Московский государственный технический
университет им. Н.Э. Баумана

Факультет Информатика и системы управления
Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Парадигмы и конструкции языков программирования»

Отчет по проекту

Выполнили:
Студент группы ИУ5-33Б:
Бегдаш Станислав
Студент группы ИУ5-31Б:
Есакова Ольга
Подпись и дата:

Проверил:
преподаватель каф.ИУ5
Нардид А. Н.
Подпись и дата:

Москва, 2025 г.

Код программы:

```
import asyncio
import logging
from aiogram import Bot, Dispatcher, Router, F
from aiogram.filters import Command
from aiogram.types import Message, CallbackQuery, InlineKeyboardMarkup, InlineKeyboardButton
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup
from aiogram.fsm.storage.memory import MemoryStorage
import random
import string


logging.basicConfig(level=logging.INFO)

BOT_TOKEN = ""

bot = Bot(token=BOT_TOKEN)
storage = MemoryStorage()
dp = Dispatcher(storage=storage)
router = Router()


class GameStates(StatesGroup):
    waiting_for_opponent = State()
    in_game = State()

#хранилище игр
games = {}

class TicTacToeGame:
    """Класс для управления игрой в крестики-нолики"""

    def __init__(self, game_id, player1_id, player1_name):
```

```
        self.game_id = game_id
        self.player1_id = player1_id
        self.player1_name = player1_name
        self.player2_id = None
        self.player2_name = None
        self.board = [[' ' for _ in range(3)] for _ in range(3)]
        self.current_player = player1_id
        self.game_started = False
        self.winner = None
        self.message_ids = {}  # Хранит ID сообщений для каждого игрока

    def add_player2(self, player2_id, player2_name):
        """Добавить второго игрока"""
        self.player2_id = player2_id
        self.player2_name = player2_name
        self.game_started = True

    def make_move(self, player_id, row, col):
        """Сделать ход"""
        if not self.game_started:
            return False, "Игра еще не началась"

        if self.winner:
            return False, "Игра уже завершена"

        if player_id != self.current_player:
            return False, "Сейчас не ваш ход"

        if self.board[row][col] != ' ':
            return False, "Эта клетка уже занята"

        # Определяем символ игрока
        symbol = '❌' if player_id == self.player1_id else '⭕'
```

```
        self.board[row][col] = symbol

        # Проверка на победу
        if self.check_winner():
            self.winner = player_id
            return True, f"Победил {self.get_player_name(player_id)}!"

        # Проверка на ничью
        if self.is_board_full():
            self.winner = "draw"
            return True, "Ничья!"

        # Смена игрока
        self.current_player = self.player2_id if self.current_player == self.player1_id else self.player1_id
        return True, "Ход сделан"

    def check_winner(self):
        # Проверка строк
        for row in self.board:
            if row[0] == row[1] == row[2] != ' ':
                return True

        # Проверка столбцов
        for col in range(3):
            if self.board[0][col] == self.board[1][col] == self.board[2][col] != ' ':
                return True

        # Проверка диагоналей
        if self.board[0][0] == self.board[1][1] == self.board[2][2] != ' ':
            return True
        if self.board[0][2] == self.board[1][1] == self.board[2][0] != ' ':
            return True

        return False
```

```
    def is_board_full(self):
        """Проверка заполненности доски"""
        for row in self.board:
            if ' ' in row:
                return False
        return True

    def get_player_name(self, player_id):
        """Получить имя игрока"""
        if player_id == self.player1_id:
            return self.player1_name
        elif player_id == self.player2_id:
            return self.player2_name
        return "Неизвестный игрок"

    def get_board_keyboard(self):
        """Создать клавиатуру с игровым полем"""
        keyboard = []
        for i in range(3):
            row = []
            for j in range(3):
                cell = self.board[i][j] if self.board[i][j] != ' ' else '□'
                row.append(InlineKeyboardButton(
                    text=cell,
                    callback_data=f"move_{self.game_id}_{i}_{j}"
                ))
            keyboard.append(row)

        # Добавляем кнопку выхода
        keyboard.append([InlineKeyboardButton(text="Выйти из игры",
callback_data=f"exit_{self.game_id}")])

        return InlineKeyboardMarkup(inline_keyboard=keyboard)
```

```
    def get_game_status(self):
        """Получить текущий статус игры"""
        if not self.game_started:
            return f"Игра #{self.game_id}\n\nОжидание второго игрока..."

        if self.winner:
            if self.winner == "draw":
                return f"Игра #{self.game_id}\n\n Ничья!"
            else:
                winner_name = self.get_player_name(self.winner)
                return f"Игра #{self.game_id}\n\n Победитель: {winner_name}!"

        current_name = self.get_player_name(self.current_player)
        symbol = '❌' if self.current_player == self.player1_id else '⭕'

        return (f"Игра #{self.game_id}\n\n"
                f"{self.player1_name} (❌) vs {self.player2_name} (⭕)\n\n"
                f"Ход: {current_name} ({symbol})")

def generate_game_id():
    """Генерация уникального ID игры"""
    return ''.join(random.choices(string.ascii_uppercase + string.digits, k=6))

@router.message(Command("start"))
async def cmd_start(message: Message):
    await message.answer(
        "Добро пожаловать в крестики-нолики!\n\n"
        "Команды:\n"
        "/new_game - Создать новую игру\n"
        "/join <ID> - Присоединиться к игре\n"
        "/my_games - Мои активные игры\n"
        "/help - Помощь"
```

```
    )

@router.message(Command("help"))
async def cmd_help(message: Message):
    await message.answer(
        "Справка по игре:\n\n"
        "1) Создайте игру командой /new_game\n"
        "2) Поделитесь ID игры с другом\n"
        "3) Друг присоединяется командой /join <ID>\n"
        "4) Играйте по очереди, нажимая на клетки\n\n"
        "Цель: собрать 3 символа в ряд\n"
        "Игрок 1 играет крестиками\n"
        "Игрок 2 играет ноликами"
    )

@router.message(Command("new_game"))
async def cmd_new_game(message: Message):
    game_id = generate_game_id()
    game = TicTacToeGame(game_id, message.from_user.id, message.from_user.first_name)
    games[game_id] = game

    await message.answer(
        f"🎮 Игра создана!\n\n"
        f"🆔 ID игры: <code>{game_id}</code>\n\n"
        f"Отправьте этот ID другу, чтобы он присоединился командой:\n"
        f"/join {game_id}\n\n"
        f"Ожидание второго игрока...",
        parse_mode="HTML"
    )

@router.message(Command("join"))
async def cmd_join(message: Message):
    # Извлекаем ID игры из команды
```

