maior ao sistema, e evitando custos de hospedagem e segurança em servidores terceiros, e evita o risco de ataques externos no sistema.

5.5.1 Estrutura do Banco de Dados

O banco de dados foi estruturado para ter uma forma simples, porém eficiente de guardar todas as informações necessárias para que a aplicação possa operar de forma correta. O banco contém 4 tabelas, sendo elas: ‘QUIZ’, ‘QUIZ_QUEST’, ‘QUIZ_ANS’, ‘USERS’, e uma *VIEW* chamada “NOTA” responsável por unir as informações das tabelas e calcular a nota dos usuários. A estrutura relacional do banco é apresentada na figura 5.

Figura 5 – Modelo relacional do banco de dados



Fonte: Acervo Pessoal

5.5.2 Tabela QUIZ

A tabela “Quiz” é a principal tabela do sistema, nela são guardados os questionários criados pelos usuários, assim como suas informações gerais, como usuário que a criou, e a janela de datas em que vai estar disponível para resposta e o número identificador único. A chave primária da tabela é o código do questionário, garantindo assim a existência de um identificador único para cada questionário criado. A tabela 1 apresenta as colunas presentes no banco de dados para a tabela “QUIZ”.

Tabela 1 – Modelo da tabela QUIZ no banco de dados da aplicação

Coluna	Tipo de Dado	Descrição
QUIZ_CODE	INT	Chave-Primária. Número inteiro identificador do questionário
NAME	VARCHAR	Nome do questionário
NUM_QUEST	INT	Número de perguntas presente no questionário
DATE_INI	DATETIME	Data e Hora Inicial em que o questionário será liberado para resposta
DATE_FIN	DATETIME	Data e Hora Final em que o questionário será liberado para resposta
CREATED_BY	VARCHAR	Usuário criador do questionário. Ligada a tabela USERS

Fonte: Autoria própria.

5.5.3 Tabela QUIZ_QUEST

A tabela “QUIZ_QUEST” contém as perguntas de todos os questionários. Na tabela, as perguntas, os textos, as opções e a alternativa correta são guardadas para serem apresentadas na aplicação, e utilizadas para corrigir as respostas após o envio. A tabela possui duas chaves primárias, sendo elas o identificador do questionário que pertencem, e o número da pergunta. As duas colunas indicadas como chaves primárias garantem que cada questionário terá apenas uma pergunta para cada “número”, porém não impede a existência de mais de uma pergunta por questionário ou a existência de mais de uma pergunta “número 1”, por exemplo. A tabela 2 apresenta as colunas presentes no banco de dados para a tabela “QUIZ_QUEST”.

Tabela 2 – Modelo da tabela QUIZ_QUEST no banco de dados da aplicação

Coluna	Tipo de Dado	Descrição
QUIZ_CODE	INT	Chave-Primária. Número inteiro identificador do questionário. Ligada a tabela Quiz
ORDER_NUM	INT	Chave-Primária. Número inteiro identificador da questão.
QUEST	VARCHAR	Texto da Questão
VALOR	FLOAT	Valor da Questão
OP1	VARCHAR	Texto da primeira opção de resposta
OP2	VARCHAR	Texto da segunda opção de resposta
OP3	VARCHAR	Texto da terceira opção de resposta
OP4	VARCHAR	Texto da quarta opção de resposta
COR_OP	INT	Opção correta da questão

Fonte: Autoria própria.

5.5.4 Tabela QUIZ_ANS

A tabela "QUIZ_ANS" guarda as respostas dos usuários as perguntas dos questionários. Na tabela, são guardados os dados sobre o questionário respondido, o número da pergunta, o usuário, a opção selecionada, e data de resposta. Ao comparar os dados dessa tabela com os apresentados na tabela de perguntas, é possível criar uma avaliação para o usuário de cada um dos questionários respondidos. Nesta tabela, o campo de questionário, número da pergunta e usuário são considerados chaves primárias, portanto, pode sempre existir apenas uma resposta, para uma mesma pergunta, de um mesmo questionário de um mesmo usuário. A tabela 3 apresenta as colunas presentes no banco de dados para a tabela "QUIZ_ANS".

Tabela 3 – Modelo da tabela QUIZ_ANS no banco de dados da aplicação

Coluna	Tipo de Dado	Descrição
QUIZ_CODE	INT	Chave-Primária. Número inteiro identificador do questionário. Ligada a tabela Quiz
ORDER_NUM	INT	Chave-Primária. Número inteiro identificador da questão.
SEL_OP	INT	Resposta selecionada pelo usuário
USER_NAME	VARCHAR	Chave-Primária. Usuário que respondeu à questão
DATE	DATETIME	Data em que a resposta foi enviada

Fonte: Autoria própria.

5.5.5 Tabela USERS

A tabela “USERS” guarda as informações sobre os usuários. Nela são registrados o nome de usuário, o nome completo, e-mail e a senha de forma criptografada, impedindo que até mesmo os administradores do banco tenham acesso à senha. A chave primária da tabela é o nome de usuário, garantindo a existência de que o nome de usuário seja único. A tabela 4 apresenta as colunas presentes no banco de dados para a tabela “USERS”.

Tabela 4 – Modelo da tabela USERS no banco de dados da aplicação

Coluna	Tipo de Dado	Descrição
USER_NAME	VARCHAR	Chave-Primária. Nome de usuário
NAME	VARCHAR	Nome completo do usuário
EMAIL	VARCHAR	Email do usuário
PASSWORD	VARCHAR	Senha criptografada do usuário

Fonte: Autoria própria.

5.5.6 View NOTA

“View” é uma busca no banco de dados que pode ser salva e utilizada como uma tabela do banco. Ela permite selecionar informações de várias tabelas diferentes, manipulá-las e apresentar da forma necessária para a aplicação. Por ser uma busca no banco, ela é constantemente atualizada e pode ser utilizada como um resumo das informações. A “view NOTA” agrupa as respostas do usuário por questionário da tabela “QUIZ_ANS” e compara elas com as respostas corretas de cada questão definidas na tabela “QUIZ_QUEST”, caso a comparação seja igual, ele soma a nota da questão, repetindo para cada questão do questionário, assim conseguindo a nota total da resposta.

A query SQL de criação da “view” é:

CREATE VIEW NOTA AS

select q.QUIZ_CODE AS QUIZ_CODE, q.NAME AS QUIZ_NAME, qa.USER_NAME AS USER_NAME, sum(case when qa.SEL_OP = qq.COR_OP then qq.VALOR else 0 end) AS NOTA, sum(case when qa.SEL_OP = qq.COR_OP then 1 else 0 end) AS ACERTOS, max(qa.DATE) AS DATA from ((quiz_db.quiz q join quiz_db.quiz_quest qq on (q.QUIZ_CODE = qq.QUIZ_CODE)) join quiz_db.quiz_ans qa on (qq.QUIZ_CODE =

qa.QUIZ_CODE and qq.ORDER_NUM = qa.ORDER_NUM)) group by q.QUIZ_CODE,qa.USER_NAME

Esta “view” resulta em uma estrutura similar à de tabela que pode ser utilizada pela aplicação para apresentar as notas dos usuários. A tabela 5 apresenta o modelo da view "NOTA” no banco de dados.

Tabela 5 – Modelo da view NOTA no banco de dados da aplicação

Coluna	Tipo de Dado	Descrição
QUIZ_CODE	INT	Número inteiro identificador do questionário. Ligada a tabela Quiz
QUIS_NAME	VARCHAR	Nome do Questionário
USER_NAME	VARCHAR	Usuário que respondeu a questão
NOTA	FLOAT	Nota final do usuário no questionário
ACERTOS	FLOAT	Acertos totais do usuário no questionário
DATA	DATETIME	Data em que a resposta foi enviada

Fonte: Autoria própria.

5.6 APLICAÇÃO

A aplicação foi criada utilizando Delphi em conjunto com o RAD Studio, permitindo um desenvolvimento rápido integrando construção gráfico simplificada através de elementos prontos, e programação das funcionalidades destes elementos e interações entre si. A comunicação com o banco de dados MySQL é feita através de chamadas HTTP utilizando scripts PHP para enviar e receber informações do servidor central.

5.6.1 Formulários

A programação em Delphi é dividida em formulários, cada um representa uma página a qual o usuário pode ter acesso e nele são colocados os elementos gráficos para interação com o sistema. Além da parte visual, cada formulário possui os códigos que serão executados nos eventos de cada item presente, podendo estes serem executados automaticamente, ou a partir da ação da pessoa utilizando o sistema.

5.6.2 Códigos PHP

Os códigos PHP são responsáveis por receber as chamadas da aplicação e processá-las em relação ao banco de dados. Cada processamento é separado em um arquivo diferente, representando assim um endereço de conexão diferente, e recebe um conjunto de parâmetros que serão utilizados para buscar ou adicionar elementos no banco.

5.6.2.1 Verificação dos parâmetros

Todos os parâmetros recebidos, são verificados antes de serem utilizados no processamento do banco, para assim garantir a integridade dos dados e das funcionalidades.

Primeiro, todos os parâmetros devem estar preenchidos, caso isso não ocorra, o código retorna um erro para o sistema.

Figura 6 – Verificação de parâmetro vazio em PHP

```php
if (empty($_POST["userName"])) {
  echo "Usuario em branco";
  exit;
}
else
{
  $userName = test_input($_POST["userName"]);
}
```

Fonte: Acervo Pessoal

Após garantir a existência do parâmetro, ele passa por uma função que remove possíveis espaços em branco indevidos, assim como caracteres especiais que possam causar erros no processamento dos dados.

