



UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO

Jogo de Damas

Daniel Pessoa Máximo

Saulo Roberto dos Santos

Maceió/AL
2023



UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO

Jogo de Damas

Relatório do projeto

Este relatório tem como objetivo apresentar a aplicação desenvolvida para a matéria de Engenharia de Software, lecionada pelo professor Dr. Arturo Hernández Domínguez.

Sumário

1. Enunciado	4
2. Product Backlog	5
3. Sprint Backlog	6
4. Diagrama de Casos de Uso	7
5. Arquitetura do Sistema	9
6. Diagrama de Classes	10
7. Tecnologias utilizadas	10
8. Funcionamento do Sistema	11
9. Vídeo do Jogo	17
10. Github	18
11. Testes de Unidade	18

1. Enunciado:

Este relatório descreve o desenvolvimento de um jogo de Damas utilizando o método ágil Scrum, Python e Pygame. O objetivo do jogo é oferecer uma experiência desafiadora e divertida aos jogadores, permitindo jogar contra um adversário controlado pelo computador ou contra outro jogador. O jogo também inclui telas para recuperação de senha, utilizando o serviço de envio de e-mails do SendGrid para enviar o código de verificação.

O projeto foi concluído em 6 semanas, seguindo o Scrum. Durante o desenvolvimento, a equipe focou na melhoria da jogabilidade, lógica do jogo e criação de uma interface intuitiva. O resultado final atingiu as expectativas em termos de qualidade e funcionalidade.

Os jogadores podem criar um novo usuário e fazer login com suas credenciais. O sistema lida com problemas como nomes de usuário ou e-mails já existentes, além de identificar erros de login, como senhas incorretas ou usuários inexistentes.

Após o login, o usuário é direcionado ao menu do jogo, com opções para jogar contra a inteligência artificial baseada no algoritmo Minimax ou contra outro jogador localmente. Durante as partidas, é possível controlar as peças, considerando a captura de múltiplas peças em uma jogada.

Adicionalmente, foram implementadas telas de recuperação de senha, onde os usuários podem solicitar a recuperação por meio do envio de um código de verificação. O SendGrid é utilizado para o envio seguro do código de verificação para o endereço de e-mail associado à conta do usuário.

Essas adições melhoraram a segurança e a usabilidade do jogo, permitindo aos usuários recuperar suas senhas de forma eficiente. A equipe utilizou as melhores práticas do Scrum e integrou tecnologias relevantes para entregar um jogo de Damas completo e satisfatório.

2. Product BackLog:

2.1. Sprint 1

ID	História	Prioridade	Estimativa
1	Criar tabuleiro	Muito alta	20 horas
2	Validação dos movimentos	Muito alta	10 horas
3	Interface e interação com o tabuleiro	Muito alta	14 horas

2.2. Sprint 2

ID	História	Prioridade	Estimativa
4	Modo contra jogador	Muito alta	22 horas
5	Modo contra o computador	Alta	12 horas
6	Criar tela de Menu	Alta	7 horas

2.3. Sprint 3

ID	História	Prioridade	Estimativa
7	Interface gráfica para Login e Cadastro	Alta	12 horas
8	Cadastrar Usuário	Média	7 horas
9	Autenticar o Usuário	Média	14 horas
8	Recuperar senha	Baixa	6 horas