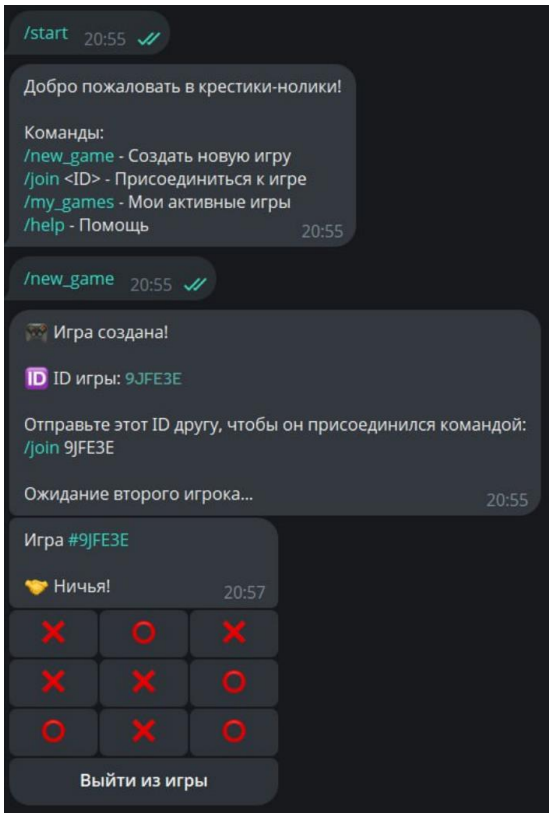
```
parts = message.text.split()
if len(parts) < 2:
    await message.answer("Укажите ID игры: /join <ID>")
    return

game_id = parts[1].upper()

if game_id not in games:
    await message.answer("Игра с таким ID не найдена")
    return

game = games[game_id]

if game.game_started:
    await message.answer("В этой игре уже играют два игрока")
    return

if game.player1_id == message.from_user.id:
    await message.answer("Вы не можете играть сами с собой")
    return

# Добавляем второго игрока
game.add_player2(message.from_user.id, message.from_user.first_name)

# Отправляем игровое поле обоим игрокам
for player_id in [game.player1_id, game.player2_id]:
    msg = await bot.send_message(
        player_id,
        game.get_game_status(),
        reply_markup=game.get_board_keyboard()
    )
    game.message_ids[player_id] = msg.message_id

await message.answer("Вы присоединились к игре! Игра началась!")
```