Figura 7 – Remoção de caracteres especiais dos parâmetros em PHP

```php
function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
```

Fonte: Acervo Pessoal

5.6.2.2 Conexão ao banco

Para garantir que as chamadas acessem o banco apenas quando necessário, todos os scripts se conectam ao banco antes de fazer a chamada e desconectam após conclui-la ou em caso de erro.

Figura 8 Modelo de código de conexão ao banco MySQL em PHP

```php
$dbServername = "SERVER_EXEMPLO";
$dbUsername = "USUARIO_EXEMPLO";
$dbPassword = "SENHA_EXEMPLO";
$dbname = "BANCO_EXEMPLO";

// Create connection
$conn = new mysqli($dbServername, $dbUsername, $dbPassword, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Falha de Conexão: " . $conn->connect_error);
  exit;
}
```

Fonte: Acervo Pessoal

5.6.3 Formulário “dbConnection” e “varUnit”

Os formulários “dbConnection” e “varUnit” foram criados na aplicação para servir como repositórios para os elementos contextuais do sistema serem guardados. O usuário não interage diretamente com estes elementos, porém eles guardam funções importantes para o sistema, como variáveis “globais”, elementos de conexão entre outros.

5.6.4 Formulário de Login

Quando o usuário abrir a aplicação, será aberto o formulário de login. Este formulário é utilizado para acessar o sistema. Nele estão presentes os campos para preencher o nome de usuário e a senha, assim como os botões para acessar o sistema ou criar uma conta.

Ao pressionar o botão de “Login” o sistema irá verificar o preenchimento dos campos de usuário e senha e com isso, enviar uma solicitação de acesso ao banco de dados, utilizando uma chamada HTTP, para executar o código em PHP *“Login.php”*. Através do método “POST”, é enviado um vetor contendo os parâmetros da chamada, no caso, usuário e senha.

Figura 9 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de Login

```
begin
  loginParameters := TStringList.Create;
  parametro := 'userName=' + userEdit.Text;
  loginParameters.Add(parametro);
  parametro := 'userPassword=' + passEdit.Text;
  loginParameters.Add(parametro);
  loginUrl := varUnit.phpConnect + 'Login.php';
  loginCheck := db.IdHTTP1.Post(loginUrl, loginParameters);
```

Fonte: Acervo Pessoal

O código PHP recebe os parâmetros da chamada e realiza uma busca no banco para verificar se a combinação de usuário e senha são válidos e existentes no banco.

Figura 10 – Código PHP com Query SQL de busca de login de usuário

```
$sql = "SELECT USER_NAME, NAME, EMAIL from users where
USER_NAME = '" . $userName . "' and PASSWORD = PASSWORD('" . $userPassword ."')";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo ":::LOGIN_CONFIRMADO:::";
} else {
    echo "VAZIO";
}
$conn->close();
```

Fonte: Acervo Pessoal

Dependendo do retorno recebido pelo processamento PHP, o usuário acessa o sistema e a página inicial é aberta, ou é retornada uma mensagem de erro ao usuário.

Figura 11 – Código em Delphi de verificação do retorno de Login

```
  if loginCheck = ':::LOGIN_CONFIRMADO:::' then
  begin
    varUnit.userName := userEdit.Text;
    Flogin.Hide;
    Fmm.Show;
    userEdit.Text := '';
    passEdit.Text := '';
  end
  else
  begin
    userEdit.Text := '';
    passEdit.Text := '';
    ShowMessage('Usuario ou Senha Invalidos');
  end;
  loginParameters.Free();
end;
```

Fonte: Acervo Pessoal

5.6.5 Formulário de Cadastro

O formulário de cadastro é utilizado para criar uma conta no sistema. Nele estão presentes os campos de informação do usuário, que devem ser preenchidos para ficarem registrados no sistema.

Ao pressionar o botão de “Cadastrar” o sistema irá verificar o preenchimento dos campos, caso estejam corretamente preenchidos. Irá enviar uma solicitação de inserção ao banco de dados, utilizando uma chamada HTTP ao item *“Signin.php”,* enviando os valores escritos como parâmetros.

Figura 12 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de Cadastro

```
begin
  signinParameters := TStringList.Create;
  parametro := 'userName=' + userEdit.Text;
  signinParameters.Add(parametro);
  parametro := 'userFullName=' + nameEdit.Text;
  signinParameters.Add(parametro);
  parametro := 'userEmail=' + emailEdit.Text;
  signinParameters.Add(parametro);
  parametro := 'userPassword=' + passEdit.Text;
  signinParameters.Add(parametro);
  signinUrl := varUnit.phpConnect + 'Signin.php';
  signinCheck := db.IdHTTP1.Post(signinUrl, signinParameters);
```

Fonte: Acervo Pessoal

O código em PHP irá verificar os parâmetros recebidos, e checar se o nome de usuário já está cadastrado no banco, retornando erro caso aconteça. Caso não exista, irá inserir o novo usuário na tabela “USERS” e retornará a mensagem de sucesso para o sistema.

Figura 13 – Código PHP com Query SQL de cadastro de usuário

```php
$sql = "SELECT USER_NAME from users where USER_NAME = '" . $userName . "'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "Usuário já existente";
} else {
    $sql = "INSERT INTO USERS(USER_NAME, NAME, EMAIL, PASSWORD)
    VALUES ('" . $userName . "','" . $userFullName . "','" . $userEmail .
    "',PASSWORD('" . $userPassword . "'))";
if ($conn->query($sql) === TRUE) {
  echo ":::USUARIO_CADASTRADO:::";
} else {
  echo "Erro no Cadastro do Usuário";
}
}
$conn->close();
```

Fonte: Acervo Pessoal

Por fim, uma notificação no aplicativo indicará o sucesso ou falha na criação da conta.

Figura 14 – Código em Delphi de verificação do retorno de cadastro

```delphi
  if signinCheck = ':::USUARIO_CADASTRADO:::' then
  begin
    ShowMessage('Usuário cadastrado com sucesso!');
    Fsignin.Close;
  end
  else
  begin
    ShowMessage(signinCheck)
  end;
end;
```

Fonte: Acervo Pessoal

5.6.6 Formulário Inicial

Quando o usuário acessa o sistema, será aberto o formulário de inicial. Este formulário contém algumas informações sobre o sistema, e sua função é atuar como uma página inicial do aplicativo.

5.6.6.1 Menu Lateral

O menu lateral está disponível para ser acessado em todos os formulários do sistema (após acessar o sistema), toda a navegação do aplicativo é feita através de seu uso. Ele pode ser aberto arrastando a parte lateral esquerda em direção ao centro, e escondido através do movimento inverso. Ao pressionar um de seus botões, o formulário desejado será carregado, tomando lugar do atual.

Figura 15 – Código em Delphi dos processos e ações do Menu Lateral

```
procedure TFmm.Button2Click(Sender: TObject);
begin
  MultiView1.Visible := TRUE;
end;

procedure TFmm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Flogin.Show;
end;

procedure TFmm.FormCreate(Sender: TObject);
begin
  MultiView1.Visible := FALSE;
end;

procedure TFmm.mvLogoutButtonClick(Sender: TObject);
begin
  Screen.ActiveForm.Close;
  Flogin.Show;
end;

procedure TFmm.mvMainMenuBtClick(Sender: TObject);
begin
  Screen.ActiveForm.Hide;
  Fmm.Show;
  Fmm.SetActive(TRUE);
end;

procedure TFmm.mvQuizAnsBtClick(Sender: TObject);
begin
  Screen.ActiveForm.Hide;
  FquizAns.Show;
end;


procedure TFmm.mvQuizCreateBtClick(Sender: TObject);
begin
  Screen.ActiveForm.Hide;
  FquizCreate.Show;
end;

procedure TFmm.mvQuizViewBtClick(Sender: TObject);
begin
  Screen.ActiveForm.Hide;
  FquizView.Show;
end;
```

Fonte: Acervo Pessoal

No menu, também está presente o botão “Sair”, para que o usuário possa fechar a aplicação, retornando a página de acesso.

5.6.7 Formulário de Criação de Questionário

Utilizando o menu de navegação lateral, é possível abrir o formulário de criação de questionário através do botão “Criar Questionário”. Ao abrir a nova tela, o usuário poderá pressionar o botão “Novo Questionário” para iniciar o processo de criação. A janela de edição será aberta.

A janela de edição começa com os campos referentes às informações gerais sobre o questionário. Sendo elas: nome, número de questões, data inicial e final. Após definida as informações, e pressionando o botão “Criar Questionário”, o usuário será enviado para a janela de criação das perguntas. Todos os valores preenchidos serão guardados temporariamente como variáveis, e utilizados na próxima janela para definir o número de questões editáveis, e depois serem salvar no banco de dados.

Figura 16 – Código em Delphi de troca para aba de questões e limpeza do vetor de questões

```
quizCreateTabC.TabIndex := 2;
numQuest := StrToInt(numQuestBox.Text);
count := 1;
SetLength(questArr, numQuest + 1);
clearQuestArr(1, numQuest);
screenUpdate();
```

Fonte: Acervo Pessoal

A nova janela possui os campos a serem preenchidos para as questões, o menu superior permite ao usuário navegar entre elas, limitados pelo número definido na fase anterior.