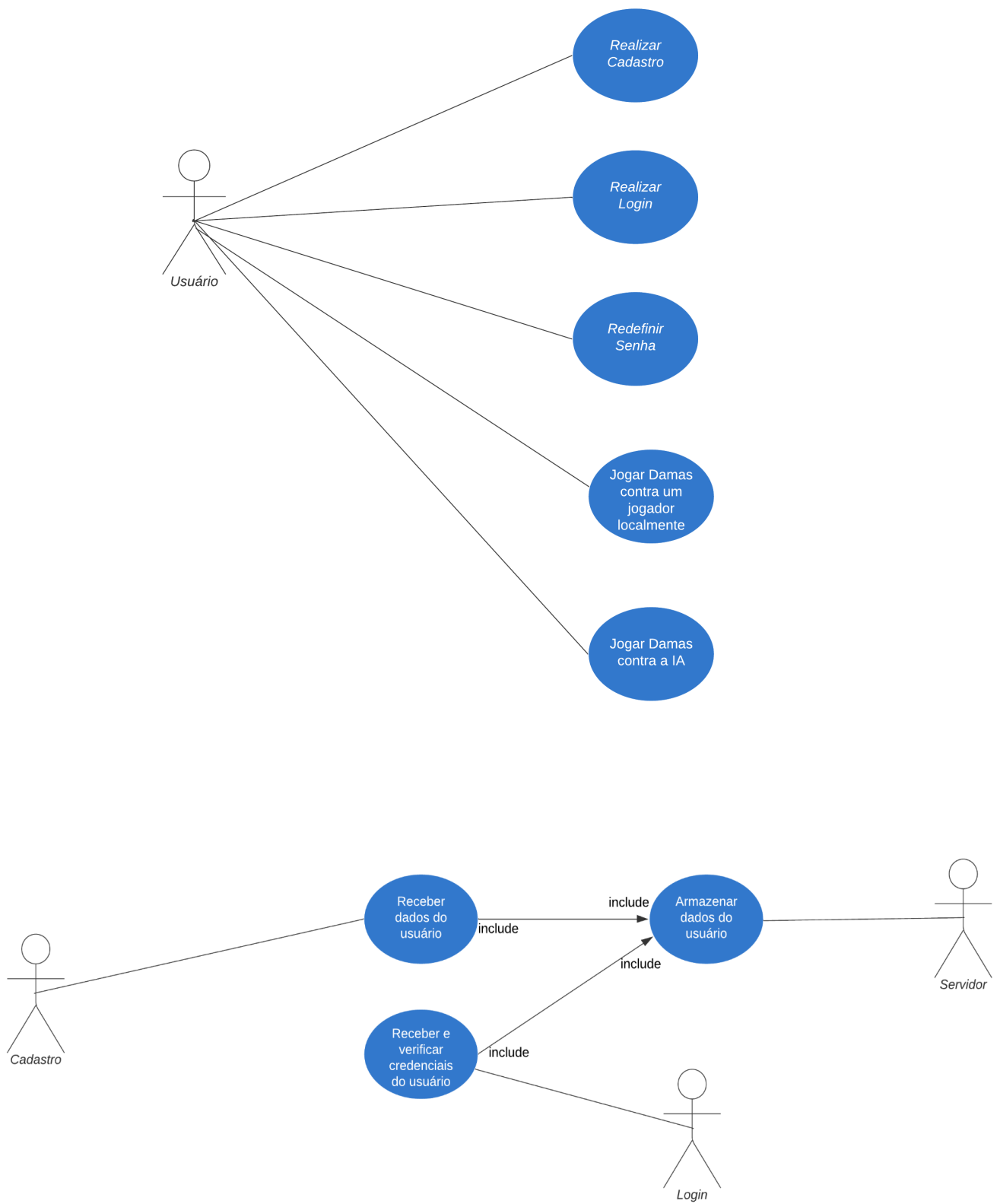
3. Sprint BackLog:

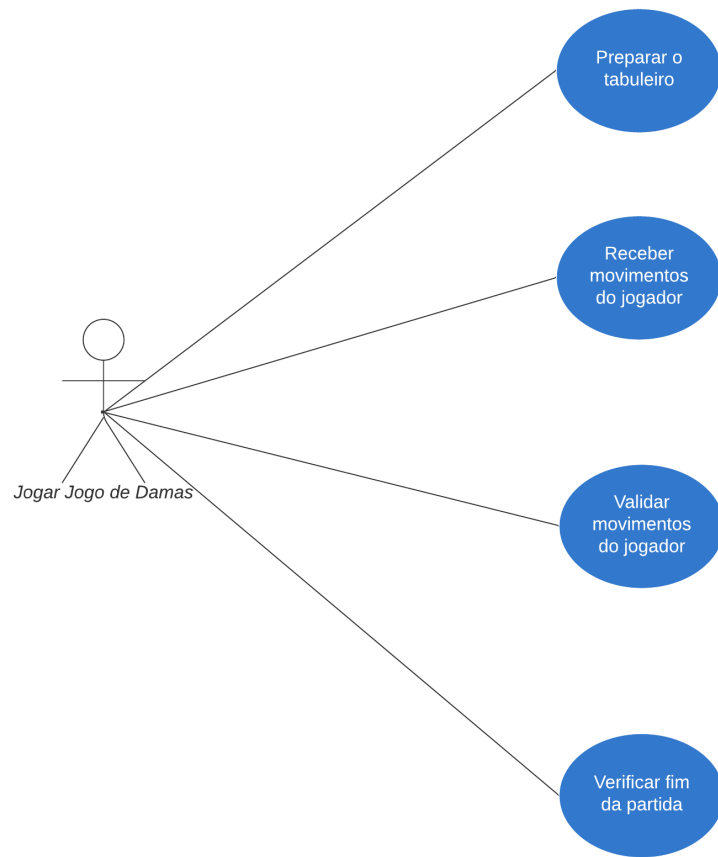
ID da Sprint	ID	Funcionalidades	Nível
Sprint 1	1	Criar tabuleiro	Alta
	2	Validação dos movimentos	
	3	Interface e interação com o tabuleiro	

ID da Sprint	ID	Funcionalidades	Nível
Sprint 2	4	Modo contra jogador	Médio
	5	Modo contra o computador	
	6	Criar tela de Menu	

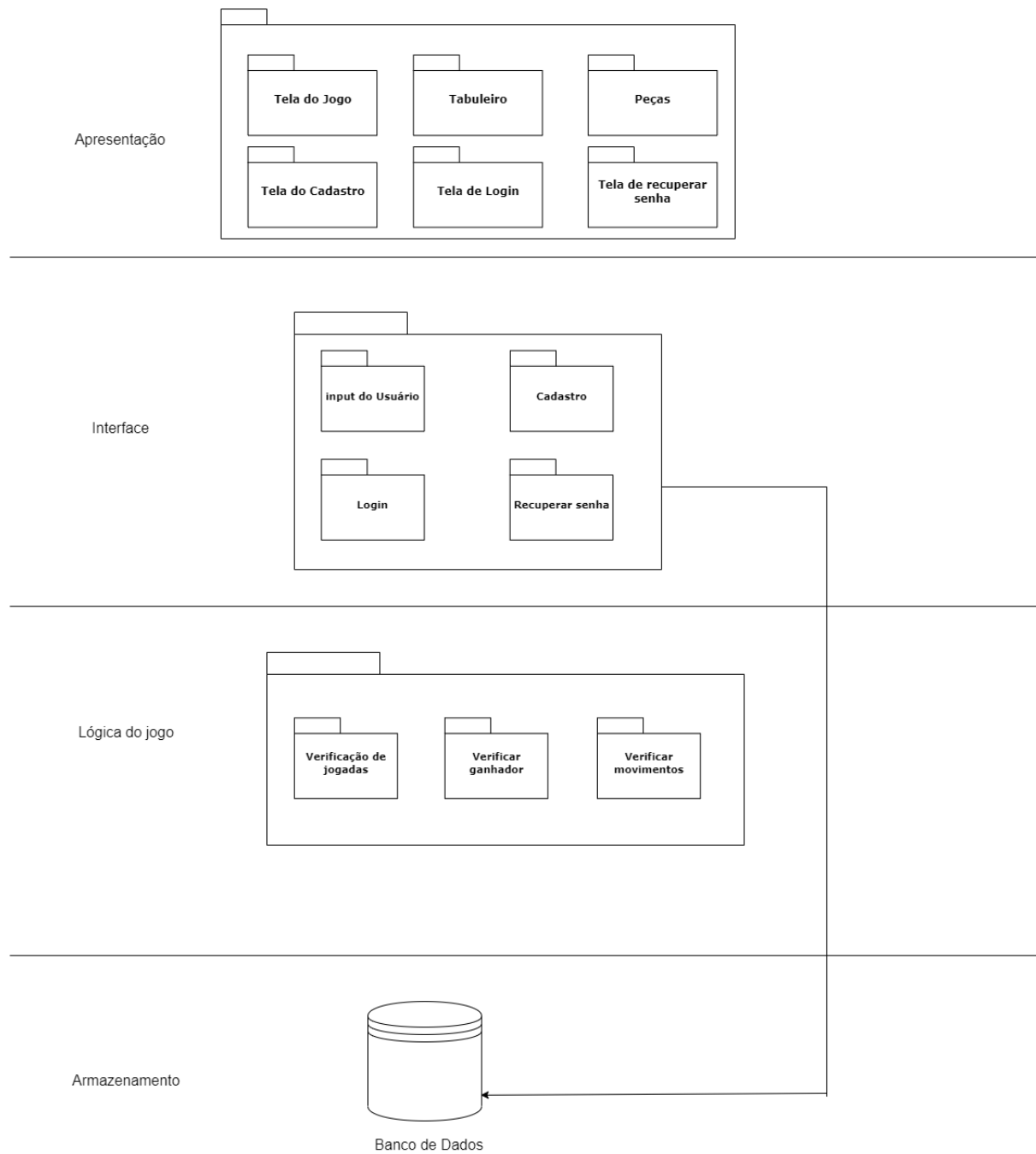
ID da Sprint	ID	Funcionalidades	Nível
Sprint 3	7	Criar tela principal	Médio
	8	Cadastrar Usuário	
	9	Validar autenticação	
	10	Recuperar senha	

