МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

КЛИЕНТСКАЯ ЧАСТЬ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ

по дисциплине

«Программные средства разработки интеллектуальных систем»

Студент гр. 1308, _____ Мельник Д. А.

Студент гр. 1308, _____ Томилов Д. Д.

Студент гр. 1308, _____ Лепов А. В.

Санкт-Петербург

2022

СОДЕРЖАНИЕ

1. ПОСТАНОВКА ЗАДАЧИ

В качестве задачи на данном этапе необходимо реализовать клиентскую часть, которая даст пользователю (клиенту) рабочий, простой в использовании десктоп интерфейс для игры в шахматы.

2. ДИЗАЙН ИНТЕРФЕЙСА КЛИЕНТСКОЙ ЧАСТИ И ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ОПЫТА

Описание пользовательского опыта:

- При запуске программы пользователь входит в систему в качестве гостя.
- Ему необходимо войти в систему, чтобы изменить роль на пользователя.

На рисунках 1-6 представлен дизайн интерфейса клиентской части.

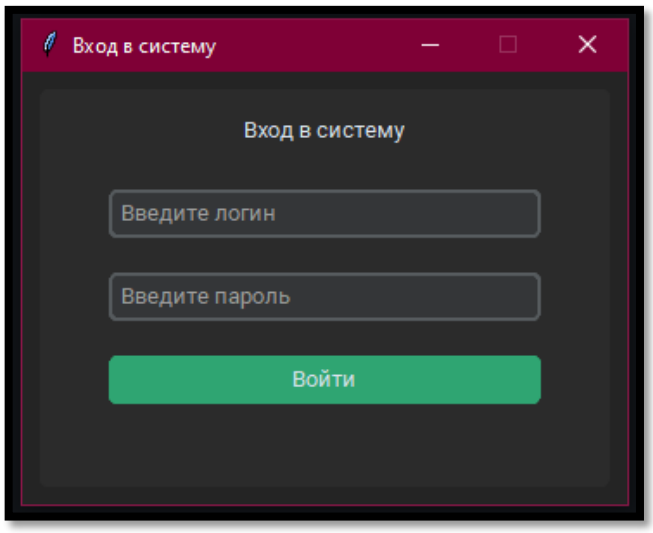

Рис. 1. Интерфейс входа в систему

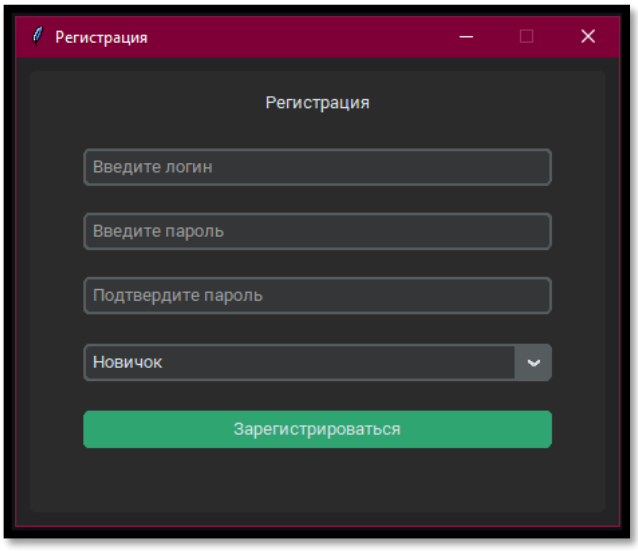

Рис. 2. Интерфейс входа в систему

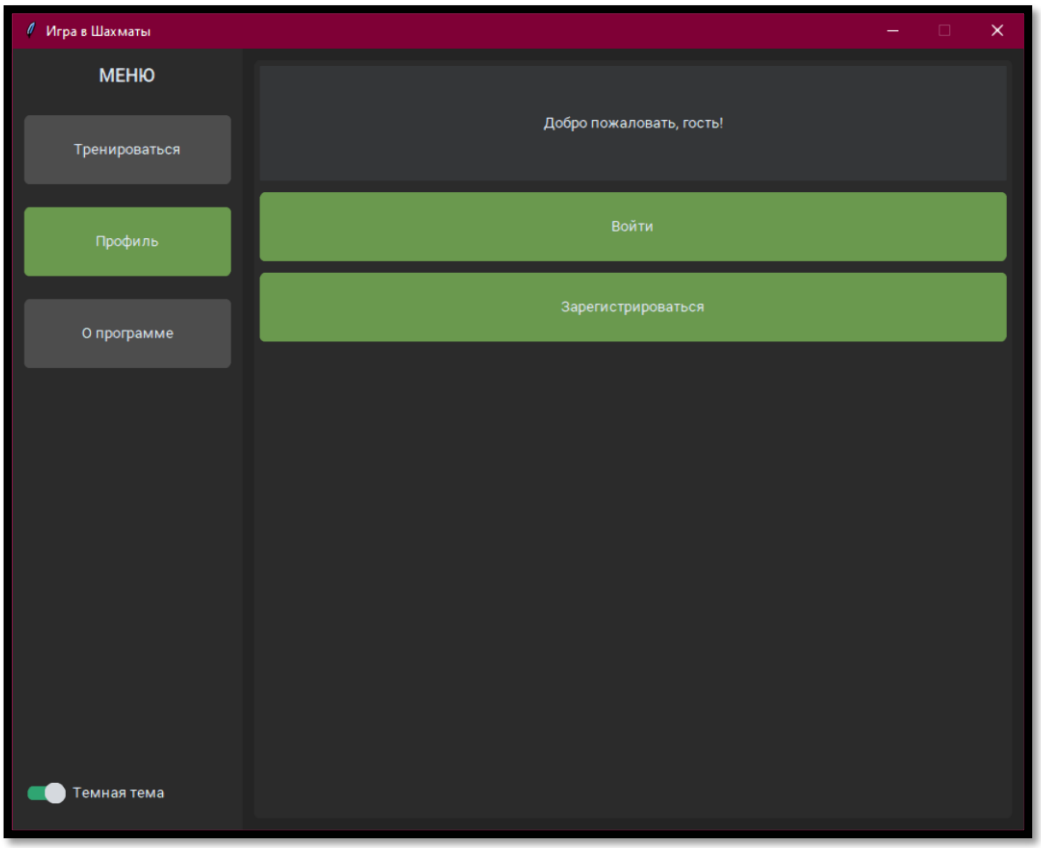

Рис. 3. Интерфейс входа в систему

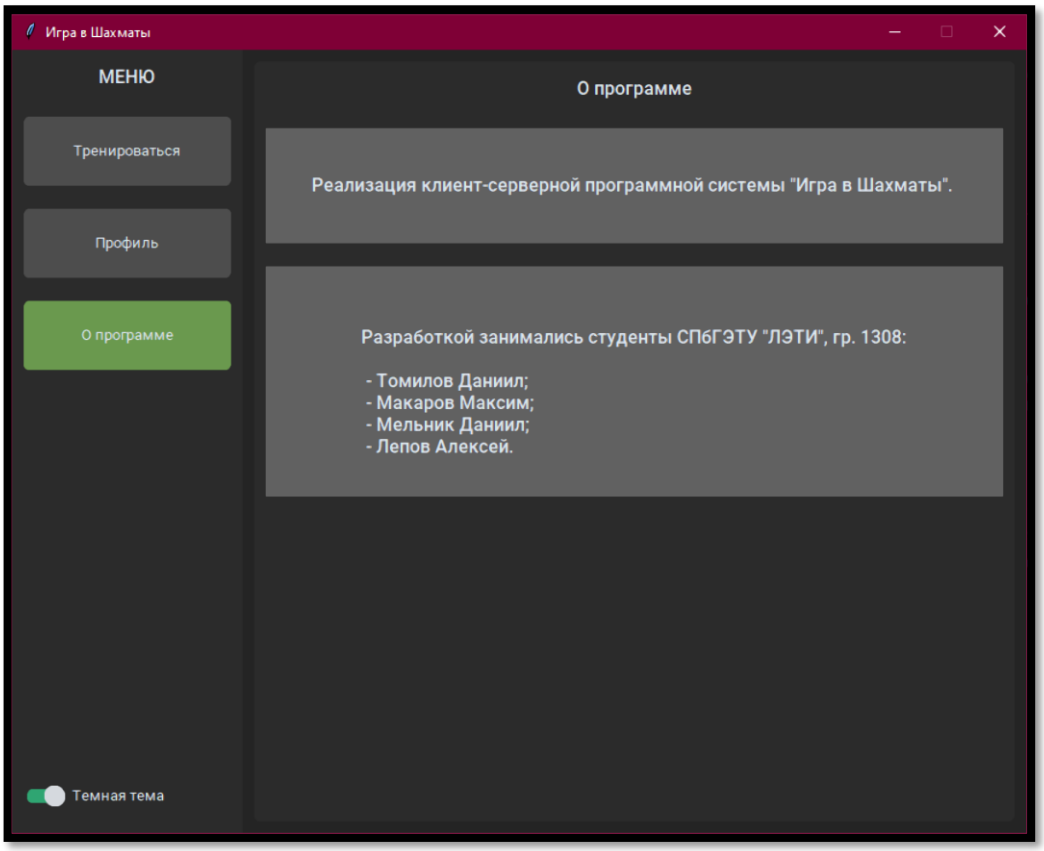

Рис. 4. Интерфейс входа в систему

Рис. 5. Интерфейс входа в систему

Рис. 6. Интерфейс входа в систему

3. ВЫБРАННЫЕ ТЕХНОЛОГИИ

Выбранные технологии:

- Python 3;
- TKinter;
- CustomTkinter;
- IDE: VS Code;
- IDE: PyCharm.