Em cada questão, é possível editar os textos da questão, das alternativas, selecionar a alternativa correta, assim como definir uma nota para cada pergunta. As novas informações preenchidas também ficarão guardadas em variáveis temporárias no sistema, até que o processo de salvamento e envio seja realizado.

Figura 17 – Código em Delphi do processo de registro das informações preenchidas na questão

```
procedure TFquizCreate.regData();
begin
  questArr[count].questText := questMemo.Text;
  questArr[count].op1 := opEdit1.Text;
  questArr[count].op2 := opEdit2.Text;
  questArr[count].op3 := opEdit3.Text;
  questArr[count].op4 := opEdit4.Text;
  questArr[count].valor := valorNumBox.Text;

  if opRadio1.IsChecked then
    questArr[count].corOp := 1
  else if opRadio2.IsChecked then
    questArr[count].corOp := 2
  else if opRadio3.IsChecked then
    questArr[count].corOp := 3
  else if opRadio4.IsChecked then
    questArr[count].corOp := 4;

end;
```

Fonte: Acervo Pessoal

Ao pressionar o botão salvar, todas as informações do questionário e das perguntas serão enviadas para o servidor através de chamada HTTP, enviando os valores escritos como parâmetros. Primeiramente serão enviadas as informações referentes ao questionário, para que possa ser gravado os dados no banco, na tabela “QUIZ”, através do item *“QuizCreate.php”*.

Figura 18 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de criação de questionário

```
if (codeCheck = TRUE) AND (whileProtect < 100) then
begin
  quizParameters := TStringList.Create;
  quizParameters.Add('quizNum=' + quizNumServer);
  quizParameters.Add('quizName=' + quizNameEdit.Text);
  quizParameters.Add('quizNumQuest=' + IntToStr(numQuest));
  quizParameters.Add('quizUser=' + varUnit.userName);
  quizParameters.Add('quizDateIni=' + dateIni);
  quizParameters.Add('quizDateFin=' + dateFin);

  quizCreateUrl := varUnit.phpConnect + 'QuizCreate.php';
  quizCreateCheck := db.IdHTTP1.Post(quizCreateUrl, quizParameters);
  quizParameters.Free;
```

Fonte: Acervo Pessoal

O código PHP irá verificar os parâmetros recebidos, e com eles irá fazer o processo de criação de um novo questionário no banco. Caso obtenha sucesso na criação, o fluxo irá retornar sucesso para que a aplicação execute a próxima tarefa.

Figura 19 – Código PHP com Query SQL de criação de questionário

```
if ($result->num_rows > 0) {
    echo ":::CODIGO_EXISTENTE:::";
} else {
    $sql = "INSERT INTO QUIZ(QUIZ_CODE, NUM_QUEST, NAME, DATE_INI, DATE_FIN, CREATED_BY)
    VALUES (" . $quizNum . "," . $quizNumQuest . ",'" . $quizName . "','" . $quizDateIni . "','"
    . $quizDateFin . "','" . $quizUser . "')";
if ($conn->query($sql) === TRUE) {
  echo ":::QUIZ_CADASTRADO:::";
} else {
  echo "Erro no criação do questionario";
}
}
$conn->close();
```

Fonte: Acervo Pessoal

Após ser criado o questionário no banco, as questões serão enviadas uma por vez, utilizando o fluxo *"QuizCreateQuest.php"* para que também possam ser salvas em sua tabela "QUIZ_QUEST", referenciado o número gravado na tabela "QUIZ", criando assim a ligação interna do banco, entre as perguntas e o quiz.

Figura 20 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de criação de questões

```
begin
  quizCreateUrl := varUnit.phpConnect + 'QuizCreateQuest.php';

  for cInsert := 1 to numQuest do
  begin
    questArr[cInsert].valor := StringReplace(questArr[cInsert].valor,
      ',', '.', []);

    quizParameters := TStringList.Create;
    quizParameters.Add('quizNum=' + quizNumServer);
    quizParameters.Add('quizQuestOrderNum=' + IntToStr(cInsert));
    quizParameters.Add('quizQuestText=' + questArr[cInsert]
      .questText);
    quizParameters.Add('quizQuestValor=' + questArr[cInsert].valor);
    quizParameters.Add('quizQuestOp1=' + questArr[cInsert].op1);
    quizParameters.Add('quizQuestOp2=' + questArr[cInsert].op2);
    quizParameters.Add('quizQuestOp3=' + questArr[cInsert].op3);
    quizParameters.Add('quizQuestOp4=' + questArr[cInsert].op4);
    quizParameters.Add('quizQuestCorOp=' +
      IntToStr(questArr[cInsert].corOp));

    quizCreateCheck := db.IdHTTP1.Post(quizCreateUrl, quizParameters);
    quizParameters.Free;
  end;
  ShowMessage('Quiz Criado.' + #13#10 + 'Código: ' + quizNumServer);
  quizCreateTabC.TabIndex := 0;
end;
```

Fonte: Acervo Pessoal

O código PHP irá receber as chamadas de criação uma por vez e verificar os parâmetros recebidos em cada uma delas. Caso estejam corretos, irá salvar as perguntas do questionário no banco, e retornar sucesso para que a aplicação envie a próxima pergunta.

Figura 21 – Código PHP com Query SQL de criação de questões do questionário

```php
$sql = "SELECT QUIZ_CODE, ORDER_NUM from QUIZ_QUEST where QUIZ_CODE = " . $quizNum
. " and ORDER_NUM = " . $quizQuestOrderNum;
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo ":::QUESTAO_EXISTENTE:::";
} else {
    $sql = "INSERT INTO QUIZ_QUEST(QUIZ_CODE, ORDER_NUM, QUEST, VALOR, OP1, OP2, OP3, OP4, COR_OP)
    VALUES (" . $quizNum . "," . $quizQuestOrderNum . ",'" . $quizQuestText . "'," . $quizQuestValor
    . ",'" . $quizQuestOp1 . "','" . $quizQuestOp2 . "','" . $quizQuestOp3 . "','"
    . $quizQuestOp4 . "'," . $quizQuestCorOp . ")";
if ($conn->query($sql) === TRUE) {
  echo ":::QUESTAO_CADASTRADA:::";
} else {
  echo "Erro no criação da questão";
}
}
$conn->close();
```

Fonte: Acervo Pessoal

5.6.7.1 Verificação do código de questionário

Ao criar um questionário, a aplicação irá gerar um novo código aleatório de seis dígitos para ser a referência única do quiz. Para verificar se o código ainda não foi usado, o programa faz uma chamada HTTP ao banco, através do código *"QuizCodeCheck.php"* para verificar a existência do código.

Figura 22 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de criação de código identificador

```
codeCheck := FALSE;
whileProtect := 1;
while (codeCheck = FALSE) AND (whileProtect < 100) do
begin
  quizNumServer := '000000' + IntToStr(random(999999) + 1);
  quizNumServer := RightStr(quizNumServer, 6);
  quizParameters := TStringList.Create;
  quizParameters.Add('quizNum=' + quizNumServer);
  quizCodeCheckUrl := varUnit.phpConnect + 'QuizCodeCheck.php';
  quizCodeCheck := db.IdHTTP1.Post(quizCodeCheckUrl, quizParameters);
  if quizCodeCheck = ':::CODIGO_OK:::' then
  begin
    codeCheck := TRUE;
  end
  else
  begin
    whileProtect := whileProtect + 1
  end;
  quizParameters.Free;
end;
```

Fonte: Acervo Pessoal

Caso o código já esteja cadastrado no sistema, o fluxo PHP retorna erro, e o programa repete o ciclo de geração do código, caso o código esteja disponível, o programa continua o fluxo de criação.

Figura 23 – Código PHP com Query SQL de verificação de existência do código de questionário

```
$sql = "SELECT QUIZ_CODE from QUIZ where QUIZ_CODE = " . $quizNum;
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo ":::CODIGO_EXISTENTE:::";
} else {
    echo ":::CODIGO_OK:::";
}
$conn->close();
```

Fonte: Acervo Pessoal

5.6.8 Formulário de reposta

Pressionar o botão “Responder Questionário” no menu de navegação irá abrir o formulário de resposta. Nessa nova janela, o usuário poderá inserir o código do quiz que deseja responder, e pressionar o botão para buscar as questões. Cada conta poderá responder e enviar apenas uma vez o questionário, caso já tenha enviado uma resposta, uma mensagem de aviso irá aparecer.

Ao buscar um código, o sistema irá fazer uma chamada HTTP para rodar o fluxo *“QuizQuestGet.php”* de busca das perguntas, enviando o código digitado e o nome do usuário como parâmetro.

Figura 24 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de busca de questionário

```
quizQuestGetParameters := TStringList.Create;
quizQuestGetParameters.Add('quizNum='+codeEdit.Text);
quizQuestGetParameters.Add('userName='+varUnit.userName);

quizQuestGetUrl := varUnit.phpConnect + 'QuizQuestGet.php';
quizQuestGetCheck := db.IdHTTP1.Post(quizQuestGetUrl, quizQuestGetParameters);
quizQuestGetParameters.Free;
```

Fonte: Acervo Pessoal

O código PHP primeiramente irá buscar se o código inserido está ligado a um questionário existente. Caso encontre, irá verificar se ele está aberto para respostas (dentro do prazo definido na criação), e depois verificar se não existem respostas enviadas pelo usuário para este item. Após passar por todas as verificações, a janela com as perguntas será aberta, e o fluxo PHP irá trazer as informações das questões.

