SENAC – PERNAMBUCO

Curso Técnico em Programador de Sistemas

Documentação de Softwares: SoluTech

Programador de Sistemas: Turma Manhã

Orientadora: Profª Fernanda de Melo Reis

PETROLINA

2022

ANA MADALENA SILVA

CAMILA APARECIDA SOUZA

DANIEL SIQUEIRA

FELIPI RODRIGUES

FERNANDO SANTOS

ISABELLY COSTA

JOSY BARROS

JONATHAS RODRIGUES

MATEUS ALENCAR

THIAGO MENDES

Documentação de Software

Projeto Integrador apresentado ao Curso de Programador de Sistemas, como parte dos requisitos necessários para a conclusão do curso.

Orientador(a): Profª Fernanda Reis

Líder de Projeto, Scrum Master: Mateus Alencar

PETROLINA

2022

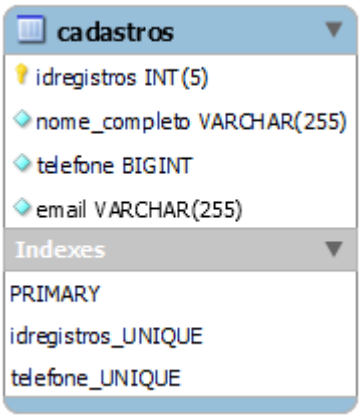
Documentação do Sistema

Banco de dados

1. BackEnd - Modelo Conceitual e Modelo Lógico

Equipe: Mateus Alencar e Thiago Mendes

Modelo Conceitual:



Modelo Lógico:

```
21
22  DROP TABLE IF EXISTS `cadastros`;
23  /*!40101 SET @saved_cs_client     = @@character_set_client */;
24  /*!50503 SET character_set_client = utf8mb4 */;
25  CREATE TABLE `cadastros` (
26    `idregistros` int(5) unsigned zerofill NOT NULL AUTO_INCREMENT,
27    `nome_completo` varchar(255) NOT NULL,
28    `telefone` bigint NOT NULL,
29    `email` varchar(255) NOT NULL,
30    PRIMARY KEY (`idregistros`),
31    UNIQUE KEY `idregistros_UNIQUE` (`idregistros`),
32    UNIQUE KEY `telefone_UNIQUE` (`telefone`)
33  ) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb3;
34  /*!40101 SET character_set_client = @saved_cs_client */;
35
36  --
37  -- Dumping data for table `cadastros`
38  --
39
40  LOCK TABLES `cadastros` WRITE;
41  /*!40000 ALTER TABLE `cadastros` DISABLE KEYS */;
42  /*!40000 ALTER TABLE `cadastros` ENABLE KEYS */;
43  UNLOCK TABLES;
44  /*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
45
```

```
45
46  /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
47  /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
48  /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
49  /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
50  /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
51  /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
52  /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
53
54  -- Dump completed on 2022-12-04 13:59:48
55
```

```
1   -- MySQL dump 10.13  Distrib 8.0.31, for Win64 (x86_64)
2   --
3   -- Host: localhost    Database: bdgoldcoin
4   -- ------------------------------------------------------
5   -- Server version 8.0.31
6
7   /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
8   /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
9   /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
10  /*!50503 SET NAMES utf8 */;
11  /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
12  /*!40103 SET TIME_ZONE='+00:00' */;
13  /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
14  /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
15  /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
16  /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
17
18  --
19  -- Table structure for table `cadastros`
20  --
```

```
21
22  DROP TABLE IF EXISTS `cadastros`;
23  /*!40101 SET @saved_cs_client     = @@character_set_client */;
24  /*!50503 SET character_set_client = utf8mb4 */;
25  CREATE TABLE `cadastros` (
26    `idregistros` int(5) unsigned zerofill NOT NULL AUTO_INCREMENT,
27    `nome_completo` varchar(255) NOT NULL,
28    `telefone` bigint NOT NULL,
29    `email` varchar(255) NOT NULL,
30    PRIMARY KEY (`idregistros`),
31    UNIQUE KEY `idregistros_UNIQUE` (`idregistros`),
32    UNIQUE KEY `telefone_UNIQUE` (`telefone`)
33  ) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb3;
34  /*!40101 SET character_set_client = @saved_cs_client */;
35
36  --
37  -- Dumping data for table `cadastros`
38  --
39
40  LOCK TABLES `cadastros` WRITE;
41  /*!40000 ALTER TABLE `cadastros` DISABLE KEYS */;
42  /*!40000 ALTER TABLE `cadastros` ENABLE KEYS */;
43  UNLOCK TABLES;
44  /*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
45
```

```
46    /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
47    /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
48    /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
49    /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
50    /*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
51    /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
52    /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
53
54    -- Dump completed on 2022-12-04 14:00:51
55
```