4. Diagramas de Caso de Uso:

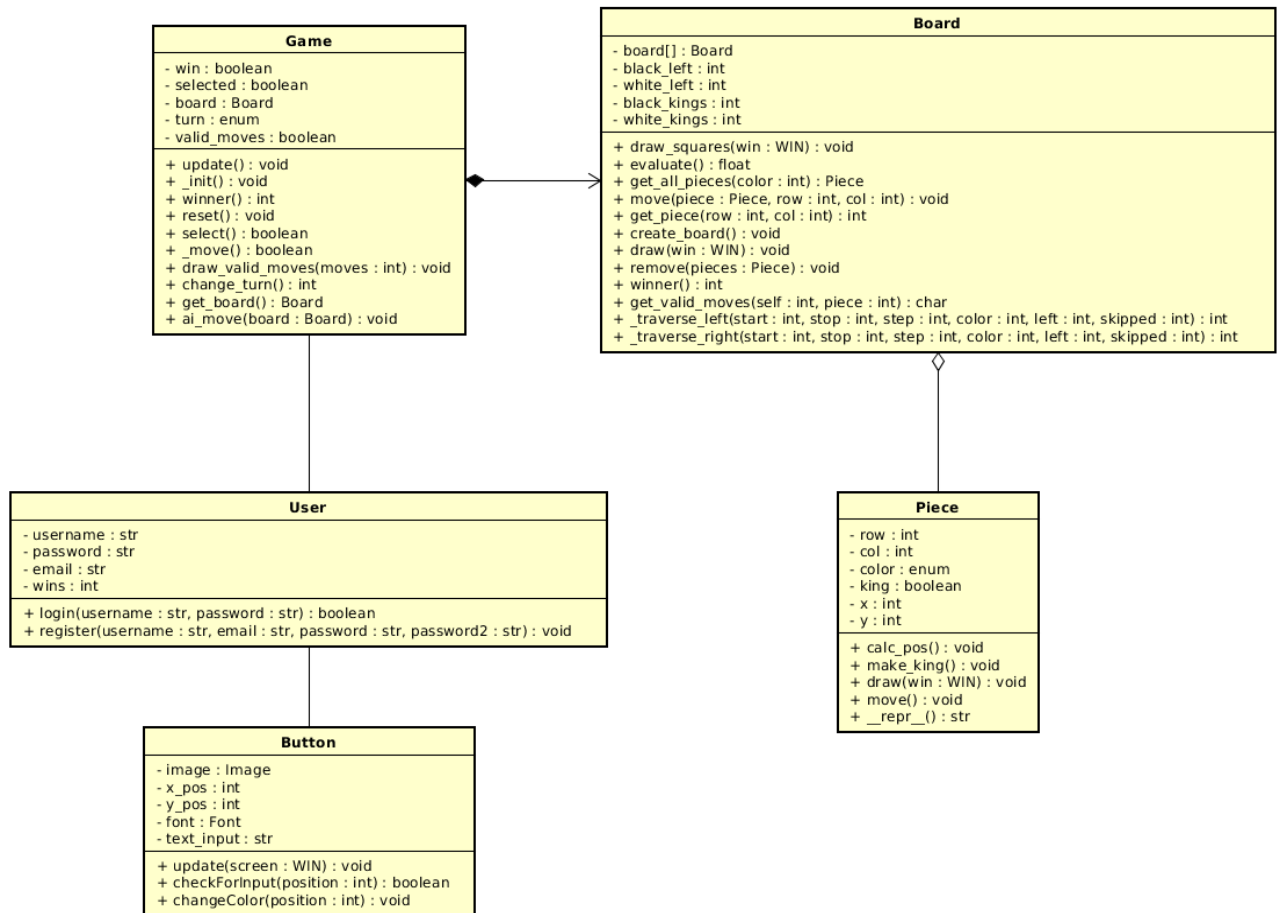




5. Arquitetura do Sistema:



6. Diagrama de Classes:



7. Tecnologias Utilizadas:

No desenvolvimento desse projeto, foram utilizadas diversas tecnologias para a criação de uma experiência de jogo completa. O jogo foi criado utilizando a linguagem de programação Python, juntamente com a biblioteca Pygame, que é amplamente utilizada na criação de jogos em 2D. Além disso, o design das telas de login, cadastro e recuperação de senha foram feitos utilizando a ferramenta QtDesigner, que permite a criação de interfaces gráficas de forma visual e intuitiva.


Para armazenar as informações dos usuários, foi utilizado o banco de dados SQLite, que é um banco de dados relacional embutido nas bibliotecas do Python. Essa escolha garantiu a segurança e a integridade das informações dos usuários cadastrados no jogo.

Além das tecnologias mencionadas, foi integrado o serviço de envio de e-mails do SendGrid para a funcionalidade de recuperação de senha. Quando um usuário solicita a recuperação de senha, um código de verificação é gerado e enviado para o endereço de e-mail associado à sua conta, utilizando a plataforma do SendGrid. Essa integração permitiu a entrega confiável dos e-mails de verificação, garantindo uma experiência tranquila para os usuários na recuperação de suas senhas.