Figura 25 – Código PHP com Query SQL de busca de questões do questionário

```php
$sql = "SELECT QUIZ_CODE from QUIZ where QUIZ_CODE = " . $quizNum;
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    $sql = "SELECT QUIZ_CODE from QUIZ where QUIZ_CODE = " . $quizNum . " AND NOW() BETWEEN DATE_INI AND DATE_FIN";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {

        $sql = "SELECT * from NOTA where QUIZ_CODE = " . $quizNum . " AND USER_NAME = '" . $userName . "'";
        $result = $conn->query($sql);

        if ($result->num_rows == 0) {
            $sql = "SELECT QUEST, VALOR, OP1, OP2, OP3, OP4 from QUIZ_QUEST where QUIZ_CODE = " . $quizNum;
            $result = $conn->query($sql);

            if ($result->num_rows == 0) {
                echo ":::SEM_QUESTOES:::";
            exit;
            }

            while($row = $result->fetch_assoc())
            {
                echo $row["QUEST"] . ":::" . $row["VALOR"] . ":::" . $row["OP1"] . ":::" . $row["OP2"]
                . ":::" . $row["OP3"] . ":::" . $row["OP4"] . ":<br>";
            }
        } else{
            echo ":::USUARIO_RESPONDIDO:::";
        }
    } else{
        echo ":::QUESTIONARIO_FORA_DE_PRAZO:::";
    }

} else {
    echo ":::QUESTIONARIO_INEXISTENTE:::";
}
$conn->close();
```

Fonte: Acervo Pessoal

A janela de perguntas contém todas as perguntas existentes no questionário, assim como suas alternativas, e valores. Nesta janela, o usuário pode navegar livremente entre as questões e selecionar as alternativas que serão suas respostas. Os valores selecionados serão gravados temporariamente em uma variável do sistema em formato de vetor, até que o ele saia do questionário, ou salve e envie suas respostas através do botão “Salvar”.

A ação de salvar, chama um “loop” que irá pegar as variáveis contendo as opções selecionadas, defini-las como parâmetros, e chamar o fluxo PHP *“QuizAnsSend.php”* que salva as respostas no banco de dados.

Figura 26 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de envio de resposta

```
quizAnsSendUrl := varUnit.phpConnect + 'QuizAnsSend.php';
errorCheck := FALSE;
prazoCheck := FALSE;
for indexSend := 1 to questNum do
begin
  quizAnsSendParameters := TStringList.Create;
  quizAnsSendParameters.Add('quizNum=' + quizCode);
  quizAnsSendParameters.Add('userName=' + varUnit.userName);
  quizAnsSendParameters.Add('orderNum=' + IntToStr(indexSend));
  quizAnsSendParameters.Add('selectedOp=' + IntToStr(selOp[indexSend-1]));

  quizAnsSendCheck := db.IdHTTP1.Post(quizAnsSendUrl,
    quizAnsSendParameters);
  quizAnsSendParameters.Free;

  if quizAnsSendCheck = ':::QUESTIONAO_NAO_ENCONTRADO:::' then
  begin
    errorCheck := TRUE;
  end
  else if quizAnsSendCheck = ':::FORA_DE_PRAZO:::' then
  begin
    prazoCheck := TRUE;
  end
  else if quizAnsSendCheck <> ':::RESPOSTA_CADASTRADA:::' then
  begin
    errorCheck := TRUE;
  end;

end;
```

Fonte: Acervo Pessoal

O código PHP irá receber as chamadas uma por vez e verificar os parâmetros recebidos em cada uma delas. Caso estejam corretos, irá salvar a resposta da questão no banco, e retornar sucesso para que a aplicação envie a próxima resposta.

Figura 27 – Código PHP com Query SQL de envio de respostas de questões do questionário

```php
$sql = "SELECT QUIZ_CODE from QUIZ where QUIZ_CODE = " . $quizNum;
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    $sql = "SELECT QUIZ_CODE from QUIZ where QUIZ_CODE = " . $quizNum . " AND NOW() BETWEEN DATE_INI AND DATE_FIN";
    $result = $conn->query($sql);
    if ($result->num_rows == 0) {
        echo ":::FORA_DE_PRAZO:::";
        exit;
    }

    $sql = "INSERT INTO QUIZ_ANS(QUIZ_CODE, ORDER_NUM, USER_NAME, SEL_OP, DATE)
    VALUES (" . $quizNum . "," . $orderNum . ",'" . $userName . "'," . $selectedOp . ",NOW())";
    if ($conn->query($sql) === TRUE) {
    echo ":::RESPOSTA_CADASTRADA:::";
} else {
  echo ":::ERRO_SALVAR_RESPOSTA:::";
}

} else {
    echo ":::QUESTIONAO_NAO_ENCONTRADO:::";
}
$conn->close();
```

Fonte: Acervo Pessoal

Após salvar, o aplicativo irá retornar para a página inicial do formulário de resposta, e apresentará uma mensagem de sucesso.

5.6.9 Formulário de Histórico

O formulário de histórico pode ser acessado através do botão “Histórico” no menu de navegação. Essa janela é dividida em duas abas, contendo todos os questionários criados, e todos os questionários respondidos pelo usuário.

Quando essa janela é aberta, são realizadas duas chamadas HTTP para os itens *“SearchCreatedHist.php”* e *“SearchAnsweredHist.php”*, estes fluxos são responsáveis por realizar as buscas no banco de dados, para trazer as informações que irão popular as listas. A primeira chamada e código PHP são responsáveis por buscar na tabela “QUIZ” todos os itens em que o usuário está listado como criador, e devolvê-los à aplicação.

Figura 28 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de busca de questionários criados

```delphi
//Preenche lista de questionarios criados
 searchParameters := TStringList.Create;
 parametro := 'userName=' + varUnit.userName;
 searchParameters.Add(parametro);
 searchUrl := varUnit.phpConnect + 'SearchCreatedHist.php';
 searchCheck := db.IdHTTP1.Post(searchUrl, searchParameters);
```

Fonte: Acervo Pessoal

Figura 29 – Código PHP com Query SQL de busca de questionários criados

```php
$sql = "SELECT q.QUIZ_CODE, q.NAME, q.NUM_QUEST, q.DATE_INI, q.DATE_FIN from QUIZ q where q.CREATED_BY = '"
. $userName . "' order by q.DATE_FIN desc";
$result = $conn->query($sql);

if ($result->num_rows == 0) {
    echo ":::SEM_QUESTIONARIOS:::";
    exit;
}

while($row = $result->fetch_assoc())
{
    echo $row["QUIZ_CODE"] . ":::" . $row["NAME"] . ":::" . $row["NUM_QUEST"] . ":::" . $row["DATE_INI"]
    . ":::" . $row["DATE_FIN"] . ":<br>";
}
$conn->close();
```

Fonte: Acervo Pessoal

A segunda chamada e código PHP buscam na view “NOTA” as respostas enviadas pela conta.

Figura 30 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de busca de questionários respondidos

```delphi
//Preenche lista de questionarios respondidos
searchParameters := TStringList.Create;
parametro := 'userName=' + varUnit.userName;
searchParameters.Add(parametro);
searchUrl := varUnit.phpConnect + 'SearchAnsweredHist.php';
searchCheck := db.IdHTTP1.Post(searchUrl, searchParameters);
```

Fonte: Acervo Pessoal

Figura 31 – Código PHP com Query SQL de busca de questionários respondidos

```php
$sql = "SELECT n.QUIZ_CODE, n.QUIZ_NAME, n.USER_NAME, n.NOTA, n.ACERTOS, n.DATA from NOTA n where n.USER_NAME = '"
. $userName . "' order by n.DATA desc";
$result = $conn->query($sql);

if ($result->num_rows == 0) {
    echo ":::SEM_QUESTIONARIOS:::";
    exit;
}

while($row = $result->fetch_assoc())
{
    echo $row["QUIZ_CODE"] . ":::" . $row["QUIZ_NAME"] . ":::" . $row["USER_NAME"] . ":::" . $row["NOTA"]
    . ":::" . $row["ACERTOS"] . ":::" . $row["DATA"] . ":<br>";
}
$conn->close();
```

Fonte: Acervo Pessoal

5.6.9.1 Aba de questionários criados

A aba de questionários criados contém uma lista com itens criados pelo usuário (utilizando as informações adquiridas na chamada *"SearchCreatedHist.php"*). Nessa lista, estarão presentes o nome e o código de cada quiz, e poderão ser pressionados para abrir mais informações sobre eles.

Ao selecionar um item da lista, será feita uma nova chamada HTTP para rodar a rotina do item *"SearchCreatedAns.php"* que busca as respostas para o questionário, utilizando a view "NOTA".

Figura 32 – Código em Delphi de chamada HTTP POST com envio dos parâmetros de busca de respostas ao questionário criado

```
searchParameters := TStringList.Create;
parametro := 'quizCode=' + createdArray[selectedCreIndex,0];
searchParameters.Add(parametro);
searchUrl := varUnit.phpConnect + 'SearchCreatedAns.php';
searchCheck := db.IdHTTP1.Post(searchUrl, searchParameters);
```

Fonte: Acervo Pessoal

Figura 33 – Código PHP com Query SQL de busca de respostas ao questionário criado

```
$sql = "SELECT n.QUIZ_CODE, n.QUIZ_NAME, u.NAME, n.NOTA, n.ACERTOS, n.DATA from NOTA n inner join USERS u
on n.USER_NAME = u.USER_NAME where n.QUIZ_CODE = '" . $quizCode . "' order by n.USER_NAME asc";
$result = $conn->query($sql);

if ($result->num_rows == 0) {
    echo ":::SEM_QUESTIONARIOS:::";
    exit;
}

while($row = $result->fetch_assoc())
{
    echo $row["QUIZ_CODE"] . ":::" . $row["QUIZ_NAME"] . ":::" . $row["NAME"] . ":::" . $row["NOTA"] . ":::"
    . $row["ACERTOS"] . ":::" . $row["DATA"] . ":<br>";
}
$conn->close();
```

Fonte: Acervo Pessoal

Concluída a busca, uma nova janela será aberta, contendo suas informações, como nome, número de perguntas, data de disponibilidade e uma nova lista, apresentando o nome de todos que enviaram suas respostas, e sua nota.

