Федеральное государственное автономное образовательное учреждение высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет _Информационный технологии_____

Кафедра «__Информационные системы и технологии__»

Направление подготовки/ специальность: _Автоматизированные системы обработки информации и управления / Программное обеспечение игровой компьютерной индустрии_

ОТЧЕТ

по проектной практике

Студент: Хуткубия Ника Игрикович

Группа: ____241-337_____

Место прохождения практики: Московский Политех, кафедра _____

Отчет принят с оценкой _____ Дата _____

Руководитель практики: _____

Москва 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

1. Общая информация о проекте

Название проекта

«Помощник для создания персонажей D&D» – Telegram-бот для автоматизации процесса создания и хранения персонажей в настольной ролевой игре Dungeons & Dragons 5-й редакции.

Цели и задачи проекта

Цель:

Создать удобный инструмент для игроков и мастеров D&D, позволяющий быстро создавать персонажей, распределять характеристики, выбирать расы, классы и заклинания, а также хранить готовые листы для последующего использования.

Задачи:

1. Разработать интерактивного Telegram-бота с использованием библиотеки `pyTelegramBotAPI`.

2. Реализовать систему создания персонажей с учетом правил D&D 5e:

- Выбор расы (человек, эльф, гном, гоблин, драконорождённый, орк).

- Распределение характеристик (15, 14, 13, 12, 10, 8) с добавлением расовых бонусов.

- Выбор класса и подкласса с уникальными особенностями.

- Добавление заклинаний для магических классов.

3. Организовать хранение данных в формате JSON.

4. Реализовать поиск персонажей по имени.

5. Обеспечить удобный интерфейс с клавиатурой и подсказками.

---

2. Общая характеристика деятельности организации (заказчика проекта)

Наименование заказчика

Условным заказчиком проекта выступает сообщество любителей настольных ролевых игр (НРИ), в частности, игроки и мастера Dungeons & Dragons.

Организационная структура

Проект разрабатывался индивидуально, однако взаимодействие велось с тестовой группой пользователей (игроки D&D), которые предоставляли обратную связь по функционалу бота.

Описание деятельности

Сообщество НРИ нуждается в удобных цифровых инструментах для упрощения подготовки к игровым сессиям. Существующие аналоги (D&D Beyond, Roll20) требуют регистрации, имеют сложный интерфейс или ограниченный бесплатный функционал. Данный бот решает проблему быстрого создания персонажей "на лету" прямо в Telegram.

3. Описание задания по проектной практике

Выполненные задачи

1. Анализ требований

- Изучены правила D&D 5e по созданию персонажей.

- Определен минимально необходимый функционал (расы, классы, характеристики).

2. Проектирование архитектуры бота

- Разработана схема взаимодействия пользователя с ботом (см. *Приложение 1*).

- Определен формат хранения данных (JSON).

3. Реализация функционала

- Создана система пошагового создания персонажа.

- Реализована валидация вводимых данных (проверка характеристик, уникальности имени).

- Добавлены расовые бонусы и особенности подклассов.

4. Тестирование

- Проверка корректности распределения характеристик.

- Тестирование хранения и поиска персонажей.

Работа в команде

Проект разрабатывался индивидуально, но для тестирования привлекались игроки D&D. Их обратная связь позволила:

- Упростить интерфейс (добавить кнопки вместо ручного ввода).

- Исправить ошибки в расчетах характеристик.

Инструменты и методы

- Управление задачами: Trello (см. *Приложение 2*).

- Версионный контроль: GitHub.

- Коммуникация: Discord для сбора отзывов.

Освоенные навыки

1. Профессиональные:**

- Работа с API Telegram.

- Обработка JSON-данных.

- Валидация пользовательского ввода.

2. Коммуникативные:

- Сбор и анализ требований от потенциальных пользователей.
- Координация обратной связи.

---

4. Достигнутые результаты

1. Реализованный функционал:
   - Создание персонажей с учетом всех правил D&D 5e.
   - Хранение данных в JSON-файле.
   - Поиск персонажей по имени.
2. Пользовательский интерфейс:
   - Интуитивно понятное меню с кнопками.
   - Подробный вывод информации о персонаже.
3. Гибкость:
   - Код легко расширяется (можно добавить новые расы, классы, заклинания).

ЗАКЛЮЧЕНИЕ

Проект успешно решает поставленные задачи:

- Упрощает процесс создания персонажей D&D.
- Позволяет хранить и быстро находить листы персонажей.
- Может быть интегрирован в крупные игровые сообщества.