resources

```
-- MySQL dump 10.13  Distrib 8.0.31, for Win64 (x86_64)
--
-- Host: localhost    Database: bdgoldcoin
-- ------------------------------------------------------
-- Server version   8.0.31

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Dumping routines for database 'bdgoldcoin'
--
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2022-12-04 14:00:52
```

1. BackEnd

Equipe: Mateus Alencar e Thiago Mendes

O backEnd foi desenvolvido a partir de um modelo conceitual e um modelo lógico de um cadastro no banco de dados.

A ferramenta usada para configurar o banco de dados do GoldCoin foi o MySQL. Durante o processo de desenvolvimento do backEnd, foi criado também o CRUD, que foi a chave que possibilitou fazer a interação de telas entre o PyCharm e o banco de dados no MySQL e com as telas do Front. O CRUD foi responsável por receber os dados do frontEnd e enviar para o banco de dados, e, enviar as informações do banco de dados para a API.

2. FrontEnd - GoldCoin

Equipe: Isabelly Costa e Ana Madalena

Para o design da aplicação foi feito um protótipo de baixa fidelidade utilizando o site Figma, onde foram definidos, além do design e visual da aplicação, a quantidade de telas e a interação entre elas.

Para fazer as telas em formatos de GUI, foi utilizado o programa QtDesigner, onde foram feitas com base no protótipo realizado anteriormente.

A interação entre telas foi programada em linguagem Python utilizando a biblioteca PyQt5, onde, no Pycharm, os arquivos das telas (*.ui*) foram convertidos para arquivos Python (*.py*) para que pudessem ser manipulados e assim serem definidos os métodos e funcionalidades dos botões para que houvesse interação entre as telas.

- **Criação da Logo e Slogan:**

Para a criação do design da logo do GoldCoin Game foi feita uma rápida pesquisa para definir o modelo que mais faria sentido para o estilo do jogo.

A partir do modelo definido, foi feito um esboço do desenho que seria a marca central da logo. O desenho foi criado no aplicativo Painter e exportado para que fosse editado e integrado aos demais elementos no site Canva.

O slogan “Descubra aqui o seu valor” foi pensado a partir da ideia de que o usuário estará dando start no game para descobrir o valor de cotação da moeda desejada. Ambos os pontos, design da logo e slogan, foram direcionados e aprovados pelo Master.

3. Game - Gold Coin

Equipe: Felipi Rodrigues e Daniel Siqueira

O jogo surgiu a partir da ideia de criar uma aplicação que enviasse cotações diárias de moedas no mercado. Assim, a equipe de Game tinha como objetivo criar um jogo que servia de ponte para a etapa que antecede o cadastro do usuário no sistema de cotação de moedas.

Para fazer o cadastro o usuário precisa passar pelo game, um jogo de habilidade de curta duração. O objetivo do jogo é pegar pequenas moedas na tela utilizando uma cobrinha. E é necessário que o player ou jogador pegue dez destas moedas para conseguir prosseguir com o seu cadastro. E, ao vencer o jogo, o usuário tem acesso para se cadastrar.