Selecionar uma linha da lista, irá abrir uma nova janela, contendo as informações sobre o envio da resposta específica, como nome do usuário, data de envio, acertos e nota.

5.6.9.2 Aba de questionários respondidos

A aba de questionários respondidos contém uma lista com as respostas enviadas pelo usuário (utilizando as informações adquiridas na chamada *"SearchAnsweredHist.php"*). Nessa lista, estarão presentes o nome e o código de cada quiz, e poderão ser pressionados para abrir mais informações sobre eles.

Na nova janela aberta, as informações sobre a resposta estarão presentes, como nome do questionário, código, nota, acertos e data enviada da resposta.

6 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados obtidos durante o desenvolvimento do projeto, contendo imagens representando o fluxo padrão de utilização da ferramenta. As telas e botões apresentados estão ligados aos desenvolvimentos descritos no capítulo 5, e correspondem de acordo com as rotinas apresentadas.

6.1 FORMULÁRIO DE LOGIN

A aplicação é aberta no formulário de login, nele o usuário terá as opções de acessar utilizando sua conta, ou cadastrar uma nova.

Figura 34 – Tela de Login do aplicativo

Fonte: Acervo Pessoal

Pressionar o botão de cadastro irá abrir o formulário de cadastro. Por sua vez, digitar seu usuário e senha e pressionar o botão “Login”, caso as informações digitadas estejam corretas, o sistema será acessado.

6.2 FORMULÁRIO DE CADASTRO

Ao abrir o formulário de cadastro, o usuário deve preencher os campos com as informações solicitadas, e pressionar o botão “Cadastrar” para criar uma conta. Caso deseje sair da janela sem criar um registro, pode pressionar o botão “Sair”

Figura 35 – Tela de cadastro do Aplicativo

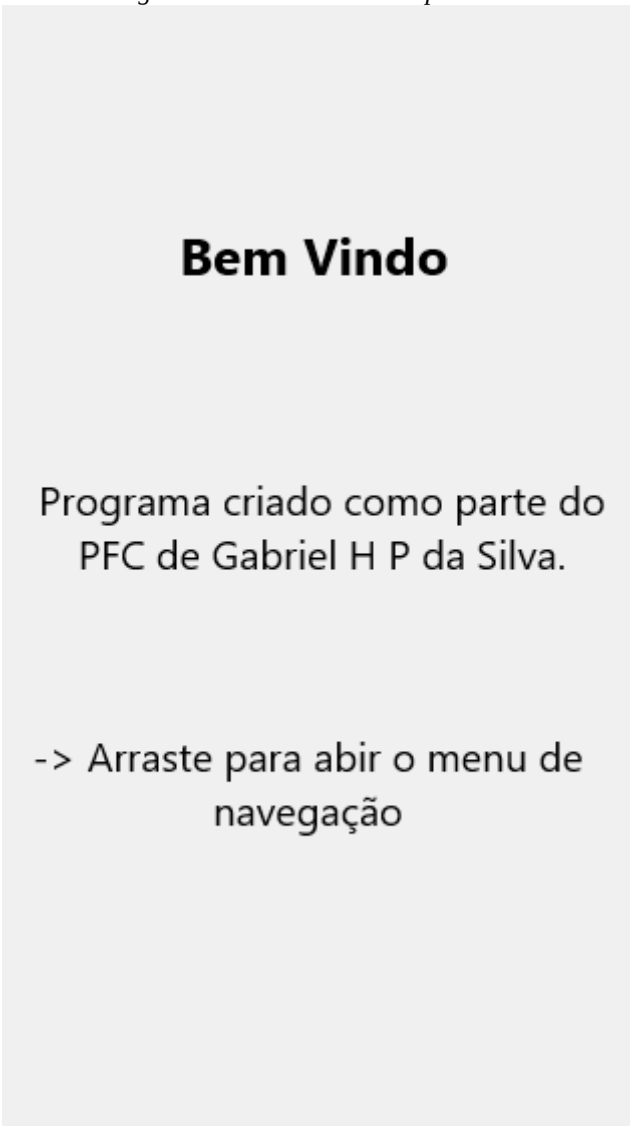
Fonte: Acervo Pessoal

Em ambos os casos, o programa retornará a tela de login após concluir as devidas ações.

6.3 FORMULÁRIO INICIAL

O formulário inicial contém algumas informações sobre o aplicativo e é utilizado apenas como ponto inicial do sistema após o usuário realizar o acesso.

Figura 36 – Tela Inicial do aplicativo

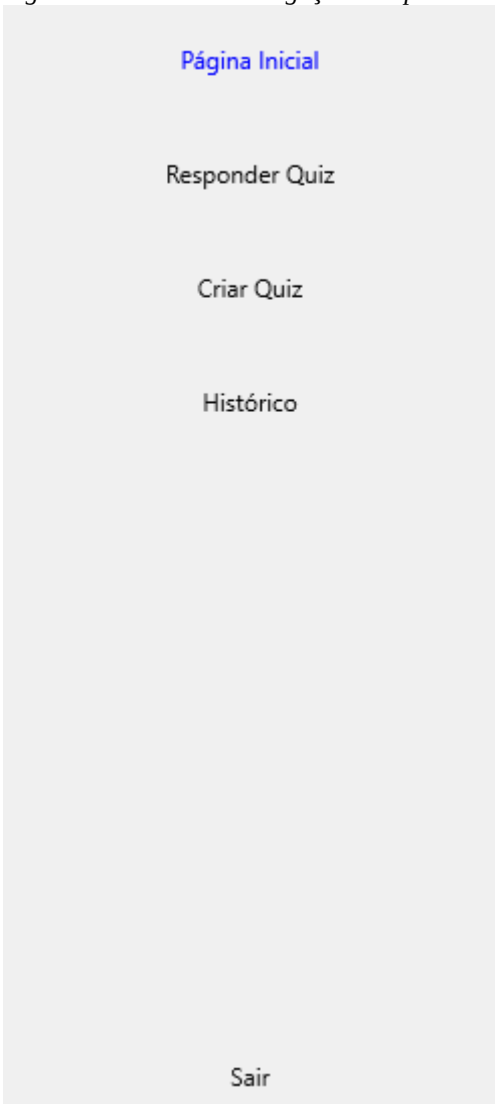


Fonte: Acervo Pessoal

6.4 MENU DE NAVEGAÇÃO

O menu de navegação lateral está disponível em todos os formulários, nele é possível navegar entre as diferentes páginas do aplicativo. Ele pode ser acessado arrastando a lateral esquerda da tela no sentido contrário.

Figura 37 – Menu de navegação do aplicativo



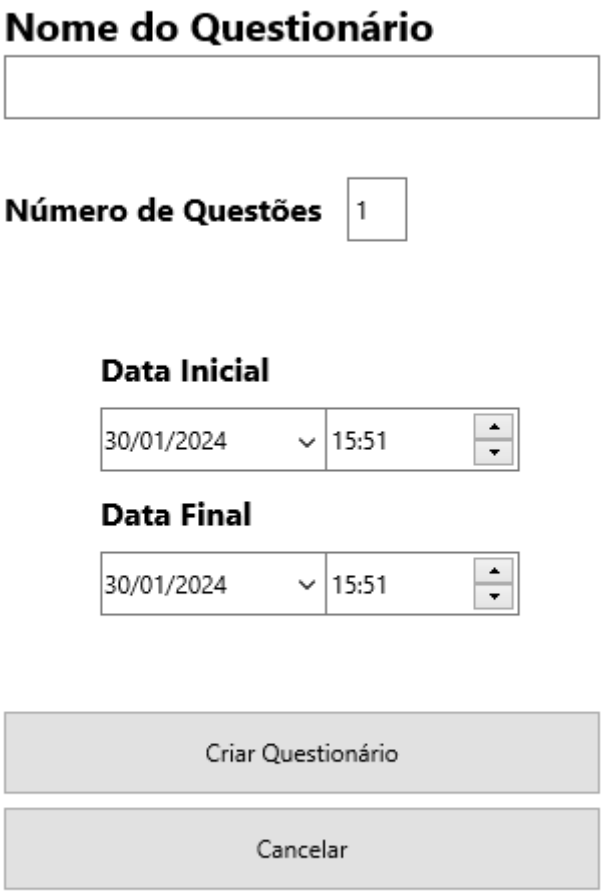
Fonte: Acervo Pessoal

6.5 FORMULÁRIO DE CRIAÇÃO DE QUESTIONÁRIO

Neste formulário o usuário pode criar um questionário para ser respondido. Na tela inicial, estará presente apenas o botão “Novo questionário”.

Ao pressionar o botão, uma nova janela será aberta para que as informações do questionário sejam preenchidas, como nome, número de questões e o período no qual vai estar disponível para ser respondido.

