Geekbrains

Frontend-разработчик

**Разработка приложения для создания и хранения резюме**

Frontend-программист

Цифровые профессии

Кутуева Ольга Сергеевна

Санкт-Петербург

2024

Содержание

[Введение 3](#_Toc160202674)

[Инструменты, использованные в работе 4](#_Toc160202675)

[Проектирование программной архитектуры 8](#_Toc160202676)

[Общая архитектура системы 8](#_Toc160202677)

[User Stories 11](#_Toc160202678)

[Технологии, планируемые к использованию в проекте 17](#_Toc160202679)

[MySQL 17](#_Toc160202680)

[Docker 18](#_Toc160202681)

[Изучение выбранных технологий, подготовка, эксперименты 21](#_Toc160202682)

[Запуск mysql в Docker 21](#_Toc160202683)

[Развитие проекта 23](#_Toc160202684)

[Список литературы 24](#_Toc160202685)

# Введение

Дипломный проект представляет собой онлайн-конструктор резюме — это инструмент, позволяющий пользователю создавать резюме в электронном виде, а затем отправить ссылку на резюме работодателю. В электронном резюме можно хранить информацию об образовании, опыте работы, навыках и достижениях. Онлайн-конструктор может быть полезны для людей, которые ищут новую работу или хотят обновить свое резюме, он позволяет сэкономить время и усилия на создание резюме, а также предоставляет возможность проверить и отредактировать уже готовое резюме перед отправкой.

**Цель проекта:** освоить полный цикл разработки программного обеспечения, пройти все стадии – от идеи до готового продукта. Разобраться, как реализовывается взаимодействие серверной части (backend), базы данных и клиентской части (frontend) приложения. Для этой цели разработать онлайн-конструктор резюме, который будет включать в себя все вышеуказанные компоненты.

**План работы:**

1. Определить цели и задачи проекта: какие функции должен выполнять конструктор резюме, какие возможности он должен предоставлять пользователям,
2. Разработать архитектуру системы — из каких частей будет состоять система, как они будут связаны между собой,
3. Описать инструменты и технологии для разработки,
4. Изучить описанные технологии и компоненты на уровне простых экспериментов,
5. Реализация готового продукта.

# Инструменты, использованные в работе

1. Хранение проекта: GitHub

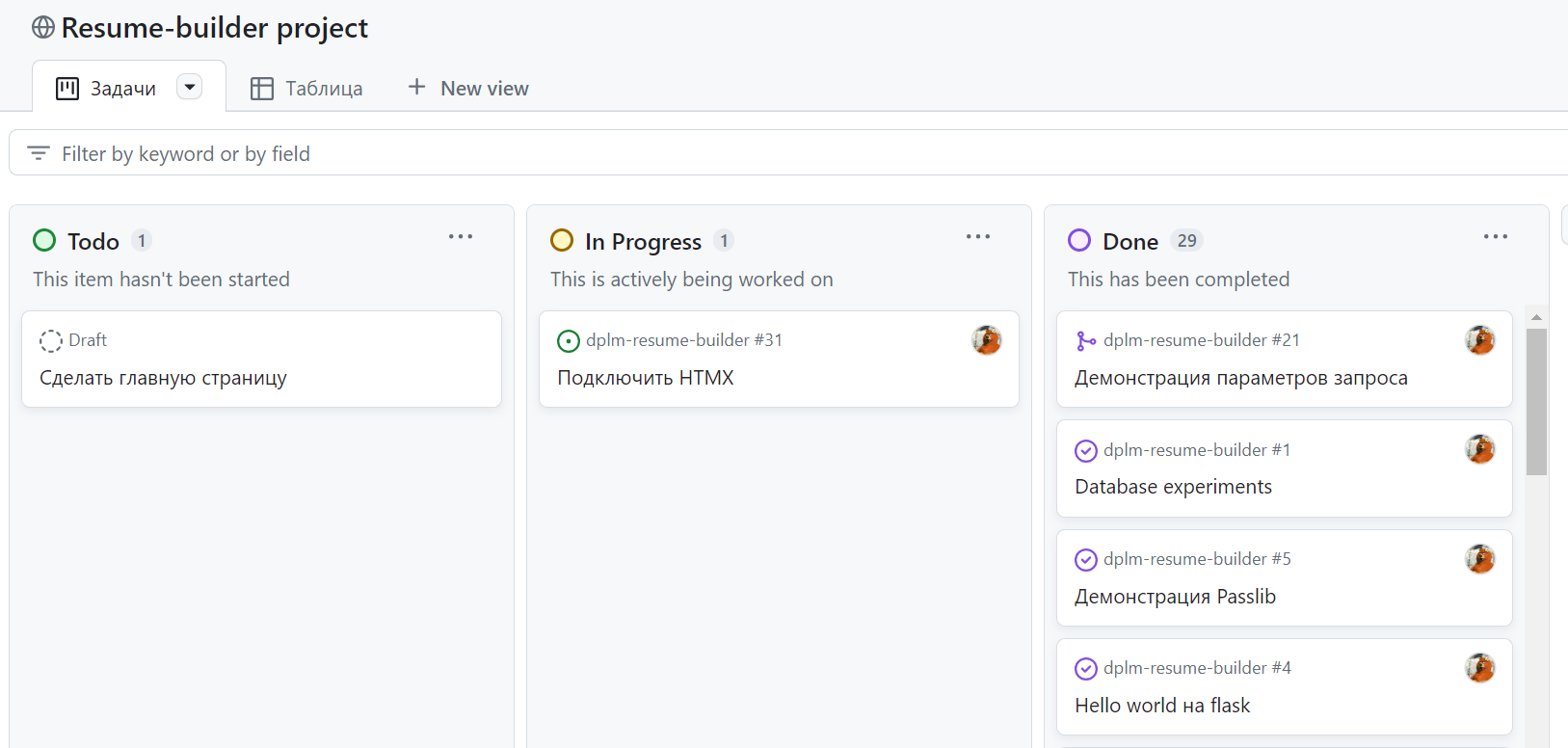
GitHub – это веб-платформа для хостинга и совместной работы над проектами с использованием системы контроля версий Git. Она предоставляет возможность разработчикам хранить, управлять и отслеживать свой код, а также обеспечивает средства для совместной разработки, управления задачами и обсуждения изменений.