Toda a estrutura do jogo foi escrita em linguagem python, com o auxílio da biblioteca pygame para êxito no desenvolvimento do mesmo.

Interação:

O player interage com a cobrinha usando as teclas (ASDW). Utilizando estas teclas o player pode obter os movimentos desejados para a cobrinha alcançar seu objetivo.
W, a cobrinha move-se para cima
S, a cobrinha move-se para baixo
A, a cobrinha move-se à esquerda
D, a cobrinha move-se à direita.

4. BackEnd - (API)

Equipe: Fernando Santos e Jonathas Rodrigues

A criação da API do projeto tinha como objetivo captar a cotação das moedas e integrá-las ao sistema de banco de dados. A API foi criada utilizando algumas bibliotecas que são necessárias para que a aplicação funcione. Essas bibliotecas foram instaladas na IDE do Pycharm, sendo elas: Requests, Selenium, Pandas, Pywhatkit e Pyautogui.

Documentação de Testes

1. BackEnd

Equipe: Mateus Alencar e Thiago Mendes

Problemas encontrados:

- **Dificuldades para receber e armazenar os dados recebidos**

  Para utilizar os dados fornecidos pelo usuário é preciso armazená-los para posteriormente manipulá-los no sistema. Durante a programação, houve dificuldades em encontrar um meio de armazenar essas informações.

- **Como foi resolvido:**

  Para resolver o problema, foi preciso criar metódos que recebessem os dados escritos pelo usuario após o clique do botão “Prosseguir” na janela da aplicação, que também precisou receber atributos que identificassem o clime, para que assim pudesse armazenar os dados.

2. FrontEnd - Gold Coins

Equipe: Isabelly Costa e Ana Madalena

Problemas encontrados:

- **Falha na funcionalidade de botões “voltar” e “sair”**

  Os botões para voltar e sair de determinada janela não estavam executando sua determinada função. Apesar de definir métodos para os botões durante a programação na linguagem Python, o código apresenta erros, e a aplicação fecha inesperadamente.

- **Como foi resolvido:**

  No software QtDesigner foram definidas ações para os botões, para que pudessem interagir diretamente com as janelas nos quais estavam inseridos, as fechando.

- **Falha na leitura e importação do arquivo .qrc**

  Ao importar os arquivos do Qt Designer em formato *.ui* para arquivos Python, eles possuíam importações de um arquivo *.qrc*, que seria a fonte das imagens que o Qt Designer usa para implementar as imagens nas telas. Este arquivo de importação não foi reconhecido pelo Python, logo, resultava em erro no código fazendo com que não fosse possível executá-lo.

  - **Como foi resolvido:**

    Como não foi reconhecido, foi necessário descartar as importações do arquivo *.qrc* do código para que assim fosse possível executá-lo.

- **Falta de imagens na aplicação**

  Sem o arquivo de imagens (citado acima) ao executar o código ele não possuía as imagens implementadas no Qt Designer.

  - **Como foi resolvido:**

    Foi necessário converter o arquivo *.qrc* para um arquivo Python(*.py*) para que assim pudesse realizar a importação desse mesmo para a *main*, para que assim as imagens aparecessem na aplicação.

- **A aplicação não abre pygame (NÃO RESOLVIDO)**

  Durante a execução da aplicação, esta deveria conter um botão que abriria um arquivo *pygame*, porém, ao tentar implementar esta função, o arquivo fechava inesperadamente ou simplesmente não abria o jogo.

3. GAME

Equipe: Felipi Rodrigues e Daniel Siqueira

Problemas encontrados:

- **Falha ao rodar o jogo:**

O game, embora atualizado, não estava rodando a versão atualizada. A tela do jogo continuava como nas primeiras fases do desenvolvimento, branca e a cobrinha era apenas um quadrado.

- **Como foi resolvido:**

  O problema do design da tela foi resolvido ao atualizar as bibliotecas, para que o jogo pudesse ser rodado na versão final.

