ГБОУ Школа №2025

Московская предпрофессиональная олимпиада школьников 2024-2025

Профиль «Информационные технологии»

Командный кейс №5 «Управление спортивным инвентарем»

Команда «Добро»

Авторы:

Крылов Игорь Дмитриевич

Магунов Кирилл Артемович

Курзин Александр Сергеевич

Лазарев Максим Дмитриевич

Руководитель:

Гонта Алексей Степанович

Оглавление

Обоснование выбора языка программирования и библиотеки Flask…..…3 – 4

Структурная и функциональная схемы программного продукта…………5 – 6

Блок-схема работы основного алгоритма…………………………………..7 – 9

Аргументация выбранного типа СУБД…………………………………...10 – 11

Схема базы данных……………………………………………………………...12

Литература…...…………………………………………………………………..13

Обоснование выбора языка программирования

Выбор Python и Flask для создания сайта обусловлен сочетанием простоты использования, быстрой разработки и удобной структуры, что критически важно для проектов с ограниченными ресурсами и временем, как часто бывает в образовательных учреждениях.

Простота Python:

Легкость изучения и использования: Python известен своим чистым и понятным синтаксисом, который напоминает английский язык. Это значительно снижает порог вхождения для начинающих программистов, что делает его идеальным выбором для школьного проекта.

Динамическая типизация: Python - это язык с динамической типизацией, что означает, что вам не нужно явно объявлять типы переменных. Это упрощает и ускоряет разработку, так как позволяет сосредоточиться на логике приложения, а не на деталях типизации.

Преимущества Flask:

Микрофреймворк: Flask - это микрофреймворк, что означает, что он предоставляет только основные компоненты для создания веб-приложений. Это делает его очень легким, гибким и легко настраиваемым.

Простая структура: Flask предлагает интуитивно понятную структуру приложения, основанную на маршрутах и функциях представления. Это облегчает понимание и организацию кода, особенно для небольших и средних проектов.

• Легкая интеграция с другими библиотеками: Flask легко интегрируется с другими популярными библиотеками Python, такими как SQLite3 (СУБД) и Jinja2 (для создания шаблонов). Это позволяет расширить функциональность приложения без необходимости изучать сложные фреймворки.

Поддержка шаблонизатора: Jinja2 - мощный и гибкий шаблонизатор, который позволяет создавать динамические HTML-страницы с использованием Python-кода. Он поддерживает наследование шаблонов, фильтры и другие полезные функции, которые упрощают разработку пользовательского интерфейса.

Маршрутизация: Flask использует декораторы *@app.route()* для определения маршрутов, которые связывают URL-адреса с функциями представления (view функции). Это делает код более выраженным и понятным, так как легко увидеть, какой URL-адрес обрабатывается какой функцией.

Простое управление запросами и ответами: Flask предоставляет удобные объекты *request* и *response* для обработки HTTP-запросов и ответов. Вы можете легко получить доступ к данным формы, заголовкам, файлам и другим параметрам запроса, а также установить заголовки ответа, куки и другие параметры.

Описание функциональной схемы

Пользовательский интерфейс (Браузер): Пользователь взаимодействует с сайтом через браузер. Основные страницы: главная, авторизация, регистрация, просмотр инвентаря, промежуточная.

Серверная часть (Flask):

app.py (Flask-приложение): Ядро приложения. Обрабатывает запросы пользователей, взаимодействует с базой данных, выбирает шаблоны и формирует ответы.

database.db (SQLite): Хранит все данные: информация о пользователях, данные о спортивном инвентаре (название, количество, состояние).

schema.sql: Используется при первом запуске для создания структуры базы данных (таблицы userlist, inventory, tobuylist).

templates: Содержит HTML-шаблоны, которые используются для генерации страниц. login.html, register.html, Blank1.html, view.html.

static: Содержит статические файлы (CSS, JavaScript, изображения), эти файлы используются для оформления и улучшения пользовательского интерфейса.

Обработка данных:

Пользователь вводит данные в форме (например, при авторизации или добавлении предмета).

Данные отправляются на сервер (POST-запрос) и обрабатываются функцией в app.py, связанной с этим маршрутом.

app.py взаимодействует с базой данных (database.db): извлекает данные (*SELECT*), добавляет новые данные (*INSERT*), изменяет данные (*UPDATE*), удаляет данные (*DELETE*).

В функции выбирается нужный шаблон из папки templates, и с помощью движка Jinja2 происходит рендеринг страницы. HTML-код страницы отправляется пользователю, где браузер отображает интерфейс пользователю.

Регистрация: Пользователь заполняет форму регистрации, данные отправляются на сервер, Flask сохраняет хешированный пароль в БД.