Ссылка на репозиторий: <https://github.com/she-bear/dplm-resume-builder>

2) Управление проектом: GitHub Projects

GitHub Projects – это инструмент, предоставляемый самим GitHub для управления проектами и организации задач. Он позволяет создавать задачи, устанавливать приоритеты, управлять карточками задач, а также отслеживать прогресс разработки проекта. GitHub Projects интегрируется с репозиториями на GitHub, что позволяет удобно связывать код и задачи.

Ссылка на GitHub Projects проекта: https://github.com/users/she-bear/projects/1



3) Средство разработки (IDE): Visual Studio Code с WSL

VSCode (Visual Studio Code) – это бесплатный редактор с открытым исходным кодом. Он предоставляет широкие возможности для разработки программного обеспечения, включая поддержку многих языков программирования, инструменты для отладки, автоматизацию задач, интеграцию с системами контроля версий и многие другие функции.

VSCode отличается наличием обширной базы расширений, которые позволяют настраивать и расширять функционал редактора под нужды разработчика. Он также обладает удобным интерфейсом, быстрым запуском и отзывчивостью.

Преимущества использования в данном проекте:

1. интеграция с GitHub;
2. обширная поддержка для языка Python, включая интегрированный интерпретатор, подсветку синтаксиса, автодополнение кода и другие функции, упрощающие разработку;
3. возможность установки различных расширений, которые облегчают разработку на Python и Flask, такие как поддержка виртуальных окружений, отладка кода и работа с базами данных;
4. предоставление доступа к командной строке напрямую из интерфейса, что удобно для запуска сервера Flask, установки зависимостей и управления виртуальными окружениями;
5. поддержка markdown;
6. поддержка работы с Node.JS;
7. поддержка отладчика для Docker, что делает возможным отладку контейнеров Docker (для запуска mysql, например).

WSL (Windows Subsystem for Linux) – это среда подсистемы для запуска систем Linux внутри операционной системы Windows. WSL предоставляет возможность разработчикам выполнять команды и запускать утилиты Linux в среде Windows без необходимости установки фактической операционной системы Linux. Также, WSL позволяет использовать Docker-контейнеры для запуска приложений на Linux-серверах, что делает разработку более удобной и быстрой.

Visual Studio Code (VSCode) с Windows Subsystem for Linux (WSL) объединяет возможности редактора кода Visual Studio Code с возможностями Linux-среды, предоставляемой WSL.