- **Falha na sintaxe e código corrompido:**

Apesar de terem sido escritos cuidadosamente, devido ao grande número de códigos que compõem o game, ao tentar rodar o programa, foi bem comum aparecer as mensagens de syntax error ou erro de sintaxe.

- **Como foi resolvido:**

  Como o erro era na sintaxe, para resolver esse problema, foi necessário revisar por completo as várias linhas de códigos e algumas partes tiveram que ser reescritas, a fim de fazer o programa rodar sem erro.

- **Erro na importação de arquivos**

Uma das falhas, entre várias, foi na de importação de arquivos, que precisavam ter o formato da imagem no final da string, sendo .png .jpg ou .jpeg.

- **Como foi resolvido:**

  Para solucionar o problema do programa não reconhecer os arquivos, foi usado o comando: imagens/nome do arquivo.formato dentro de uma função do pygame chamada: pygame.image.load (), o comando que citado anteriormente dentro dos parênteses do segundo comando. Assim foi possível importar a imagem de ficheiros não alocados.

- **Erro na interligação de Bibliotecas**

Outra falha foi a interligação das bibliotecas, que estavam dando erro apesar de o código estar escrito corretamente.

- **Como foi resolvido:**

  Para resolver esse problema, foi usado um dos principais comandos para chamar a biblioteca "pygame" usando o import. Assim, foi possível importar várias partes do código por inteiro. Especificando como "import pygame" para a importação do pygame no programa ou usando o import também para importar imagens ou áudios.

4. BackEnd - (API)

Equipe: Fernando Santos e Jonathas Rodrigues

Problemas encontrados:

- **Aprendizado e desenvolvimento:**

  Tem sido desafiador fazer parte do projeto, pois é necessário buscar informações sobre as bibliotecas certas para instalar e assim rodar a API.

  - **Como foi resolvido:**

    Com pesquisas e estudos, aprendemos para que serve a API e a sua função e conseguimos ganhar um certo domínio sobre o assunto e, de certa forma, alcançar o objetivo inicial.

- **Problema ao Abrir o navegador (NÃO RESOLVIDO)**

  Durante a execução do API, o navegador abre mas não fixa, e, apesar dos esforços da equipe, continua fechando depois de alguns segundos.

5. Integração FrontEnd e BackEnd

Equipe: Isabelly Costa e Mateus Alencar

Documentação Gold Coins - Integração FrontEnd e BackEnd

Para a integração entre BackEnd e FrontEnd, os códigos foram feitos em Python utilizando a ferramenta Pycharm.

O objetivo da integração era fazer com que os dados recebidos nas janelas da aplicação (FrontEnd) fossem lidas pelo BackEnd, sendo processadas, repassadas para o Banco de Dados e então sendo utilizadas para que as mensagens fossem enviadas de forma personalizada para o usuário.

Problemas encontrados:

- **Dificuldades para receber e armazenar os dados recebidos**

Para utilizarmos os dados fornecidos pelo usuário é preciso que possamos armazená-los para posteriormente manipulá-los no sistema. Durante a programação, foram encontradas dificuldades em encontrar um meio de armazenar essas informações.

- **Como foi resolvido:**

Foi preciso criar métodos que recebessem os dados preenchidos após o clique no botão “Prosseguir” na janela da aplicação, o qual também precisou receber atributos que identificassem o clique, para que assim pudesse armazenar os dados nas variáveis nos métodos criados.

- **Falha na utilização de variáveis em diferentes arquivos (NÃO SOLUCIONADO)**

Os dados dos usuários recebidos pelo FrontEnd são armazenados em variáveis, para que assim possam ser manipulados e processados da forma desejada. Porém, como a aplicação possui diversos arquivos, as variáveis ficam armazenadas em cada um deles, sendo necessário fazer a importação para os arquivos da parte de BackEnd e utilizá-los no CRUD, entretanto, as importações não parecem funcionar e o BackEnd não recebe os dados.

1. Link do repositório Git Projeto:

<<https://github.com/goldcoinsenac/goldcoin.git>>