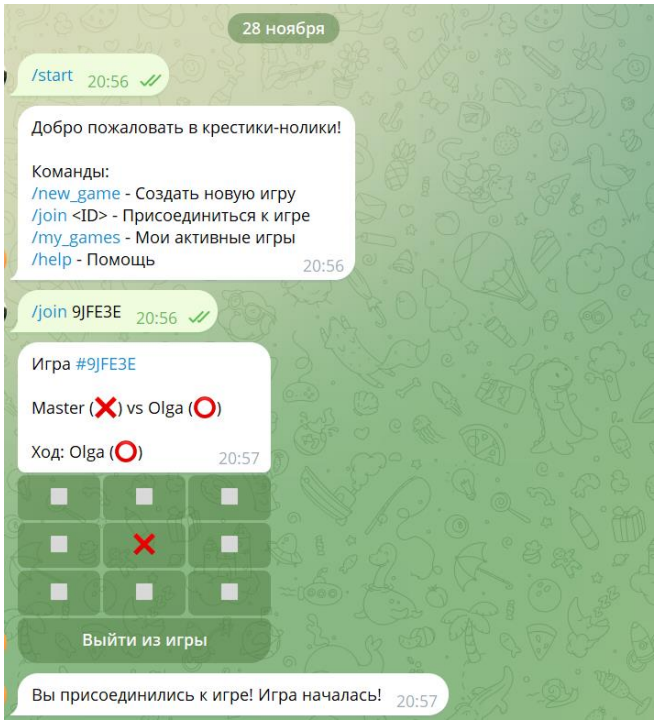
```
@router.message(Command("my_games"))
async def cmd_my_games(message: Message):
    """Показать активные игры пользователя"""
    user_games = []
    for game_id, game in games.items():
        if message.from_user.id in [game.player1_id, game.player2_id]:
            status = "□ Идет" if game.game_started and not game.winner else "□ Ожидание"
            if game.winner:
                status = "● Завершена"
            user_games.append(f"ID: {game_id} - {status}")

    if not user_games:
        await message.answer("У вас нет активных игр. Создайте новую: /new_game")
    else:
        games_list = "\n".join(user_games)
        await message.answer(f"🎮 Ваши игры:\n\n{games_list}")

@router.callback_query(F.data.startswith("move_"))
async def handle_move(callback: CallbackQuery):
    """Обработка хода"""
    parts = callback.data.split("_")
    game_id = parts[1]
    row = int(parts[2])
    col = int(parts[3])

    if game_id not in games:
        await callback.answer("Игра не найдена", show_alert=True)
        return

    game = games[game_id]
    success, result_message = game.make_move(callback.from_user.id, row, col)
```

