Документация проекта: «Управление спортивным инвентарем»

**1. Титульный лист**

* **Название кейса:** Командный кейс №5 «Управление спортивным инвентарем»
* **Члены команды:**
  + Бардин Константин Алексеевич
  + Новиков Виктор Николаевич
  + Саргаева Анна Сергеевна
  + Юрченко Семён Сергеевич
  + Климин Марк Иванович
* **Руководитель:**
  + Гоптарь Евгений Андреевич
  + Малевин Дмитрий Сергеевич
* **Школа:** Школа № 2098 имени Героя Советского Союза Л.М. Доватора

**2. Обоснование выбора языка программирования и используемых программных средств**

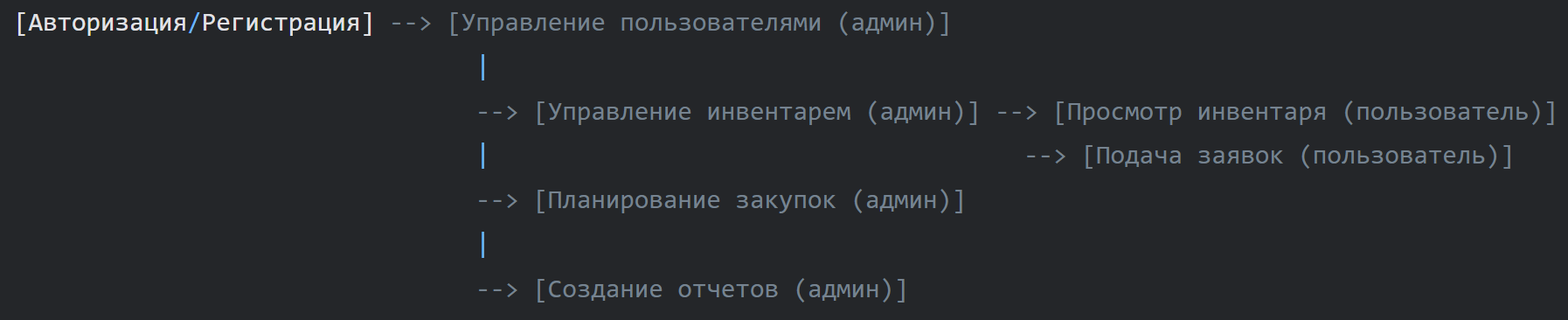
* **Язык программирования:** Python 3.x
  + **Обоснование:** Python выбран из-за его простоты в освоении, широкой распространенности, богатой экосистемы библиотек и фреймворков, а также его пригодности для веб-разработки и обработки данных.
* **Веб-фреймворк:** Flask
  + **Обоснование:** Flask – микрофреймворк, который предоставляет необходимые инструменты для создания веб-приложений, оставаясь при этом легким и гибким. Это позволяет быстро прототипировать и разрабатывать небольшие и средние проекты.
* **Фронтенд:** HTML, CSS, JavaScript
  + **Обоснование:** HTML используется для структуры веб-страниц, CSS для стилизации и оформления, а JavaScript для добавления интерактивности и динамического поведения на стороне клиента. Это стандартный набор технологий для разработки веб-интерфейсов.
* **Система управления базами данных (СУБД):** JSON-файлы (планируется переход на SQLAlchemy)
  + **Обоснование:** На начальном этапе JSON-файлы используются для упрощения разработки и отладки. В дальнейшем планируется переход на SQLAlchemy для обеспечения более надежного и масштабируемого хранения данных.
* **Система контроля версий:** Git
  + **Обоснование:** Git позволяет отслеживать изменения в коде, совместно работать над проектом и возвращаться к предыдущим версиям при необходимости.
* **Менеджер пакетов:** pip
  + **Обоснование:** pip упрощает установку и управление зависимостями Python, обеспечивая воспроизводимость проекта на разных машинах.