4) Ведение документации: Notion

Notion – это платформа для организации, планирования и совместной работы над проектами, задачами и документами. Она предоставляет инструменты для создания различных типов контента – от заметок и списков до баз данных и досок задач. Notion позволяет пользователям создавать персонализированные рабочие пространства и организовывать информацию по своим потребностям.

В данном проекте Notion использовался как для ведения документации, так и для совместной работы (обмен информацией, заметками, уведомлениями и комментариями с руководителем проекта).

5) Совместное планирование и обсуждение: Excalidraw

Excalidraw – это открытое и свободное программное обеспечение для создания векторных диаграмм и чертежей. Он предоставляет интуитивный интерфейс и возможность быстро и легко создавать простые или сложные изображения, отображая структуру, процессы или концепции.

В данном проекте для обсуждения использовались разделяемые доски, и также экспорт и импорт диаграмм в различные форматы.

Разработка велась под руководством наставника: Яковлева Владимира Вячеславовича, что позволило дополнительно освоить технологию командной разработки (работа с Issues в GitHub Projects, CodeReview в Git).

# 

# Проектирование программной архитектуры

Для начала определим общую структуру и организацию системы, ее компоненты, их взаимосвязь и взаимодействие. Это позволит учесть основные аспекты проектирования до начала разработки и обеспечить возможность масштабирования и изменения в будущем.

Для этого рассмотрим укрупненную архитектуру системы, user stories (описание пользовательских историй) и er-диаграмму (диаграмму сущность-связь). Каждый из этих аспектов представляет собой способ представления или описания системы с разных точек зрения.

## Общая архитектура системы

Архитектура системы при разработке программного обеспечения представляет собой структурное описание программной системы, включающее в себя описание ее компонентов, их взаимодействие и отношения. Она определяет общий механизм работы системы, ее основные компоненты и их взаимодействие, а также описывает принципы, на основе которых происходит создание программного обеспечения.

Архитектура системы обычно включает в себя высокоуровневые диаграммы, описывающие структуру системы, ее модули и связи между ними, а также принципы, на которых основано разделение системы на компоненты и их взаимодействие.

Формирование архитектуры системы является одним из первых и самых важных этапов в разработке программного обеспечения, поскольку она определяет общее строение системы, которое в дальнейшем будет использоваться для разработки, тестирования и поддержания программного продукта.

Архитектурная структура web-приложения обычно состоит из четырех основных частей, каждая из которых будет входить в разрабатываемый проект (см. рис. 1):

**Backend (серверная часть)** – это часть приложения, которая отвечает за обработку данных и выполнение логики. Она обычно работает на стороне сервера и взаимодействует с frontend через API. Backend может включать в себя различные компоненты, такие как серверы приложений, базы данных и сервисы интеграции.

**Сервер** – это аппаратное или виртуальное устройство, которое обеспечивает вычислительные ресурсы для работы приложений. На сервере может работать один или несколько экземпляров backend-приложения. В зависимости от архитектуры приложения, сервер может также содержать базу данных или другие компоненты.

**База данных** – это система хранения данных, которая используется для хранения и управления информацией, необходимой для работы приложения. Backend-приложение обычно взаимодействует с базой данных через SQL или другие языки запросов. База данных может быть локальной или удаленной, и может находиться на том же сервере, что и backend, или на другом сервере.

**Frontend (клиентская часть)** – это интерфейс, который пользователь видит и с которым взаимодействует. Он обычно работает на стороне клиента и общается с backend через API для получения и отправки данных. Frontend может быть реализован в виде web-страницы, мобильного приложения или другого интерфейса, который позволяет пользователю взаимодействовать с приложением.

Выбор конкретных технологий для разработки отдельных частей системы не входит в рамки специальности и данного дипломного проекта, технологии были предложены дипломным руководителем.

|  |
| --- |
| Рис. 1 Общая архитектура системы |

Совместное использование сервера Nginx с базой данных MySQL и web-приложением, написанным на Python (Flask), представляет собой распространенную архитектурную модель для web-приложений:

**Nginx** используется в качестве web-сервера для обработки HTTP-запросов от клиентов и маршрутизации их к приложению Flask или статическим файлам. Nginx принимает запросы от клиентов и маршрутизирует их к приложению Flask на основе определенных правил маршрутизации.

База данных **MySQL** используется для хранения данных приложения и обеспечивает манипуляцию и управление данными, используемыми web-приложением Flask. Flask взаимодействует с базой данных MySQL для получения и обновления данных, необходимых для обработки запросов клиентов.