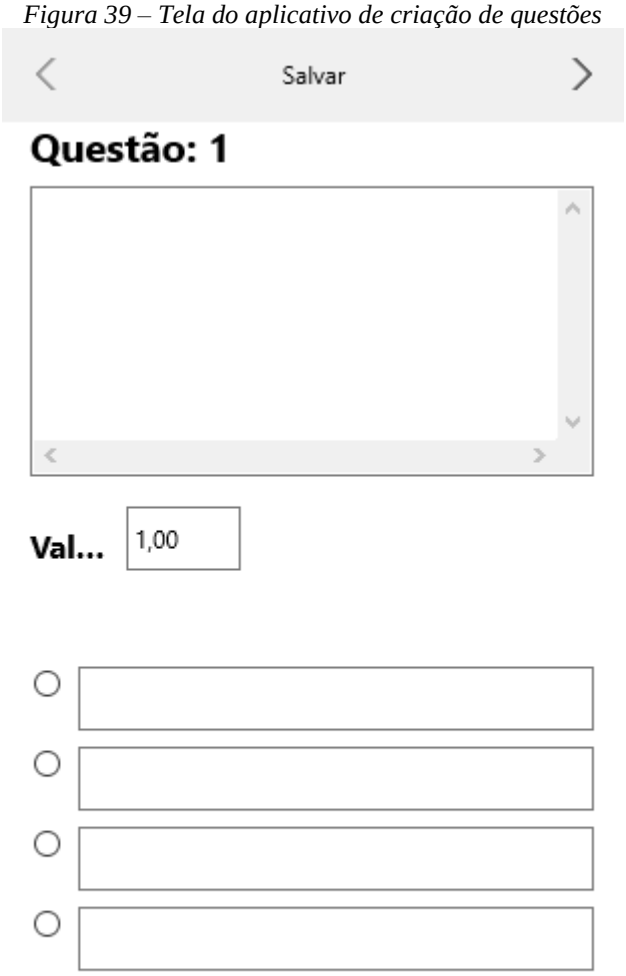
Figura 38 – Tela do aplicativo de criação de questionário



Fonte: Acervo Pessoal

Após preencher as informações, e pressionar o botão “Criar Questionário”, o usuário será enviado para a página de criação e edição das perguntas.

Figura 39 – Tela do aplicativo de criação de questões



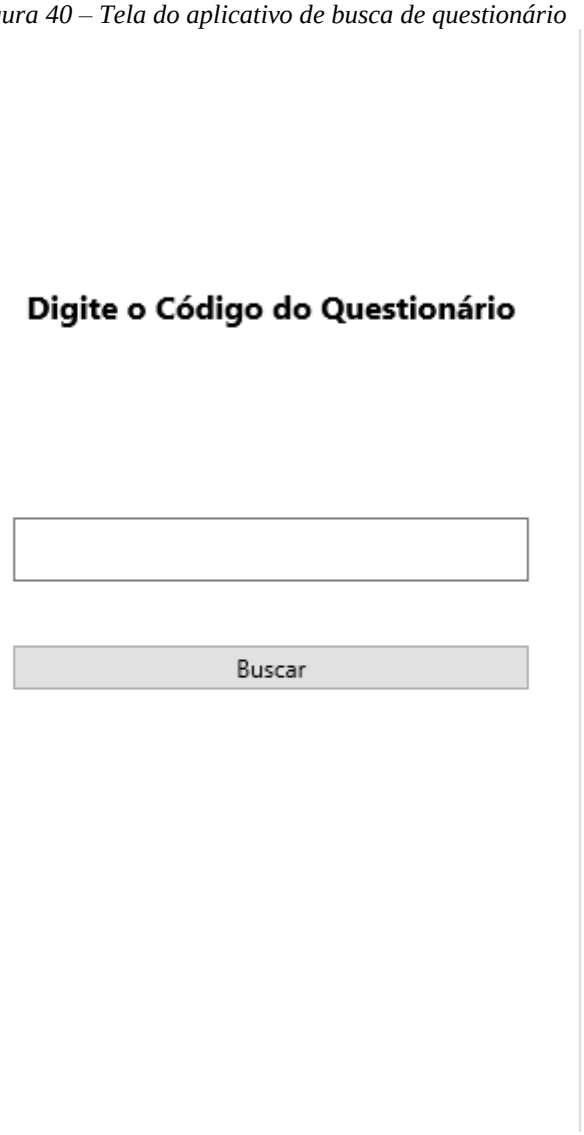
Fonte: Acervo Pessoal

Na tela de criação de perguntas, devem ser informados os textos e valores de cada uma delas, assim como suas alternativas e indicação de alternativa correta. Após finalizado, utiliza-se o botão “Salvar” para enviar as informações ao banco de dados, salvando assim o questionário. O aplicativo irá retornar a página inicial do formulário, e uma mensagem indicando o sucesso da operação e o código do questionário irá aparecer.

6.6 FORMULÁRIO DE RESPOSTA

O formulário de resposta é utilizado para responder questionários existentes. Ao abrir este questionário, haverá um campo para que o código do quiz desejado seja digitado, e um botão para procurar as questões.

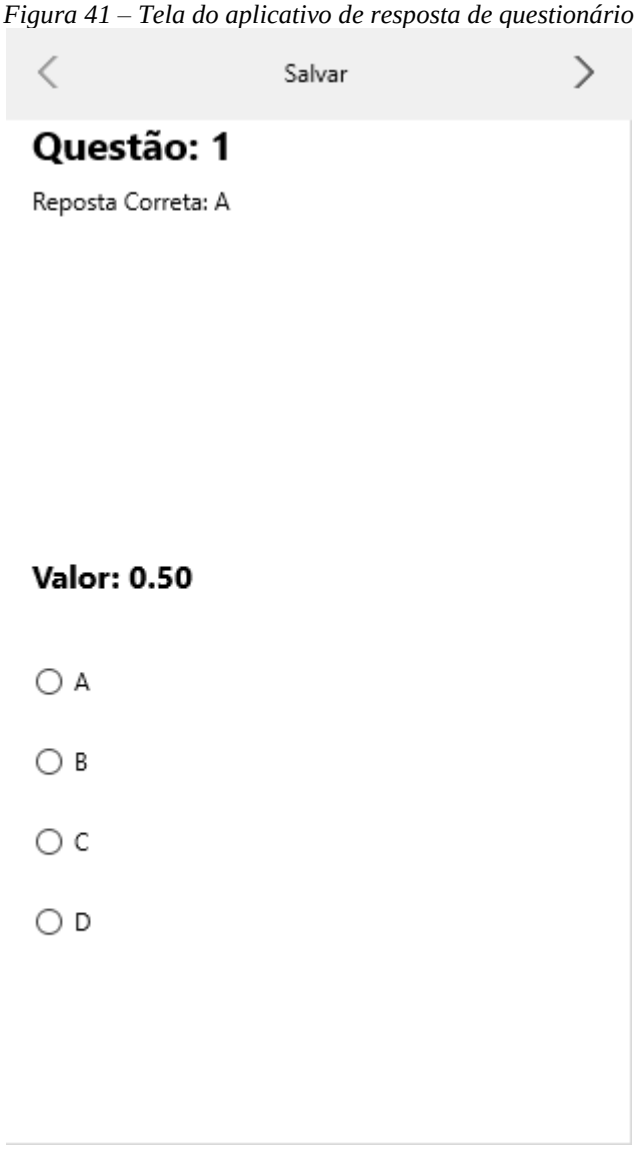
Figura 40 – Tela do aplicativo de busca de questionário



Fonte: Acervo Pessoal

Caso o quiz seja encontrado, uma nova página será aberta, contendo as questões a serem respondidas. O usuário pode navegar pelas questões utilizando o menu superior. Ao preencher as respostas, o usuário deve pressionar o botão “Salvar” para enviar suas respostas ao servidor.

Figura 41 – Tela do aplicativo de resposta de questionário



Fonte: Acervo Pessoal

Uma mensagem de sucesso irá aparecer, confirmando o envio das respostas.

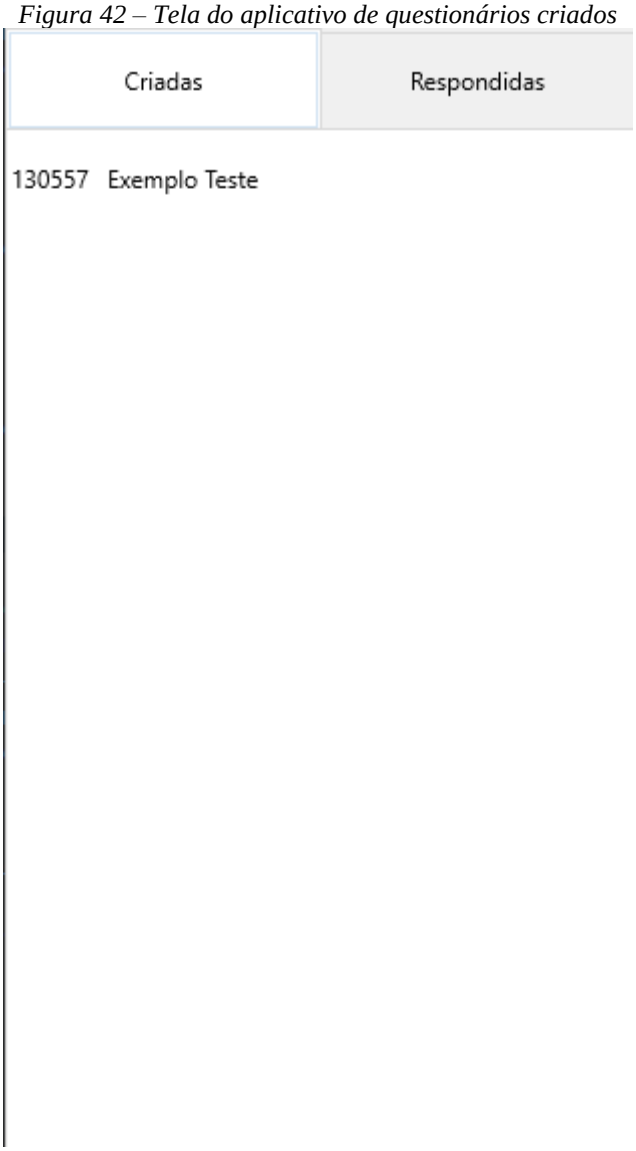
6.7 FORMULÁRIO DE HISTÓRICO

O formulário de histórico contém todos os questionários criados e respondidos pelo usuário, divididos em duas abas. O usuário pode navegar entre as abas para alternar entre os criados e respondidos.