Por fim, o código do projeto foi armazenado no repositório do Github, uma plataforma de hospedagem de código-fonte e colaboração em equipe, permitindo o controle de versão do código, a colaboração de vários desenvolvedores e a facilidade de acesso e compartilhamento do projeto.

8. Funcionamento do Sistema

8.1. Tela de Login:





A tela de login possui um fundo azul sólido. No topo, há dois campos de entrada de texto brancos com bordas arredondadas. O primeiro campo é precedido por um ícone de perfil de usuário e contém o placeholder 'Digite seu username'. O segundo campo é precedido por um ícone de cadeado e contém o placeholder 'Digite sua senha'. Abaixo dos campos, há um botão ciano com o texto 'Login' em preto. Logo abaixo do botão, o texto 'Recuperar a senha' aparece em uma cor mais clara. Na base da tela, o texto 'Não é Cadastrado' está à esquerda, e um botão laranja com o texto 'Cadastrar' em preto está à direita.


8.2. Tela de Cadastro:

[Voltar](#)

Preencha os campos abaixo para se cadastrar










Cadastrar

[Voltar](#)

Preencha os campos abaixo para se cadastrar













Cadastrar

MainWindow

Preencha todos os campos!

 saulo 

 srbt@gmail.com 

 srbt2023 


Repita a senha

Cadastrar

8.3. Tela de recuperar senha:

[Voltar](#)

iremos um código de verificação para o seu en

 daniel

 dpm@ic.ufal.br

Enviar email

Voltar

Confira seu email e digite abaixo o código recebido

Digite seu código

Enviar

8.4. Login:



danie



.....