**Flask** – это микрофреймворк для Python, который используется для разработки web-приложений. Он обрабатывает HTTP-запросы от клиентов, взаимодействует с базой данных MySQL для получения и обновления данных, и возвращает ответы клиентам.

**Клиентские запросы** отправляются на сервер Nginx, Nginx получает **ответы** от приложения Flask и передает их обратно **клиентам.**

Как уже было указано выше, такая архитектурная модель является довольно распространенной и обеспечивает высокую производительность, при необходимости масштабируемость и безопасность для web-приложений, а также упрощает управление и конфигурацию различных компонентов. Кроме того, использование Nginx позволит легко добавить дополнительные сервисы и масштабировать приложение в случае его развития.

## User Stories

**User Stories** – это короткие описания функциональности, которую должно предоставить разрабатываемое программное обеспечение, рассказанные с точки зрения конечного пользователя.

В данном проекте User Stories помогут понять, какие функции действительно важны для пользователя и какие задачи пользователь будет выполнять, чтобы оценить функционал, который будет заложен в проект.

Диаграмма, используемая для описания User Stories, называется “Карта истории пользователя” (User Story Map). Такая диаграмма помогает визуализировать, как пользователь будет взаимодействовать с продуктом, и позволяет увидеть все User Stories в контексте реального использования продукта.

Каждая User Story обычно состоит из трех основных элементов:

* Кто (Who): определенный пользователь или роль, который будет пользоваться программным продуктом.
* Что (What): описывает конкретную функциональность или задачу, которую пользователь хочет выполнять.
* Зачем (Why): обоснование или цель того, почему пользователю нужна эта функциональность.

«Как [тип клиента], [хочу то-то], [чтобы делать что-то]».

В случае разрабатываемого конструктора резюме можно выделить две роли “who” – Пользователь и HR-специалист.

1. Я, как пользователь, хочу иметь профессионально оформленное резюме, ссылку на которое могу отправить для соискания работы.
   1. Я, как пользователь, хочу иметь возможность зарегистрироваться на сайте (US001).
   2. Я, как пользователь, хочу иметь возможность создать одно или несколько резюме (US003).
   3. Я, как пользователь, хочу войти на сайт, чтобы получить доступ к своим резюме (US002).
   4. Я, как пользователь, хочу иметь возможность отредактировать ранее созданное резюме, чтобы получить новый вариант резюме.
   5. Я, как пользователь, хочу удалить ранее созданное резюме.
   6. Я, как пользователь, хочу получить ссылку на резюме, чтобы отправить её для публичного просмотра.
2. Я, как HR-специалист, хочу открыть ссылку на резюме, чтобы просмотреть резюме соискателя.

**Список действий пользователя:**

* Регистрация (запрос логина и пароля)
* Авторизация (запрос логина и пароля)
* Получение списка резюме
* Создание резюме
* Редактирование резюме
* Удаление резюме
* Получение ссылки для публичного просмотра резюме

Список действий HR-специалиста:

* Получение резюме по прямой ссылке

На рисунке 2 приведена диаграмма User Stories для данного проекта.