6.7.1 Aba de questionários criados

A aba de questionários criados contém uma lista com todos criados pelo usuário. Ao pressionar um dos itens da lista, uma nova janela será aberta, contendo as informações do item selecionado.

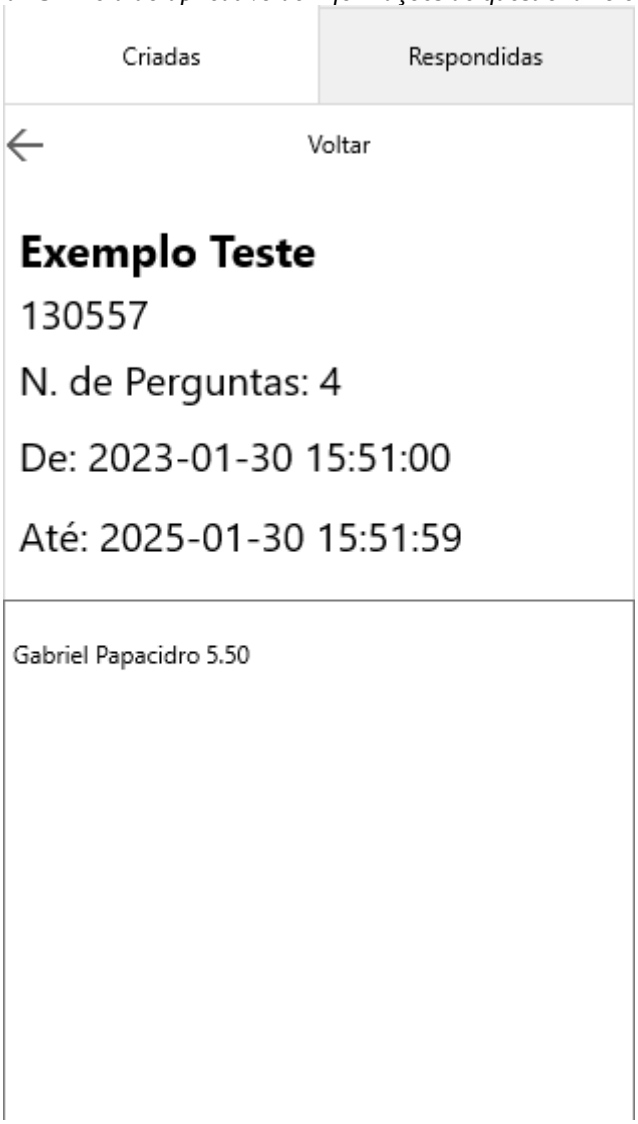
Figura 42 – Tela do aplicativo de questionários criados



Fonte: Acervo Pessoal

Nesta nova janela, são apresentadas informações como nome, número do quiz e período de atividade, assim como uma nova lista contendo todas as respostas enviadas a ele. Selecionar um item desta lista, irá abrir uma janela contendo informações sobre a resposta.

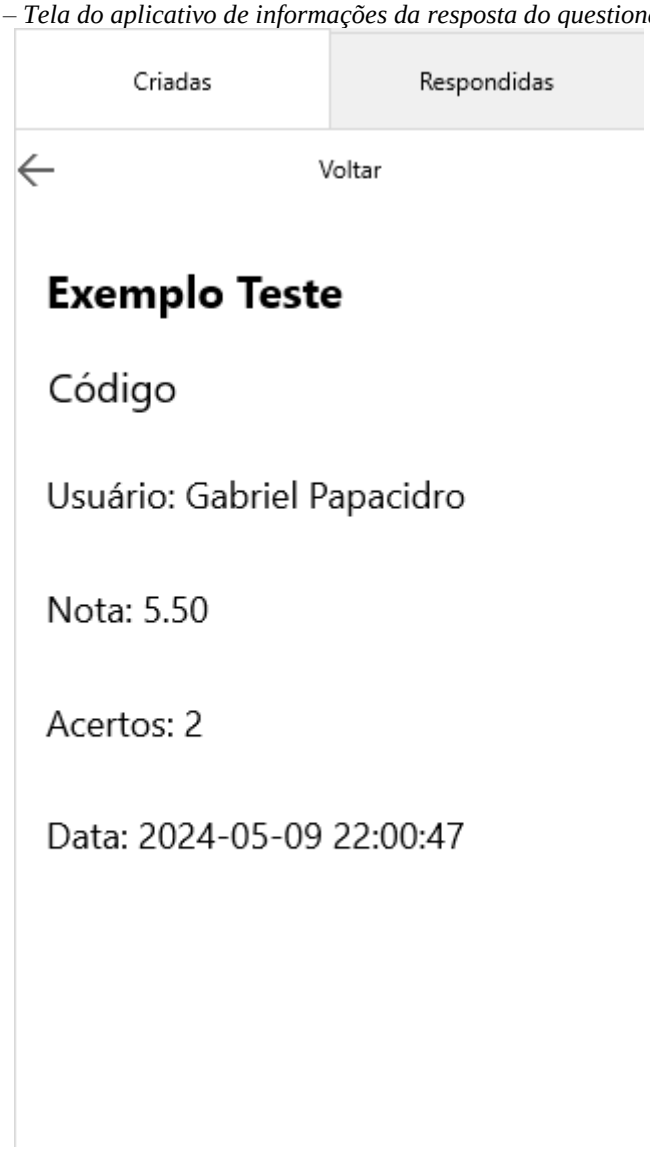
Figura 43 – Tela do aplicativo de informações de questionário criado



Fonte: Acervo Pessoal

Nesta terceira janela, as informações de usuário, nome, data de resposta, acertos e nota poderá ser visualizada pelo criador, podendo assim visualizar dados sobre a resposta enviada.

Figura 44 – Tela do aplicativo de informações da resposta do questionário criado

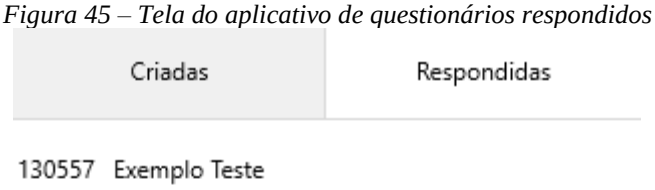


Fonte: Acervo Pessoal

6.7.2 Aba de questionários respondidos

A aba de questionários respondidos contém uma lista com todas as respostas enviadas pelo usuário. Ao pressionar um dos itens da lista, uma nova janela será aberta, contendo as informações do item selecionado.

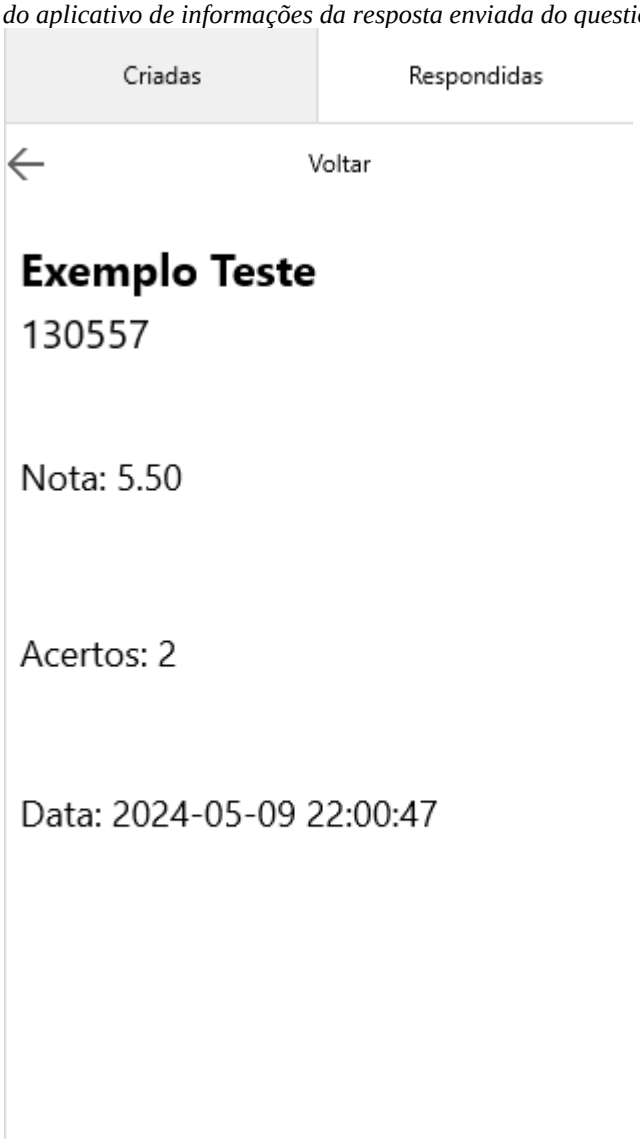
Figura 45 – Tela do aplicativo de questionários respondidos



Fonte: Acervo Pessoal

Nesta nova janela, as informações sobre data da resposta, acertos e nota poderá ser visualizada pelo usuário, podendo assim ter uma visão geral sobre seu desempenho no teste.