Перспективы развития:

- Добавление генерации PDF-листов.
- Поддержка многопользовательского режима (для мастеров).

ПРИЛОЖЕНИЯ

Подробный разбор программы Telegram-бота для создания персонажей D&D

Программа написана на Python с использованием библиотеки `pyTelegramBotAPI` (она же `telebot`).

---

1. Основные компоненты бота

1.1. Что делает этот бот?

- Помогает создать персонажа для D&D 5e.
- Позволяет выбрать:
  - Расу (человек, эльф, гном и др.).
  - Класс (воин, маг, жрец и др.).
  - Подкласс (например, для мага — «Школа некромантии»).
  - Характеристики (сила, ловкость и др.).
  - Заклинания (если класс магический).
- Сохраняет персонажей в JSON-файл.
- Позволяет искать созданных персонажей по имени.

---

2. Как устроен код?

Программа состоит из нескольких частей:

1. Настройка бота (импорт библиотек, токен бота).

2. Хранение данных (расы, классы, заклинания).

3. Основные функции (создание и поиск персонажей).

4. Обработка команд пользователя.

2.1. Настройка бота

```python
import telebot
from telebot import types
import json
import os

bot = telebot.TeleBot('YOUR_TELEGRAM_BOT_TOKEN')
```

- `telebot` — библиотека для работы с Telegram API.
- `json` — для сохранения данных в файл.
- `os` — для работы с файлами.
- `bot = telebot.TeleBot('TOKEN')` — создание бота (замени `TOKEN` на свой, полученный от @BotFather).

2.2. Хранение данных

Расы и их бонусы

```python
RACES = {
```

```
    'Человек': {'bonuses': {'strength': 1, 'dexterity': 1, 'constitution': 1, 'intelligence': 1, 'wisdom': 1, 'charisma': 1}},
    'Эльф': {'bonuses': {'dexterity': 2, 'wisdom': 1}},
    'Гном': {'bonuses': {'constitution': 2, 'wisdom': 1}},
    'Гоблин': {'bonuses': {'dexterity': 2, 'constitution': 1}},
    'Драконорождённый': {'bonuses': {'strength': 2, 'charisma': 1}},
    'Орк': {'bonuses': {'strength': 2, 'constitution': 1}}
}
```

Здесь указано, какие бонусы даёт каждая раса. Например, эльф получает `+2` к ловкости (`dexterity`) и `+1` к мудрости (`wisdom`).

Классы и подклассы

```python
CLASSES = {
    'Воин': {
        'subclasses': ['Чемпион', 'Боевой мастер', 'Мистический рыцарь'],
        'spellcaster': False  # Не использует магию
    },
    'Волшебник': {
        'subclasses': ['Школа воплощения', 'Школа некромантии', 'Школа иллюзий'],
        'spellcaster': True  # Использует магию
    },
    # ... остальные классы
}
```

- `subclasses` — доступные подклассы.
- `spellcaster` — если `True`, персонаж может выбирать заклинания.

Заклинания

```python
SPELLS = {
    'Волшебник': {
        'cantrips': ['Огненный снаряд', 'Луч холода', 'Магическая рука'],  # Заговоры
        'spells': ['Волшебная стрела', 'Огненный шар', 'Невидимость']  # Обычные заклинания
    },
    'Жрец': {
        'cantrips': ['Священное пламя', 'Слово силы', 'Лечение ран'],
        'spells': ['Наказующий удар', 'Воскрешение', 'Защита от зла и добра']
    }
}
```

Здесь хранятся заклинания для каждого магического класса.

2.3. Сохранение и загрузка персонажей

```python
CHARACTERS_FILE = 'characters.json'  # Файл для хранения данных
```

```
def load_characters():
    if os.path.exists(CHARACTERS_FILE):
        with open(CHARACTERS_FILE, 'r') as f:
            return json.load(f)
    return {}

def save_characters(characters):
    with open(CHARACTERS_FILE, 'w') as f:
        json.dump(characters, f, indent=4)
```

- `load_characters()` — загружает сохранённых персонажей из файла.
- `save_characters()` — сохраняет новых персонажей в файл.