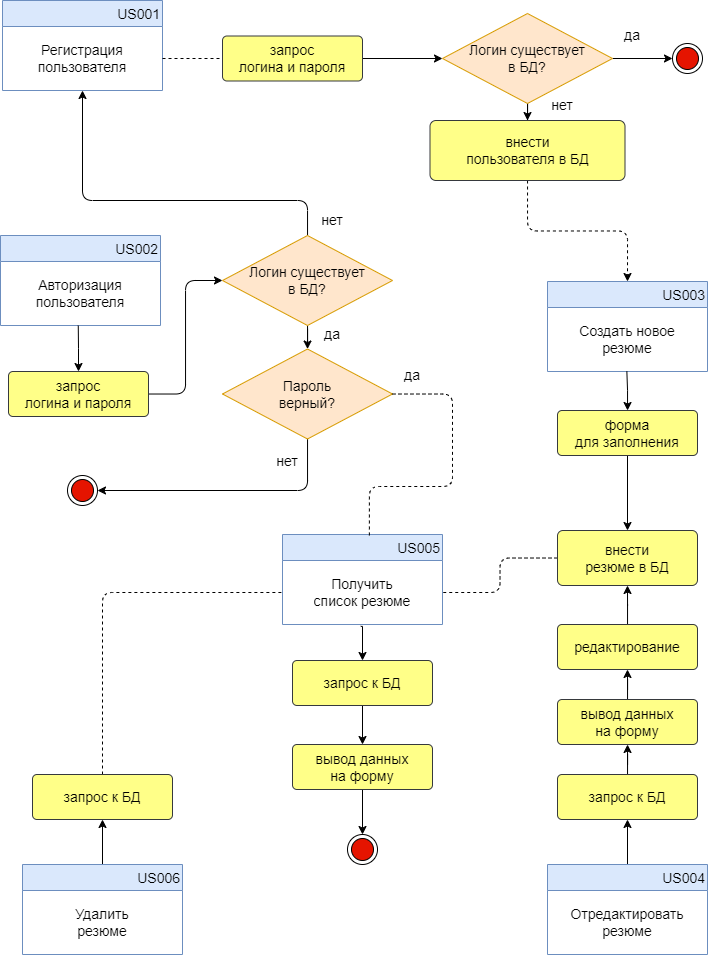


Рис. 2 Диаграмма User Stories

**Er-диаграмма**

ER-диаграммы (Entity-Relationship diagrams) – это графическое представление структуры данных в информационной системе.

При разработке web-приложений ER-диаграммы играют важную роль. Они помогают определить структуру базы данных, которая будет использоваться приложением. ER-диаграммы позволяют описать сущности, их атрибуты и связи между ними, что помогает понять логику работы приложения и организовать данные эффективным образом.

Опираясь на user stories посмотрим, как может быть организовано хранение данных в контексте нашего приложения.

Итак, есть пользователь (User), для которого системе требуется хранить одно или несколько резюме. Пользователь будет иметь уникальный идентификатор (ID), логин (Login) и пароль (Password). Резюме пользователя также будет иметь уникальный идентификатор (ID), заголовок (Title) и текст резюме (Text).

Отношения сущностей представлены на рисунке 3.

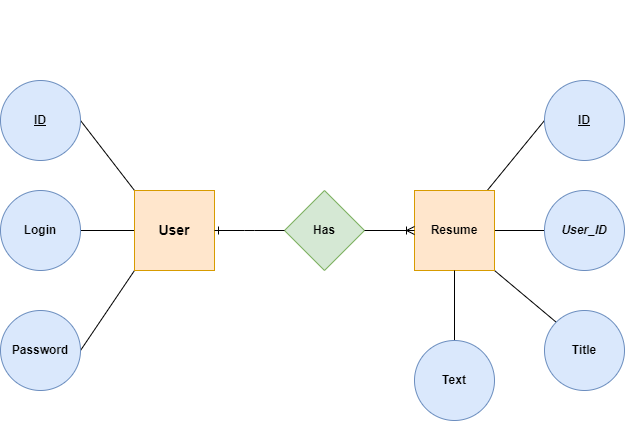


Рисунок 3. Отношение сущностей

На основе диаграммы отношения сущностей можем перейти к ERD-диаграмме базы данных, которая будет является визуальным представлением структуры базы данных, состоящей из таблиц (entities), связей между таблицами (relationships) и атрибутов (attributes), определенных для каждой таблицы (см. рисунок 4).

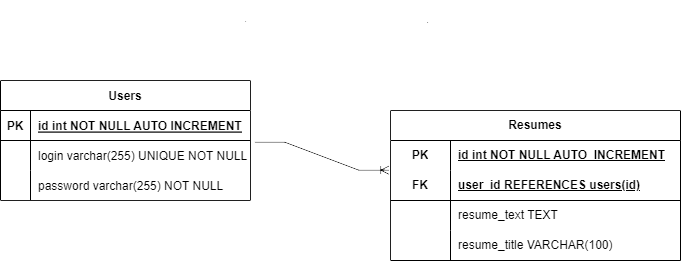


Рисунок 4. ERD-диаграмма базы данных

В дальнейшем, на основе этой структуры перейдем к непосредственному созданию таблиц базы данных на языке SQL, запросов и манипулирования данными.