Figura 46 – Tela do aplicativo de informações da resposta enviada do questionário



Fonte: Acervo Pessoal

7 CONCLUSÃO

Após concluído o projeto, analisando os resultados obtidos e as funcionalidades implementadas, são chegadas as conclusões. O projeto alcançou seu objetivo no desenvolvimento de um aplicativo para criação de questionários de múltipla-escolha para ser utilizado como apoio durante as aulas. O aplicativo permite a criação de contas, questionários e perguntas com alternativas editáveis, permitindo aos alunos responderem as questões de forma independente, acessando os questionários através de um código disponibilizado. O sistema limita as respostas dos usuários automaticamente, ao liberar e bloquear o acesso ao quiz de acordo com a data e hora definidas pelo criador. O produto automaticamente calcula a nota das respostas e devolve de forma fácil e acessível, para que possa ser visualizado quando necessário, tanto pela pessoa que respondeu, quanto por quem criou as perguntas.

Além de tudo, o sistema se foi apto ao ser instalado em um servidor local, permitindo acesso de forma remota por todos os membros conectados à mesma rede, além de prover uma aplicação para computadores utilizando Windows, assim como aparelhos móveis baseados em Android.

Vale ressaltar, que como toda aplicação computacional, o sistema pode sempre ser melhorado com a criação de novas atualizações, e adições de ferramentas, como por exemplo, edição dos testes, sistemas de segurança mais elaborados, adição de imagens e vídeos aos questionários, além de novos sistemas que possam auxiliar na gestão e particularidades de que possam surgir devido a variedade de aplicações. Novos discentes, apoiados e orientados podem seguir o desenvolvimento, garantindo assim, sempre uma melhora no sistema.

REFERÊNCIAS

ANDRADE, Marcos Vinícius Mendonça; ARAÚJO, Carlos Fernando Jr; SILVEIRA, Ismar Frango**.** Critérios de qualidade para aplicativos educacionais no contexto dos dispositivos móveis (m-learning). *In*: XX Congreso Internacional de Informática Educativa, TISE 2015, Santiago, Chile. **Nuevas ideas en informática educativa vol. 11**. Santiago: Jaime Sánchez, 2015. p. 544- 549. Disponível em: https://www.tise.cl/volumen11/TISE2015/521-526.pdf. Acesso em: 05 maio 2024

APACHE. [Página inicial]. [S. l.]: Apache, [201-?]. Disponível em: https://www.apachefriends.org/pt_br/index.html. Acesso em: 23 abr. 2024.

BOTTENTUIT, João Batista Junior. **Aplicativos de interação em sala de aula: análise de três possibilidades pedagógicas com recursos digitais.** UFMA, São Luis – Maranhão: 15 abr. 2020. Disponível em: https://periodicos.uepa.br/index.php/cocar/article/view/3313/1683. Acesso em: 04 maio 2024.

CANTÙ, Marco. **Mastering delphi 6**. [S. l.]: SYBEX, E-book, [201-?]. Disponível em: https://books.google.com.br/books?id=YbS3dPcobl8C&lpg=PP4&ots=wrSPddkcub&dq=delphi%20programming%20language&lr&hl=pt-BR&pg=PP1#v=onepage&q=delphi%20programming%20language&f=false. Acesso em: 01 maio 2024.

CNN. CNN Brasil. **Procura por profissionais de tecnologia cresce 671% durante a pandemia**. São Paulo: CNN, 27 out. 2021. Disponível em: https://www.cnnbrasil.com.br/economia/procura-por-profissionais-de-tecnologia-cresce-671-durante-a-pandemia/. Acesso em: 05 maio 2024.

EMBARCADERO. **Fire DAC: Multi-Device access library.** [S. l.]: Embarcadero, [201-?a]. Disponível em: https://www.embarcadero.com/br/products/rad-studio/firedac. Acesso em: 05. abr 2024.

EMBARCADERO. **What is Delphi.** [S. l.]: Embarcadero, [201-?b]. Disponível em: https://www.embarcadero.com/products/delphi. Acesso em: 05 abr. 2024.

ESCOLA SUPERIOR DE REDES**. Sistemas gerenciadores de banco de dados – SGBD: um guia com as informações mais relevantes dessa tecnologia**. [S. l.]: 13 abr. 2023. Disponível em: https://esr.rnp.br/desenvolvimento-de-sistemas/sistemas-gerenciadores-de-banco-de-dados/. Acesso em: 06 maio 2024.

FREITAS, Sissi Maria. **Uso de aplicativos como ferramenta para trabalhar educação em saúde no ensino médio.** Universidade do estado do Rio Grande do Norte. Mossoró: 2019. Disponível em: https://www.profbio.ufmg.br/wp-content/uploads/2021/01/SissiFreitasI-TCM-Final.pdf. Acesso em: 25 abr. 2024.

GALENDI, Breno Zielinski. **Desenvolvimento de aplicativo educacional**. 2021. Trabalho de conclusão de curso. Universidade Estadual Paulista, Instituto de Ciência e Tecnologia de Sorocaba. Sorocaba: 2021.

HO, Don. **Wha tis Notepad++**. [S. l.]: [2011?]. Disponível em: https://notepad-plus-plus.org/. Acesso em: 27 abr. 2024.

KALMYKOV, Yuriy. **Programaçâo em Delphi para iniciantes.** [S. l.]: Embarcadero, [2016?]. Disponível em: https://embarcaderobr.com.br/wp-content/uploads/2016/10/DelphiProgramming4Beginners_PTBR_1.1.pdf. Acesso em: 05 abr. 2024.

LEITE, Bruno Silva. M-Learning: o uso de dispositivos móveis como ferramenta didática no ensino de química**. Revista Brasileira de Informática na Educação**, **Volume 22, Número 3**. [S.l.]: RBIE, 2014. p. 55 - 68. Disponível em: https://www.academia.edu/11729289/M_learning_o_uso_de_Dispositivos_M%C3%B3veis_como_ferramenta_did%C3%A1tica_no_Ensino_de_Qu%C3%ADmica. Acesso em: 04 maio 2024

LONGEN, Andrei. **O que é MySQL? Guia simples e direto para iniciantes.** [S. l.]: Hotsinger tutoriais, 01 fev. 2024. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-mysql. Acesso em: 01 maio 2024.

MILETTO, Evando Manra; BERTAGNOLLI, Silvia de Castro. **Desenvolvimento de software II: Introdução ao desenvolvimento web com HTML, CSS, JAVASCRIPT e PHP.** Porto Alegre: Bookman, 2014. E-book. Disponível em: https://books.google.com.br/books?id=lcLFAwAAQBAJ&lpg=PR1&ots=kSLIqz06ur&dq=PHP&lr&hl=pt-BR&pg=PR1#v=onepage&q=PHP&f=false. Acesso em: 02 maio 2024.

OLIVEIRA, Danielle; DIAS, Giovanna. **Saiba tudo sobre SQL – A linguagem padrão para trabalhar com banco de dados relacionais.** [S. l.]: Alura, 29 nov. 2023. Dísponivel em: https://www.alura.com.br/artigos/o-que-e-sql#:~:text=SQL%20%C3%A9%20a%20sigla%20para,Microsoft%20SQL%20Server%2C%20entre%20outros. Acesso em: 04 maio 2024.

ORACLE. [Página inicial]. [S. l.]: Oracle, [200-?]. Disponível em: https://www.mysql.com/. Acesso em 24. abr. 2024.

PCIEVITCH, Yuri. **História da programação.** [S. l.]: Info Escola 2021. Disponível em: https://www.infoescola.com/informatica/historia-da-programacao/. Acesso em: 06 maio 2024.

SCHAFFHAUSER, Dian. **Google Classroom is top educational app download.** [S. l.]: **T**he Journal, 09 abr. 2020. Disponível em: https://thejournal.com/articles/2020/04/09/google-classroom-is-top-education-app%20download.aspx. Acesso em: 09 maio 2024.

SHRESTHA, Prashansa. **Educational apps in focus: Trends and strategies for success in 2023 and beyond.** [S. l.]: Adjust, 19 set. 2023. Disponível em**:** https://www.adjust.com/blog/educational-app-trends-insights-strategies/. Acesso em: 25 abr. 2024.

SONEGO, Anna Helena Silveira; BEHAR, Patricia Alejandra. M-Learning: Reflexões e perspectivas com o uso de aplicativos educacionais. *In*: XX Congreso Internacional de Informática Educativa, TISE 2015, Santiago, Chile. **Nuevas ideas en informática educativa vol. 11**. Santiago: Jaime Sánchez, 2015. p. 521- 526. Disponível em: https://www.tise.cl/volumen11/TISE2015/544-549.pdf. Acesso em: 05 maio 2024