Авторизация: Пользователь вводит логин и пароль, Flask проверяет соответствие логина и пароля в БД, если данные верны, создает сессию для пользователя.

Просмотр инвентаря: Flask извлекает данные об инвентаре из БД и отображает их списком на странице view.html.

Добавление инвентаря: Администратор заполняет форму добавления инвентаря, данные отправляются на сервер, Flask сохраняет данные в БД.

Блок-схема работы основного алгоритма: 1. Регистрация/Авторизация

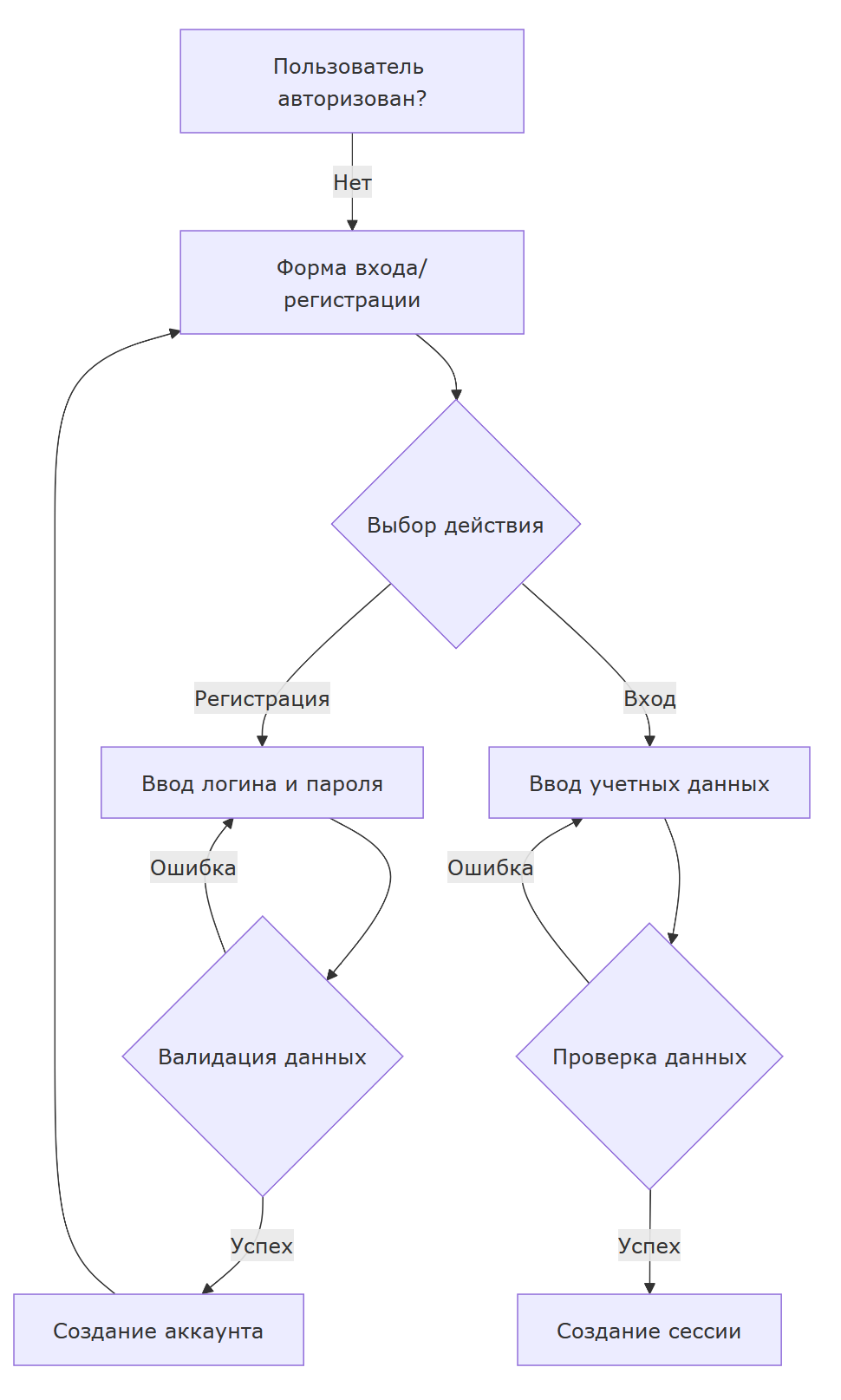


Рис. 1

1. Действия администратора

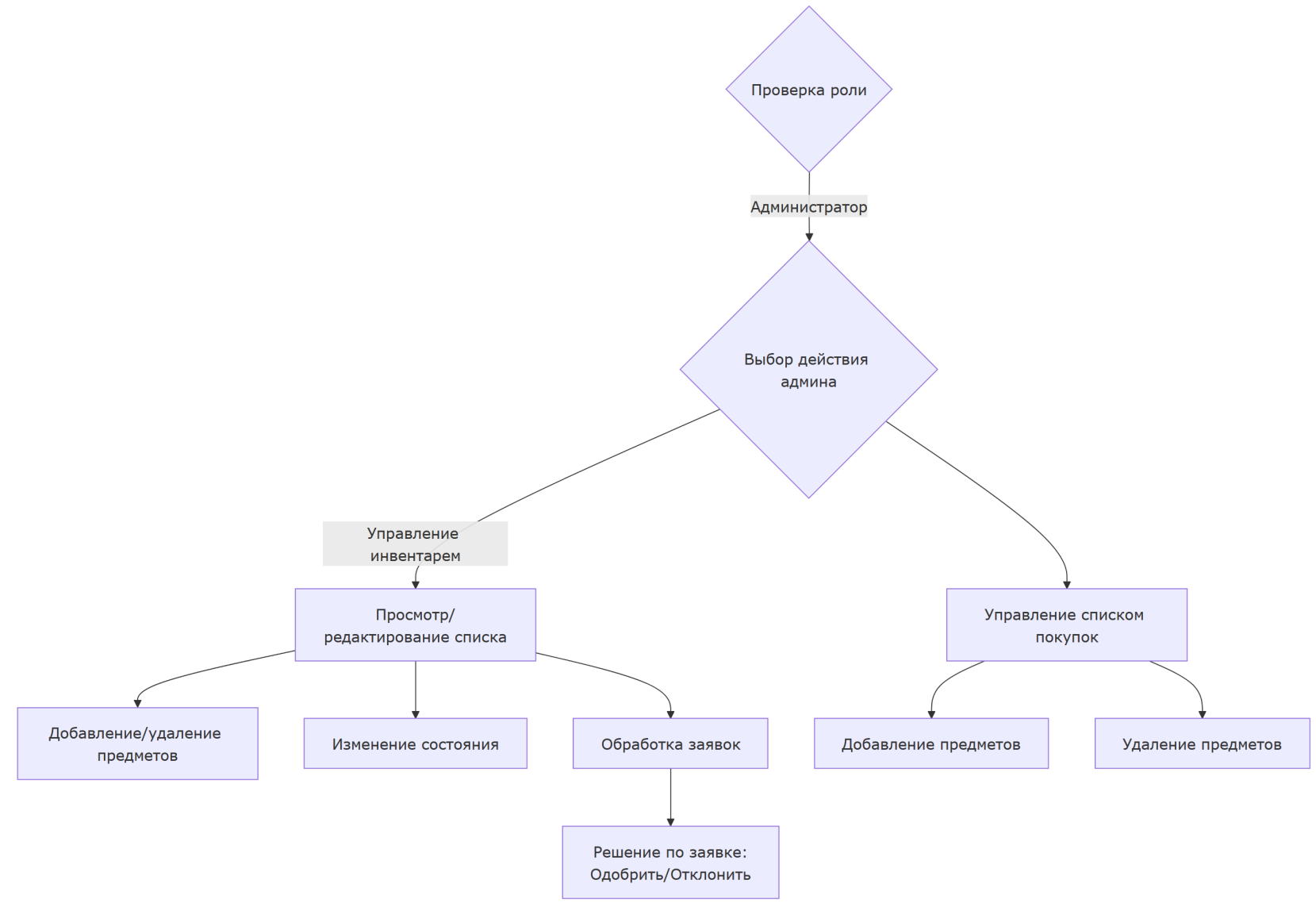


Рис. 2

1. Действия пользователя

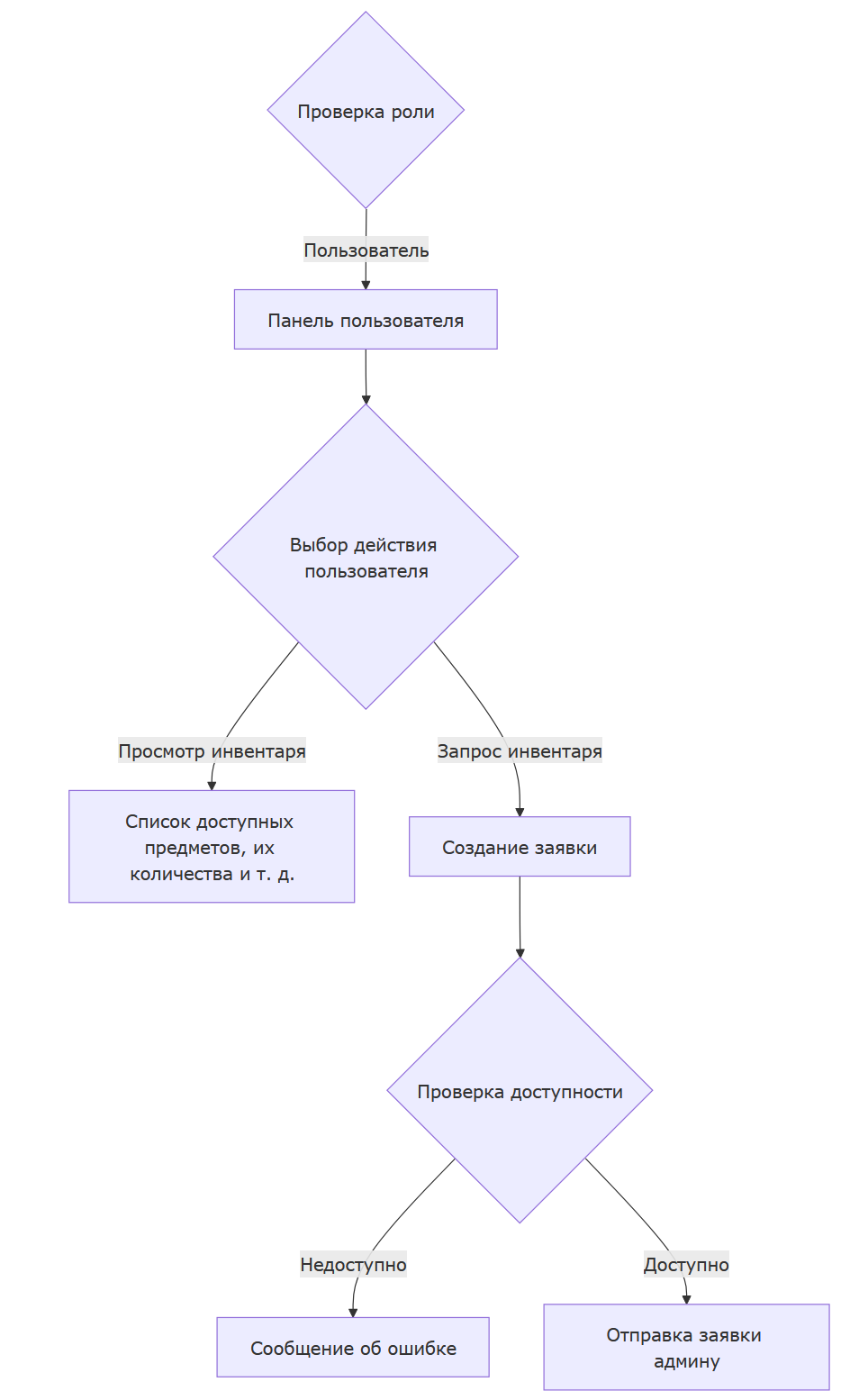


Рис. 3

Аргументация выбранного типа СУБД