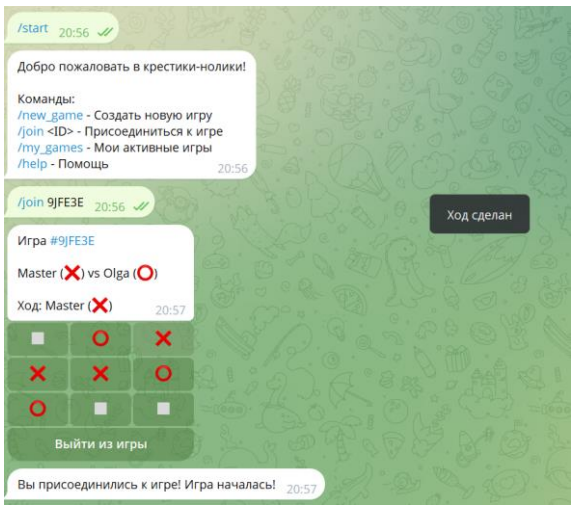
```
if not success:
    await callback.answer(result_message, show_alert=True)
    return

# Сначала отвечаем на callback
await callback.answer(result_message)

# Затем обновляем игровое поле у обоих игроков
for player_id, msg_id in game.message_ids.items():
    try:
        # Обновляем и текст, и клавиатуру одновременно
        await bot.edit_message_text(
            text=game.get_game_status(),
            chat_id=player_id,
            message_id=msg_id,
            reply_markup=game.get_board_keyboard()
        )
    except Exception as e:
        # Логируем ошибки вместо молчаливого игнорирования
        logging.error(f"Не удалось обновить сообщение для игрока {player_id}: {e}")
        # Пробуем обновить только клавиатуру
        try:
            await bot.edit_message_reply_markup(
                chat_id=player_id,
                message_id=msg_id,
                reply_markup=game.get_board_keyboard()
            )
        except Exception as e2:
            logging.error(f"Не удалось обновить клавиатуру: {e2}")

# Если игра закончена, удаляем её через 30 секунд
if game.winner:
    await asyncio.sleep(30)
    if game_id in games:
```

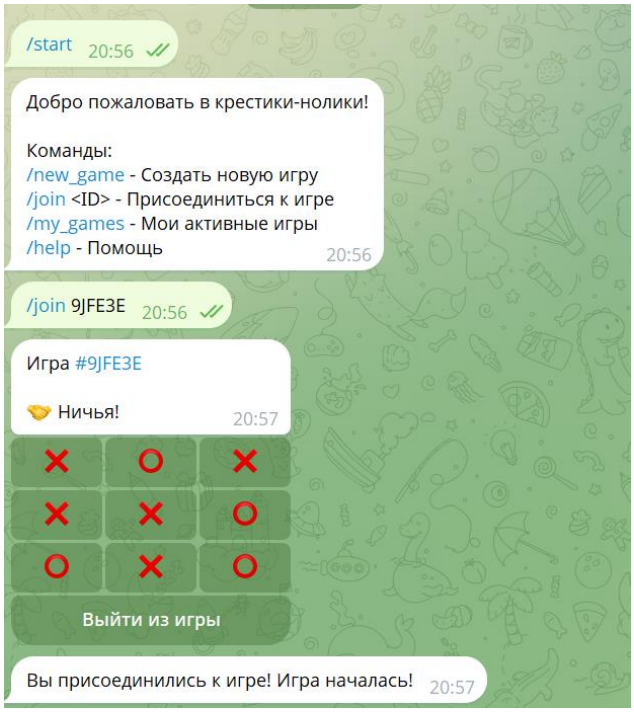
```
            del games[game_id]


@router.callback_query(F.data.startswith("exit_"))
async def handle_exit(callback: CallbackQuery):
    """Выход из игры"""
    game_id = callback.data.split("_")[1]

    if game_id in games:
        game = games[game_id]

        # Уведомляем всех игроков о завершении
        for player_id in [game.player1_id, game.player2_id]:
            if player_id:
                try:
                    await bot.send_message(
                        player_id,
                        f"🚪 {callback.from_user.first_name} вышел из игры. Игра завершена."
                    )
                except:
                    pass

        del games[game_id]

    await callback.answer("Вы вышли из игры")

async def main():
    dp.include_router(router)
    await dp.start_polling(bot)

if __name__ == "__main__":
    asyncio.run(main())
```