Usuário ou senha incorretos!

Login

Recuperar a senha

Não é Cadastrado

Cadastrar



A login form on a blue background. It features two input fields: the first for a username with a person icon and the text 'daniel', and the second for a password with a lock icon and masked characters '.....'. Below the fields is a red 'Login' button. Underneath the button is a link 'Recuperar a senha'. At the bottom left is the text 'Não é Cadastrado', and at the bottom right is an orange 'Cadastrar' button.

Login

[Recuperar a senha](#)

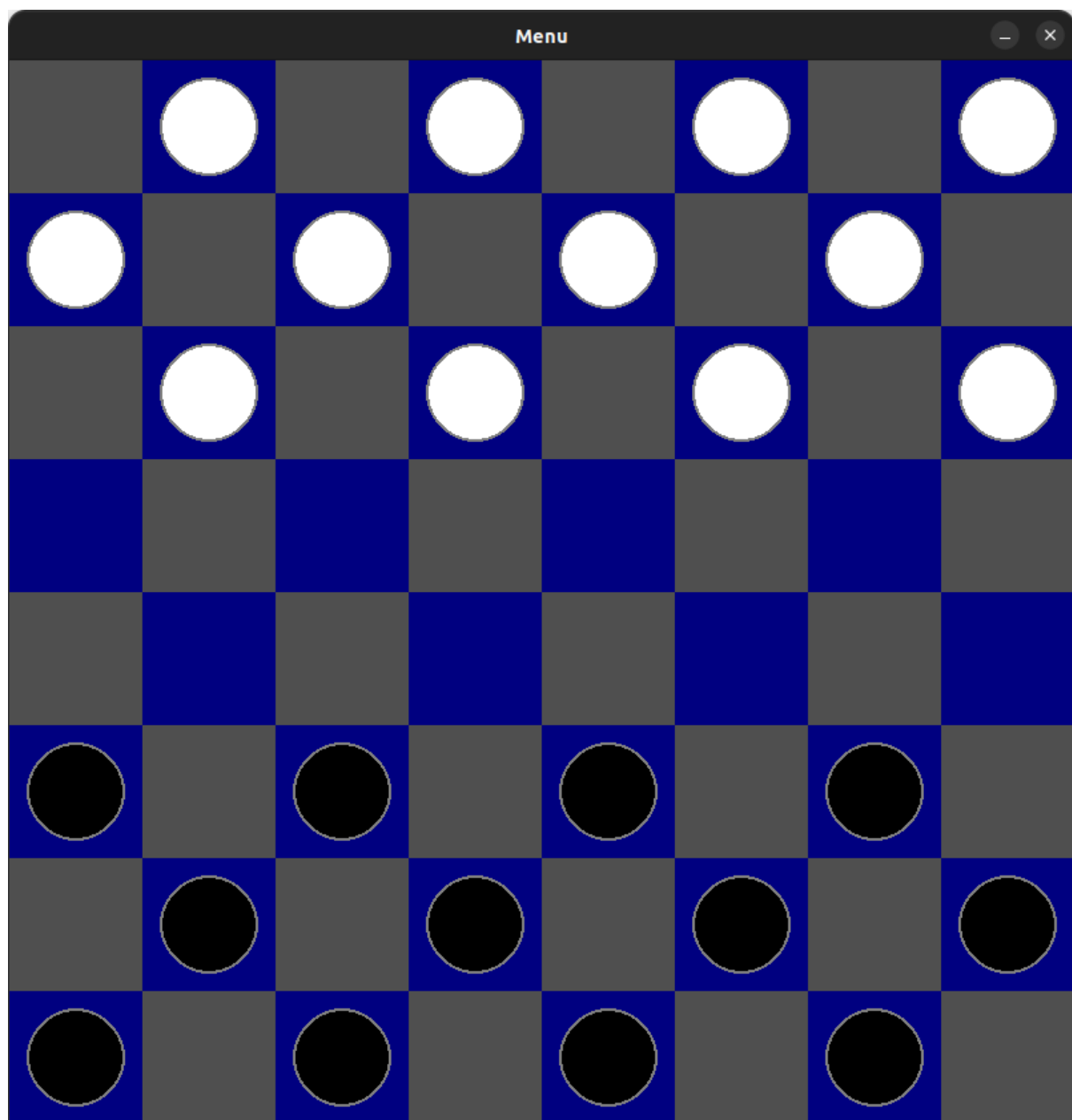
Não é Cadastrado

Cadastrar

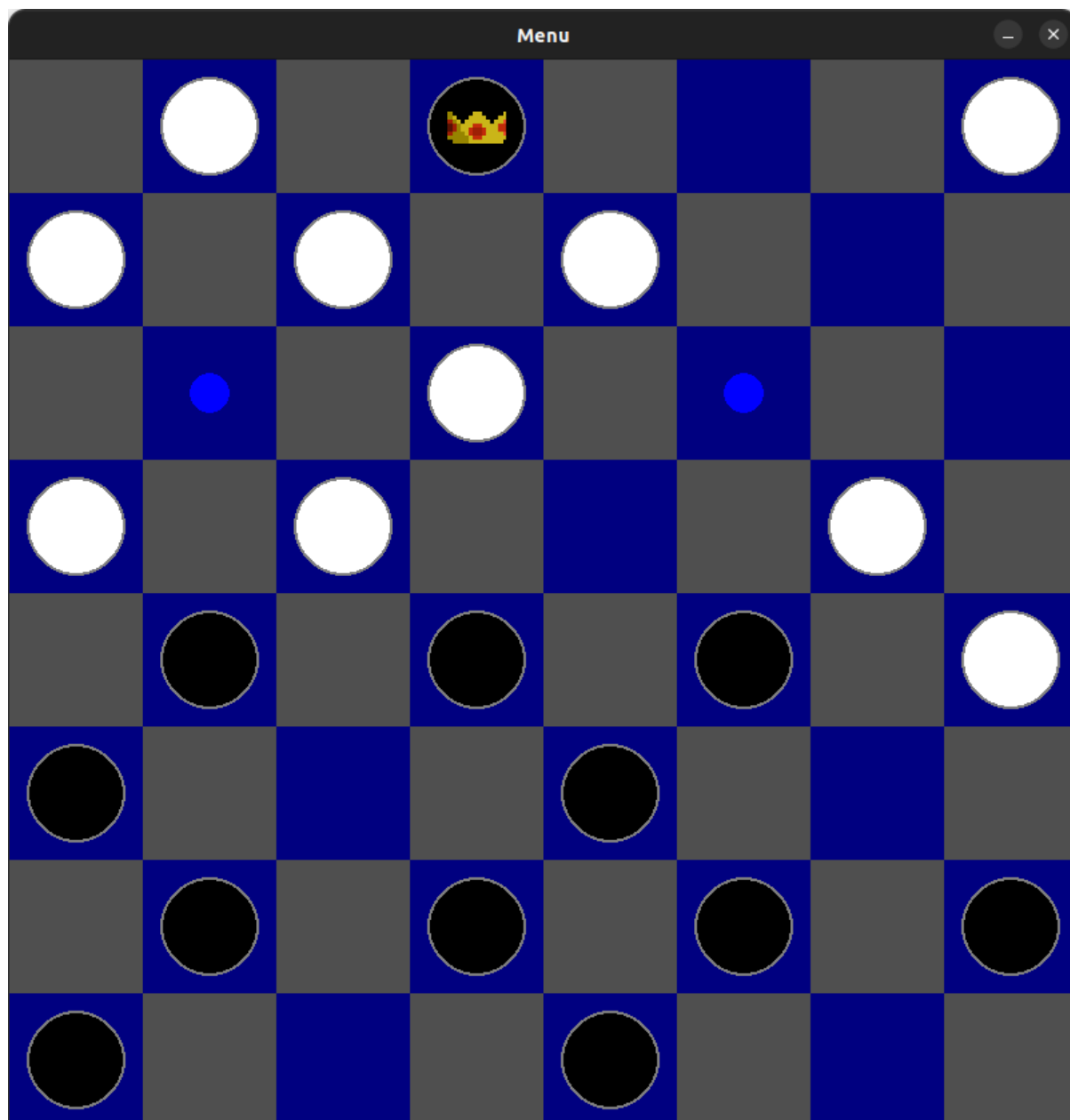
8.5. Tela do Menu Principal:



8.6. Tela do Jogo:



8.7. Peça Rainha:



9. Vídeo do jogo:

<https://youtu.be/pRFTf6gtaQs>

10. Github:

<https://github.com/saulolv/Checkers-SoftwareEng>

11. Testes de unidade

```
import unittest
import sqlite3
import os
from user import User

class UserTests(unittest.TestCase):
    def setUp(self):
        self.db_file = "test.db"
        self.user = User(self.db_file)
        self.conn = sqlite3.connect(self.db_file)

    def tearDown(self):
        self.conn.close()
        os.remove(self.db_file)

    def test_login(self):
        self.conn.execute("INSERT INTO users VALUES ('testuser', 'test@example.com', 'password')")
        self.conn.commit()

        self.assertEqual(self.user.login("testuser", "password"), True)
        self.assertEqual(self.user.login("testuser", "wrongpassword"), False)
        self.assertEqual(self.user.login("wronguser", "password"), False)
        self.assertEqual(self.user.login("", ""), "Preencha todos os campos!")

    def test_register_success(self):
        result = self.user.register('newuser', 'new@example.com', 'password', 'password')
        self.assertEqual(result, 'Usuário cadastrado com sucesso!')

        cursor = self.conn.cursor()
        cursor.execute("SELECT * FROM users WHERE username = ?", ('newuser',))
        user = cursor.fetchone()
        self.assertIsNotNone(user)

    def test_register_existing_username(self):
        self.conn.execute("INSERT INTO users VALUES ('existinguser', 'existing@example.com', 'password')")
        self.conn.commit()

        result = self.user.register('existinguser', 'new@example.com', 'password', 'password')
        self.assertEqual(result, 'o Username já está sendo usado!')

    def test_register_existing_email(self):
        self.conn.execute("INSERT INTO users VALUES ('existinguser', 'existing@example.com', 'password')")
        self.conn.commit()

        result = self.user.register('newuser', 'existing@example.com', 'password', 'password')
        self.assertEqual(result, 'Este email já está sendo usado, tente outro!')

    def test_register_password_mismatch(self):
        result = self.user.register('newuser', 'new@example.com', 'password', 'wrong_password')
        self.assertEqual(result, 'As senhas não coincidem!')

    def test_register_empty_fields(self):
        result = self.user.register('', '', '', '')
        self.assertEqual(result, 'Preencha todos os campos!')

    def test_change_password_success(self):
        self.conn.execute("INSERT INTO users VALUES ('testuser', 'test@example.com', 'password')")
        self.conn.commit()

        self.user.change_password('testuser', 'new_password')

        cursor = self.conn.cursor()
        cursor.execute("SELECT * FROM users WHERE username = ?", ('testuser',))
        user = cursor.fetchone()
        self.assertEqual(user[2], 'new_password')

    def test_check_user_existing(self):
        self.conn.execute("INSERT INTO users VALUES ('testuser', 'test@example.com', 'password')")
        self.conn.commit()

        result = self.user.check_user('testuser', 'test@example.com')
        self.assertTrue(result)

    def test_check_user_non_existing(self):
        result = self.user.check_user('nonexistinguser', 'nonexisting@example.com')
        self.assertFalse(result)

    def test_check_user_empty_fields(self):
        result = self.user.check_user('', '')
        self.assertEqual(result, 'Preencha todos os campos!')

if __name__ == '__main__':
    unittest.main()
```

.....

Ran 10 tests in 0.172s

OK