SQLite - это легковесная, самодостаточная, серверная, транзакционная СУБД, встроенная в C-библиотеку. Она предоставляет все основные функции реляционной СУБД, но при этом обладает рядом особенностей, делающих её особенно удобной для использования с Flask, особенно в определённых сценариях.

SQLite не требует отдельного серверного процесса для работы. Вместо этого, вся база данных хранится в одном файле на диске, к которому приложение получает прямой доступ.

Для работы с SQLite не требуется установка, настройка или администрирование. Просто создайте файл базы данных, и можно начинать работать.

Библиотека SQLite имеет небольшой размер и потребляет мало ресурсов, что делает её подходящей для встраиваемых систем и мобильных устройств.

Для использования SQLite с Flask требуется минимальная настройка. Достаточно установить библиотеку sqlite3 (если она еще не установлена) и указать путь к файлу базы данных в конфигурации Flask.

Flask-приложение напрямую подключается к файлу базы данных SQLite, что упрощает взаимодействие с СУБД.

Несмотря на свои преимущества, SQLite имеет и ограничения:

SQLite имеет ограниченную поддержку параллельных операций записи. Если приложение требует высокой степени параллелизма, то лучше выбрать другую СУБД.

SQLite не подходит для высоконагруженных приложений, требующих масштабируемости.

SQLite не поддерживает некоторые расширенные функции, доступные в других СУБД (например, хранимые процедуры, триггеры, сложные типы данных.)