4. РЕАЛИЗАЦИЯ КЛИЕНТСКОЙ ЧАСТИ И СВЯЗЬ ЕЁ С СЕРВЕРНОЙ ЧАСТЬЮ

Технологии серверной части приложения:

При создании серверной части приложения была использованы средства библиотеки socket.

Для привязки сокета к адресу использован метод bind, принимающий присваемый ip и порт в нём.

Для получения сообщения от определённого (по адресу) клиента используется метод recvfrom(), принимающий на вход максимальный объём информации для получения, возвращающий данные и адрес, с которого они были получены

Для отправки сообщений использован метов sendto(), принимающий данные и адрес принимающей стороны.

Технологии клиентской части приложения:

При создании клиентской части приложения была использованы средства библиотек socket и threading.

Для отправки сообщений использован метов sendto(), принимающий данные и адрес принимающей стороны.

Для приёма сообщения был использован метод recv(), принимающий на вход максимальный объём информации, возможный к приёму.

Для корректности попеременной работы клиентов с сервером был реализован поток через объект Thread библиотеки Threading.

5. ТЕСТ-КЕЙСЫ

```
import unittest
import main

class TestStringMethods(unittest.TestCase):

    def test_castling_1(self):
        board = main.chessEngine.Board.from_FEN("r3k2r/8/8/8/8/8/8/R3K2R w KQkq -
0 1")
        board_arr = board.get_piece_arr()
        self.assertTrue(board.move_piece(board_arr[0][4],
main.chessEngine.Position(0,6)))

    def test_false_castling_2(self):
        board = main.chessEngine.Board.from_FEN("rrrrkrrr/8/8/8/8/8/8/RRRRKRRR w
KQkq - 0 1")
        board_arr = board.get_piece_arr()
        self.assertFalse(board.move_piece(board_arr[0][4],
main.chessEngine.Position(0,2)))

    def test_castling_3(self):
        board = main.chessEngine.Board.from_FEN("r3k2r/8/8/8/8/8/8/R3K2R b KQkq -
0 1")
        board_arr = board.get_piece_arr()
        self.assertTrue(board.move_piece(board_arr[7][4],
main.chessEngine.Position(7,6)))

    def test_false_castling_4(self):
        board = main.chessEngine.Board.from_FEN("rrrrkrrr/8/8/8/8/8/8/RRRRKRRR b
KQkq - 0 1")
        board_arr = board.get_piece_arr()
        self.assertFalse(board.move_piece(board_arr[7][4],
main.chessEngine.Position(7,2)))

    def test_piece_eat(self):
        board =
main.chessEngine.Board.from_FEN("rnbqkbnr/ppp1pppp/8/3p4/4P3/8/PPPP1PPP/RNBQKBNR
w KQkq - 0 1")
        board_arr = board.get_piece_arr()
        self.assertTrue(board.move_piece(board_arr[3][4],
main.chessEngine.Position(4,3)))

    def test_false_piece_eat(self):
        board =
main.chessEngine.Board.from_FEN("rnbqkbnr/ppp1pppp/8/3p4/4P3/8/PPPP1PPP/RNBQKBNR
w KQkq - 0 1")
```

```
        board_arr = board.get_piece_arr()
        self.assertFalse(board.move_piece(board_arr[4][3],
main.chessEngine.Position(3,4)))

    def test_queenValue(self):
        self.assertEqual(main.chessEngine.Piece.get_value("queen"), 9)

    def test_getStartWhitePownPos(self):
        self.assertEqual(main.chessEngine.Piece.get_start_position("pawn",
main.chessEngine.Color.WHITE), [main.chessEngine.Position(1, i) for i in
range(8)])

    def test_isupper(self):
        self.assertFalse((main.chessEngine.Piece.__str__(main.chessEngine.Piece(m
ain.chessEngine.Color.BLACK,"pawn",1,None,True))=="White"))

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

6. ПРИМЕР РАБОТЫ

С примерами работы можно ознакомиться, перейдя на онлайн-репозиторий проекта: https://github.com/AlexeyLepov/ClientServerChessApp