**3. Структурная и функциональная схемы программного продукта**

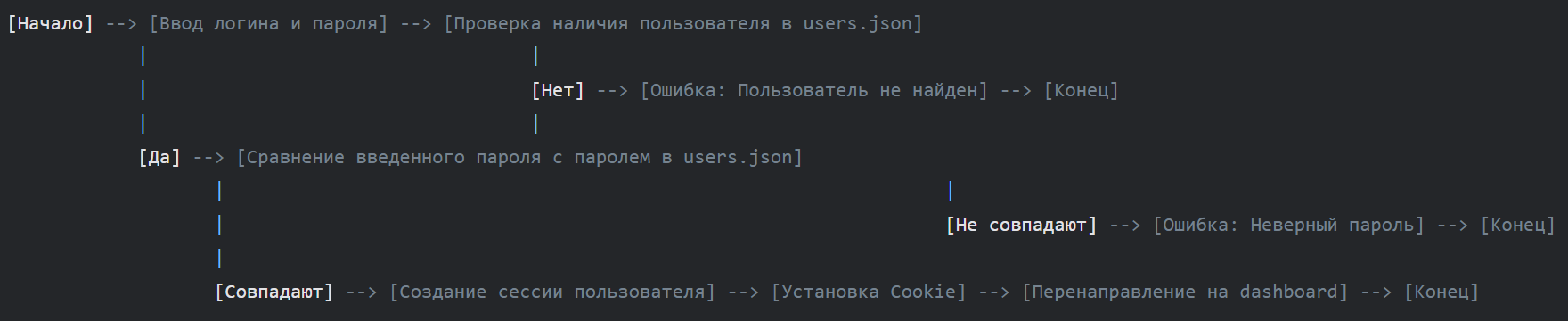
* **Структурная схема:**



**Описание:** Клиент взаимодействует с сервером Flask через HTTP-запросы. Flask обрабатывает запросы, взаимодействует с JSON-файлами для получения и хранения данных, и возвращает ответы клиенту.

* **Функциональная схема:**
  + 
  + **Описание:** Схема показывает основные функциональные блоки приложения и их взаимосвязи.

**4. Блок-схема работы основного алгоритма (Авторизация)**



**5. Описание особенностей и аргументация выбранного типа СУБД**

* **Текущая СУБД: JSON-файлы**
  + **Особенности:** Простота реализации, отсутствие необходимости в установке и настройке отдельной СУБД.
  + **Аргументация:** Удобно на начальном этапе разработки для быстрого прототипирования и тестирования.
* **Планируемая СУБД: SQLAlchemy (ORM для Python)**
  + **Особенности:** ORM (Object-Relational Mapping) позволяет работать с базой данных как с Python-объектами, упрощая разработку и повышая безопасность. Поддержка различных СУБД (PostgreSQL, MySQL, SQLite и др.).
  + **Аргументация:** SQLAlchemy обеспечивает более надежное, масштабируемое и безопасное хранение данных по сравнению с JSON-файлами. ORM упрощает взаимодействие с базой данных, уменьшая количество ручного SQL-кода.
* **Возможные варианты:**
  + SQLite ([https://dt.miet.ru/it/info/docs/methods](https://www.google.com/url?sa=E&q=https%3A%2F%2Fdt.miet.ru%2Fit%2Finfo%2Fdocs%2Fmethods))
  + MySQL ([https://dev.mysql.com/doc/](https://www.google.com/url?sa=E&q=https%3A%2F%2Fdev.mysql.com%2Fdoc%2F))

**6. Схема базы данных**

*(В данном разделе нужно представить схему базы данных. Так как сейчас используются JSON-файлы, можно представить структуру каждого файла в виде таблицы. Когда будет использоваться SQLAlchemy, нужно будет добавить ER-диаграмму.)*

* **users.json:**

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | Integer | Уникальный идентификатор пользователя |
| username | String | Имя пользователя для входа |
| password | String | Пароль пользователя |
| role | String | Роль пользователя (admin/user) |
| last\_login | String | Дата и время последнего входа |

* **inventory.json:**

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | Integer | Уникальный идентификатор предмета |
| name | String | Название предмета |
| image | String | URL изображения |
| image\_type | Integer | Тип изображения (0-нет, 1-папка, 2-URL) |
| quantity | Integer | Количество |
| state | String | Состояние (новый, используется, сломан) |

* **inventory\_assignments.json:**

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | Integer | Уникальный идентификатор назначения |
| user\_id | Integer | ID пользователя, за которым закреплено |
| inventory\_id | Integer | ID предмета инвентаря |
| quantity\_assigned | Integer | Количество назначенного инвентаря |
| assignment\_date | String | Дата назначения |