2.4. Основные команды бота

Старт (`/start`)

```python
@bot.message_handler(commands=['start'])
def start(message):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    btn1 = types.KeyboardButton('Создать персонажа')
    btn2 = types.KeyboardButton('Найти персонажа')
    markup.add(btn1, btn2)
    bot.send_message(message.chat.id, "Добро пожаловать в генератор персонажей D&D!", reply_markup=markup)
```

- При запуске `/start` бот показывает кнопки:
  - `Создать персонажа`
  - `Найти персонажа`

Создание персонажа

1. Ввод имени

   ```python
   def create_character(message):
       msg = bot.send_message(message.chat.id, "Введите имя персонажа:", reply_markup=types.ReplyKeyboardRemove())
       bot.register_next_step_handler(msg, process_name_step)
   ```

   - Бот ждёт, пока пользователь введёт имя.

2. Выбор расы

   ```python
   def process_name_step(message):
       name = message.text
       user_data = {'name': name, 'level': 1}
       markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
       buttons = [types.KeyboardButton(race) for race in RACES.keys()]
       markup.add(*buttons)
       bot.send_message(message.chat.id, "Выберите расу:", reply_markup=markup)
       bot.register_next_step_handler(msg, process_race_step, user_data)
   ```

- Показывает кнопки с расами (человек, эльф и др.).

3. Выбор класса и подкласса

```python
def process_race_step(message, user_data):
    race = message.text
    user_data['race'] = race
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True, row_width=2)
    buttons = [types.KeyboardButton(cls) for cls in CLASSES.keys()]
    markup.add(*buttons)
    bot.send_message(message.chat.id, "Выберите класс:", reply_markup=markup)
    bot.register_next_step_handler(msg, process_class_step, user_data)
```

- После выбора расы бот предлагает классы.

4. Распределение характеристик

```python
def process_stats_step(message, user_data):
    stats = list(map(int, message.text.split()))  # Получаем числа [15, 14, 13, 12, 10, 8]
    user_data['stats'] = {
        'strength': stats[0],
        'dexterity': stats[1],
        'constitution': stats[2],
        'intelligence': stats[3],
        'wisdom': stats[4],
        'charisma': stats[5]
    }
```

```
    # Добавляем расовые бонусы
    race_bonuses = RACES[user_data['race']]['bonuses']
    for stat, bonus in race_bonuses.items():
        user_data['stats'][stat] += bonus
```

- Пользователь вводит числа 15, 14, 13, 12, 10, 8 (например, `15 14 13 12 10 8`).
- Бот автоматически добавляет бонусы от расы.

5. Выбор заклинаний (если класс магический)

```python
if CLASSES[user_data['class']]['spellcaster']:
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    buttons = [types.KeyboardButton(spell) for spell in SPELLS[user_data['class']]['cantrips']]
    markup.add(*buttons)
    bot.send_message(message.chat.id, "Выберите заговор:", reply_markup=markup)
    bot.register_next_step_handler(msg, process_cantrip_step, user_data)
```

- Если класс магический, бот предлагает выбрать заговоры и заклинания.

6. Сохранение персонажа

```python
def complete_character(message, user_data):
    characters = load_characters()
    characters[user_data['name']] = user_data
    save_characters(characters)
    bot.send_message(message.chat.id, "Персонаж создан!", reply_markup=main_menu())
```

```

- Персонаж сохраняется в JSON-файл.

---

Поиск персонажа

```python
def find_character(message):
    msg = bot.send_message(message.chat.id, "Введите имя персонажа:", reply_markup=types.ReplyKeyboardRemove())
    bot.register_next_step_handler(msg, process_find_character)

def process_find_character(message):
    name = message.text
    characters = load_characters()
    if name in characters:
        character_info = format_character_info(characters[name])
        bot.send_message(message.chat.id, character_info, reply_markup=main_menu())
    else:
        bot.send_message(message.chat.id, "Персонаж не найден.", reply_markup=main_menu())
```

- Пользователь вводит имя персонажа.
- Бот ищет его в файле и выводит информацию, если находит.

---

3. Как запустить бота?
```

1. Установи Python ([скачать с официального сайта](https://www.python.org/downloads/)).

2. Установи библиотеку `pyTelegramBotAPI`:

   ```bash
   pip install pyTelegramBotAPI
   ```

3. Создай бота в Telegram через @BotFather и получи токен.

4. Вставь токен в код:

   ```python
   bot = telebot.TeleBot('ТВОЙ_ТОКЕН_ЗДЕСЬ')
   ```

5. Запусти бота:

   ```bash
   python bot.py
   ```

---

4. Что можно улучшить?

- Добавить генерацию PDF-листа** персонажа.

- Сделать интерактивные кнопки** для выбора характеристик.

- Добавить больше рас и классов** из D&D.

Вывод

Этот бот —удобный инструмент для быстрого создания персонажей D&D.