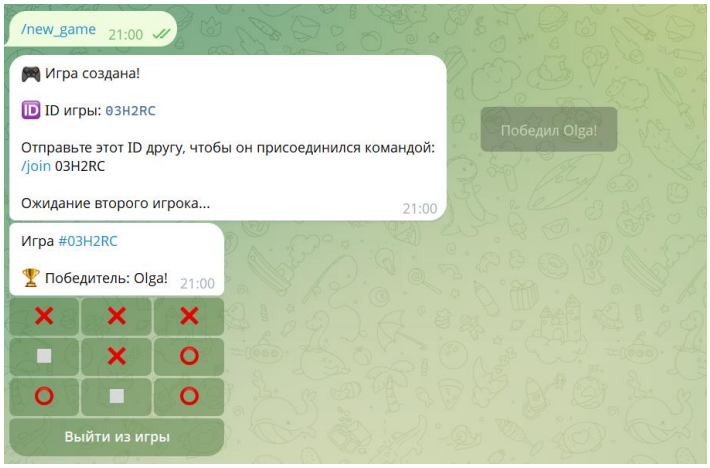
Результаты выполнения программы:

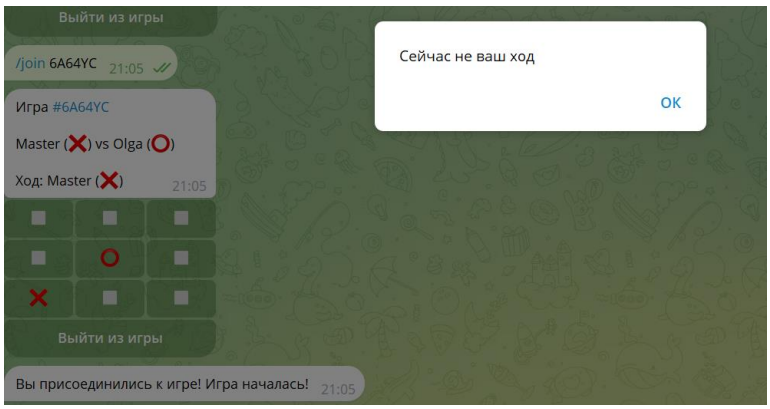


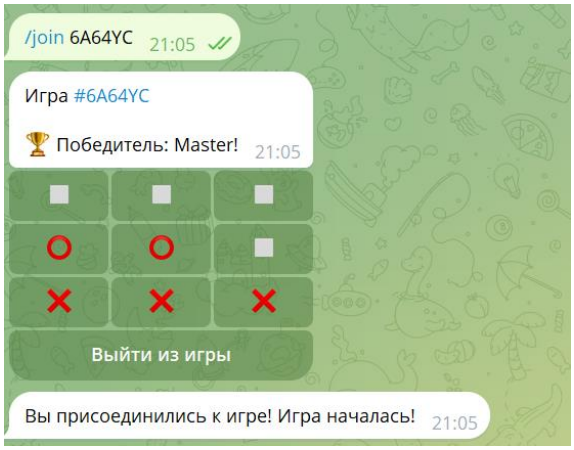



/start 20:56
Добро пожаловать в крестики-нолики!
Команды:
/new_game - Создать новую игру
/join <ID> - Присоединиться к игре
/my_games - Мои активные игры
/help - Помощь
20:56
/join 9JFE3E 20:56
Ход сделан
Игра #9JFE3E
Master (❌) vs Olga (⭕)
Ход: Master (❌) 20:57
Выйти из игры
Вы присоединились к игре! Игра началась! 20:57


/start 20:56
Добро пожаловать в крестики-нолики!
Команды:
/new_game - Создать новую игру
/join <ID> - Присоединиться к игре
/my_games - Мои активные игры
/help - Помощь
20:56
/join 9JFE3E 20:56
Игра #9JFE3E
🤝 Ничья! 20:57
Выйти из игры
Вы присоединились к игре! Игра началась! 20:57


/new_game 21:00
🎮 Игра создана!
ID игры: 03H2RC
Отправьте этот ID другу, чтобы он присоединился командой:
/join 03H2RC
Ожидание второго игрока... 21:00
Победил Olga!
Игра #03H2RC
🏆 Победитель: Olga! 21:00
Выйти из игры


Выйти из игры
/join 6A64YC 21:05
Игра #6A64YC
Master (❌) vs Olga (⭕)
Ход: Master (❌) 21:05
Выйти из игры
Вы присоединились к игре! Игра началась! 21:05
Сейчас не ваш ход
OK


/join 6A64YC 21:05
Игра #6A64YC
🏆 Победитель: Master! 21:05
Выйти из игры
Вы присоединились к игре! Игра началась! 21:05