* **purchase\_plans.json:**

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | Integer | Уникальный идентификатор плана |
| inventory\_id | Integer | ID предмета инвентаря для закупки |
| quantity | Integer | Количество для закупки |
| price | Integer | Цена за единицу |
| supplier | String | Название поставщика |

* **requests.json:**

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| id | Integer | Уникальный идентификатор заявки |
| user\_id | Integer | ID пользователя, подавшего заявку |
| inventory\_id | Integer | ID предмета инвентаря |
| quantity\_requested | Integer | Запрашиваемое количество |
| status | String | Статус заявки (pending, approved, rejected) |
| request\_type | String | Тип заявки (get, repair, replace) |
| request\_date | String | Дата подачи заявки |
| request\_date | String | Дата подачи заявки |

* **user\_pages\_settings.json:**

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип данных** | **Описание** |
| admin | Массив объектов | массив объектов, с подстраницами для администратора |
| user | Массив объектов | массив объектов, с подстраницами для пользователя |
| name | String | Название страницы |
| subpages | Массив объектов | массив объектов с подподстраницами |
| name | String | Название подподстраницы |
| order | String | Порядок отображения |

**7. Программный код**

* **Ссылка на репозиторий:** (Укажите ссылку на репозиторий на GitHub)

*README.md*

# Система учета спортивного инвентаря

Веб-приложение для учета и контроля спортивного инвентаря в школе.

## Функциональность

\* Авторизация/Регистрация пользователей (администратор, пользователь)

\* Управление инвентарем (добавление, редактирование, удаление)

\* Закрепление инвентаря за пользователями

\* Планирование закупок

\* Формирование отчетов

\* Подача заявок на инвентарь (пользователь)

\* Отслеживание статуса заявок (пользователь)

## Инструкция по установке/развертыванию

1. Клонируйте репозиторий:

git clone <repository\_url>

2. Перейдите в каталог проекта:

cd <project\_directory>

3. Создайте и активируйте виртуальное окружение:

python3 -m venv venv

source venv/bin/activate # Linux/macOS

venv\Scripts\activate.bat # Windows

4. Установите зависимости:

pip install flask

5. Запустите приложение:

python main.py

6. Откройте веб-браузер и перейдите по адресу `https://0.0.0.0:443/` (или адресу, указанному в консоли при запуске приложения).

*Содержимое main.py (основной файл Flask-приложения):*

* Все файлы в файлах в начале ответа.
* Основные моменты:
  + Реализована базовая функциональность Flask для обработки запросов и рендеринга шаблонов.
  + Функции для загрузки и сохранения JSON-данных.
  + Реализованы маршруты для авторизации, регистрации и dashboard.
  + Реализованы маршруты для работы с инвентарем, назначениями, закупками и заявками.
  + Обработка ошибок и логирование.
* *Содержимое requirements.txt (список зависимостей Python):*  
  *В данном проекте requirements.txt отсутствует, все зависимости указаны выше.*
* *Файлы HTML-шаблонов (templates/\*.html):*  
  *В данном проекте все файлы шаблонов указаны выше.*
* Шаблоны HTML для отображения страниц авторизации, регистрации и dashboard.
* Шаблоны для отображения информации об инвентаре, назначениях, закупках и заявках.
* Формы для добавления, редактирования и удаления данных.
* *Файлы статики (static/css/\*.css, static/js/\*.js, static/images/\*):*  
  *В данном проекте все файлы статики указаны выше.*
* CSS-файлы для стилизации веб-страниц.
* JavaScript-файлы для добавления интерактивности и обработки данных на стороне клиента.
* Изображения, используемые в приложении.
* *Файлы данных (data/\*.json):*  
  *В данном проекте все файлы данных указаны выше.*
* Файлы JSON для хранения информации о пользователях, инвентаре, назначениях, закупках и заявках.

**8. Заключение**

Данная документация предоставляет подробное описание разработанного веб-приложения для учета спортивного инвентаря. Она включает в себя обоснование выбора технологий, структурные и функциональные схемы, описание схемы базы данных и инструкции по установке и развертыванию.