Он использует:

- Telegram Bot API для общения с пользователем.

- JSON для хранения данных.

- Клавиатуры для удобного выбора.

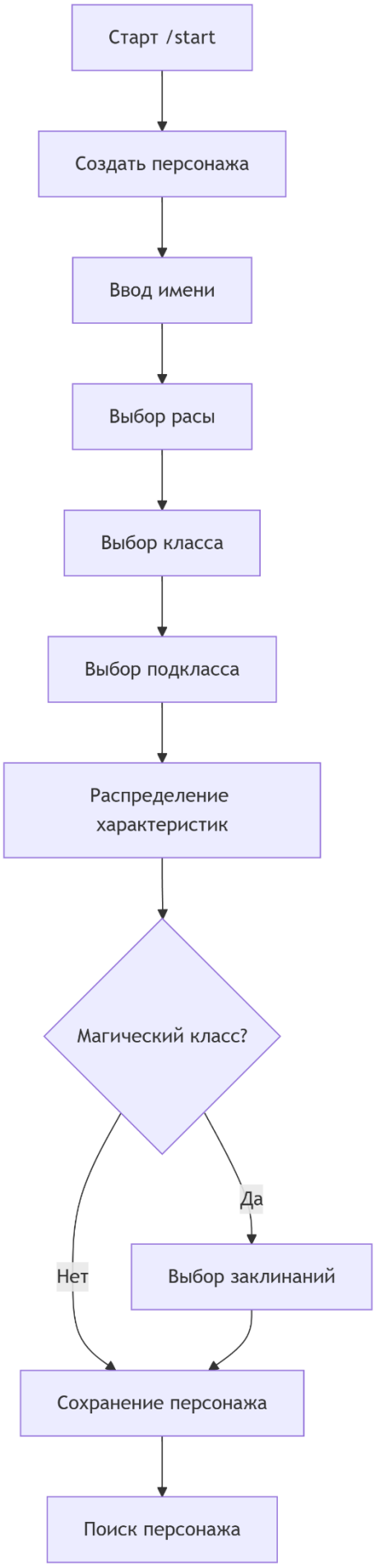
Код можно расширять и дорабатывать под свои нужды! 🚀

Приложение 1

Схема взаимодействия с ботом:

```mermaid

graph TD
```


Старт /start
Создать персонажа
Ввод имени
Выбор расы
Выбор класса
Выбор подкласса
Распределение характеристик
Магический класс?
Да
Нет
Выбор заклинаний
Сохранение персонажа
Поиск персонажа

Приложение 2

Пример Trello-доски:

Задача	Статус
Реализация выбора расы	✔ Выполнено
Добавление заклинаний	✔ Выполнено
Тестирование поиска	✔ Выполнено

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И ИСТОЧНИКОВ

1. Официальные материалы по D&D 5e

1. **Player’s Handbook (D&D 5e)** – базовые правила создания персонажей:

https://dnd.wizards.com/products/tabletop-games/rpg-products/rpg_playershandbook

2. **Basic Rules (бесплатная версия правил D&D 5e)** – основы механик:

https://dnd.wizards.com/resources/basic-rules

3. **D&D Beyond** – онлайн-инструменты для создания персонажей (аналог для сравнения):

https://www.dndbeyond.com/

---

2. Техническая документация

4. **Официальная документация pyTelegramBotAPI** – библиотека для создания Telegram-ботов:

https://github.com/eternnoir/pyTelegramBotAPI

5. **Python JSON Documentation** – работа с JSON в Python:

https://docs.python.org/3/library/json.html

6. **Telegram Bot API** – официальная документация Telegram:

https://core.telegram.org/bots/api

---

3. Дополнительные источники

7. **Статья «Как создать Telegram-бота на Python» (Medium)** – базовый гайд:

https://medium.com/@andrewmarmstrong/how-to-create-a-telegram-bot-using-python-4cfb3d3f5f28

8. **Гайд по работе с JSON в Python (Real Python)** – сохранение и загрузка данных:

https://realpython.com/python-json/

9. **D&D 5e Race/Class Guides (RPGBOT)** – баланс рас и классов:

https://rpgbot.net/dnd5/

10. **Примеры ботов для D&D (GitHub)** – вдохновение и аналоги:

https://github.com/topics/dnd-bot

- **IEEE Xplore** (https://ieeexplore.ieee.org/) для статей по разработке ботов.

- **Springer** (https://link.springer.com/) для исследований по UX в чат-ботах.

Сылка на бота

@dndCharacterCreation_bot