SQLite не предназначен для прямого доступа через сеть. Если требуется доступ к базе данных из нескольких удаленных приложений, следует использовать другую СУБД.

Важно учитывать ограничения и выбирать другую СУБД, если ваш проект требует высокой производительности, масштабируемости, расширенных функций или многопользовательского доступа. Выбор SQLite часто является разумным компромиссом между простотой и функциональностью.

Схема базы данных

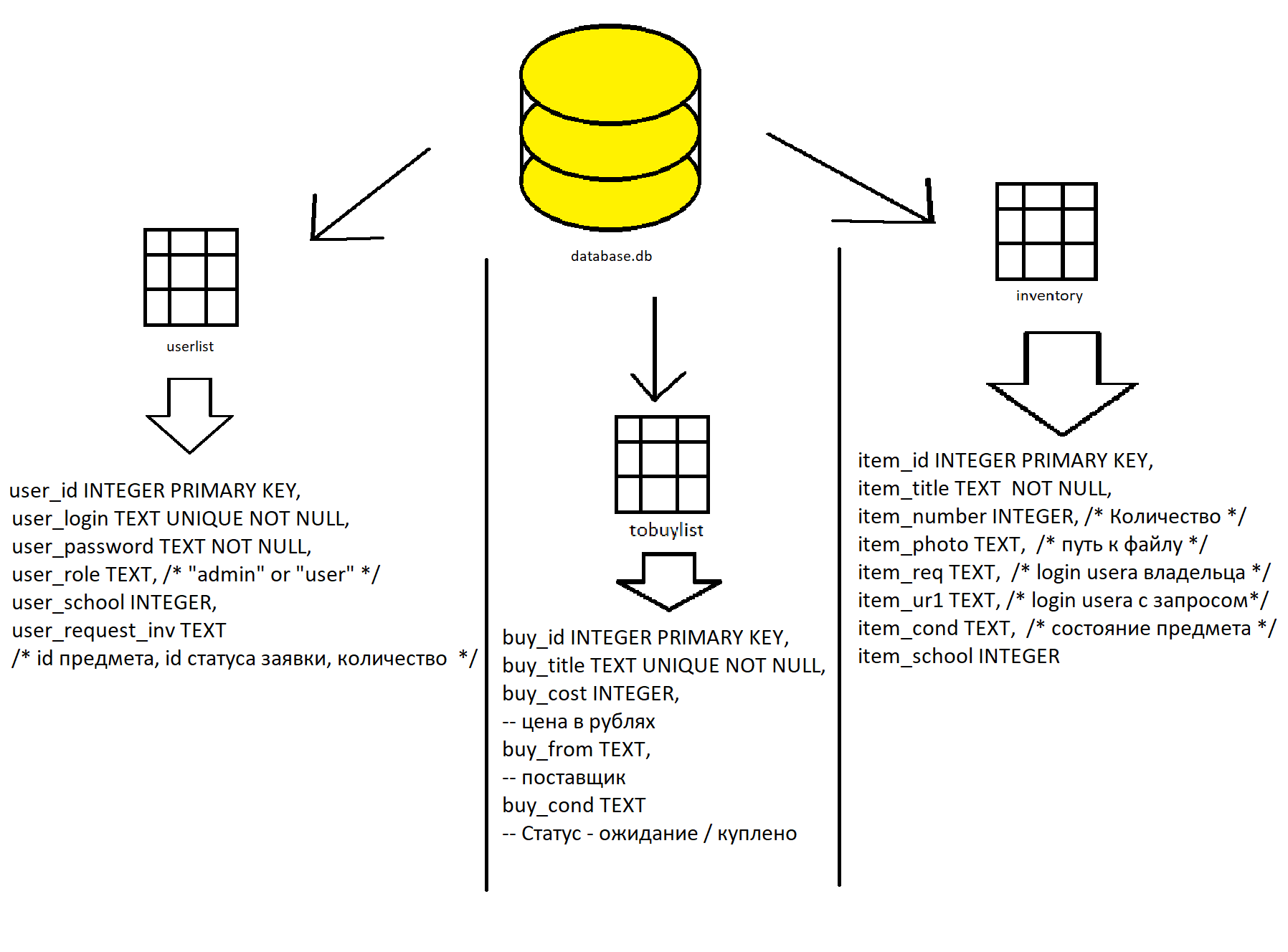


Рис. 4

Литература

<https://flask.palletsprojects.com/en/stable/>

<https://www.w3schools.com/>

<https://developer.mozilla.org/en-US/docs/Web>

<https://www.sqlitetutorial.net>

Ссылка на репозиторий:

<https://github.com/DonPolloEnjoyer/Predprof-Project.git>