# Технологии, планируемые к использованию в проекте

## MySQL

MySQL – это одна из самых популярных систем управления реляционными базами данных (СУБД), которая широко используется в web-разработке. MySQL предоставляет ряд функций и возможностей:

1. Открытый исходный код: MySQL имеет открытый исходный код, что означает, что его можно использовать бесплатно и без ограничений. Это делает его доступным для всех разработчиков и организаций.
2. Многоверсионность: MySQL поддерживает множество версий и редакций, что позволяет пользователям выбирать наиболее подходящую версию для своих нужд.
3. Масштабируемость: MySQL может масштабироваться от небольших проектов до крупных корпоративных систем с большим количеством пользователей.
4. Производительность: MySQL использует оптимизатор запросов, который автоматически оптимизирует запросы к базе данных для обеспечения максимальной производительности.
5. Безопасность: MySQL имеет встроенные функции безопасности, такие как шифрование данных, аутентификация пользователей и защита от атак на уровне приложений.

Причины выбора MySQL для данного проекта:

1. Открытый исходный код,
2. MySQL основан на реляционной модели данных (данные организованы в виде таблиц, связанных друг с другом), что как раз подходит для разработанной архитектуры системы (см. пп. Проектирование программной архитектуры),
3. Масштабируемость: в случае, если разрабатываемый проект в дальнейшем потребуется перевести на крупную базу данных, то MySQL справится с этой задачей,
4. Поддержка транзакций, обеспечение целостность данных и выполнение операций ACID (атомарность (A), согласованность (C), изоляция (I), долговечность (D)). При разработке не придется заботиться о контроле изменения данных, т.к. изменения данных, произведенные в результате выполнения одной транзакции, будут либо полностью применены, либо полностью отменены.
5. В данном проекте планируется использовать язык программирования Python c микрофреймворком Flask. Для совместного использования MySQL и Flask, MySQL предоставляет модуль mysql-connector-python, который обеспечит прямое подключение к базе данных MySQL из приложения Flask без использования ORM-библиотеки (сноска).

## Docker

Docker – это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры Docker позволяют упаковать приложение в единообразный набор, включая код, среду выполнения и все зависимости, необходимые для его работы, такие как библиотеки, файлы конфигурации и т.д. Это обеспечивает изолированное и повторяемое окружение, которое можно легко перемещать между различными средами разработки и развертывания, а также масштабировать в зависимости от потребностей.

Рассмотрим основные преимущества:

1. Изолированность: контейнеры Docker обеспечивают изоляцию приложений и их зависимостей, что позволяет запускать несколько приложений на одном хосте без конфликтов и вмешательства друг в друга. Это особенно полезно в средах разработки и развертывания, где могут использоваться различные версии языков программирования или библиотек.
2. Портативность: Docker-контейнеры могут быть легко перемещены между различными средами, такими как разработка, тестирование и развертывание.
3. Легковесность и быстрота: контейнеры Docker используют виртуализацию на уровне операционной системы, что делает их легкими и быстрыми в запуске.
4. Масштабируемость: Docker позволяет масштабировать приложения горизонтально путем запуска нескольких контейнеров с одним и тем же образом приложения.
5. Управление версиями и ресурсами: Docker позволяет управлять версиями приложений и их зависимостями с помощью Dockerfile, что упрощает процесс развертывания и обновления приложений.

Использование Docker в данном проекте обусловлено:

1. Для работы с базой данной планируется использовать MySQL Server – Docker обеспечит изоляцию базы данных и её зависимостей в контейнере, а также портативность базы данных при её развертывании. Это позволит избежать конфликтов с другими приложениями или сервисами, работающими на хост-системе. В случае необходимости вертикального (изменение ресурсов контейнера) или горизонтального (добавление дополнительных экземпляров контейнера) масштабирования, Docker также обеспечит необходимое управление нагрузкой на базу данных.
2. В данном проекте планируется использовать язык программирования Python с web-фреймворком Flask – Docker позволит упаковать приложение Python с его зависимостями (включая Flask и любые другие библиотеки, необходимые для его работы) в контейнер. Это обеспечит изолированное окружение, в котором приложение может быть запущено без необходимости установки и настройки зависимостей на хост-системе.
3. Веб-серверы: Docker поддерживает различные веб-серверы, такие как Apache, Nginx, IIS и другие, позволяя разработчикам легко настраивать и управлять ими.

# Изучение выбранных технологий, подготовка, эксперименты

Для последовательного изучения выбранных технологий, задача была разделена на небольшие части, каждой из которых соответствовала своя Issue в GitHub Projects (см. <https://github.com/users/she-bear/projects/1>). Порядок выполнения задач определялся наставником. По каждой задаче можно увидеть список подзадач и историю коммитов.

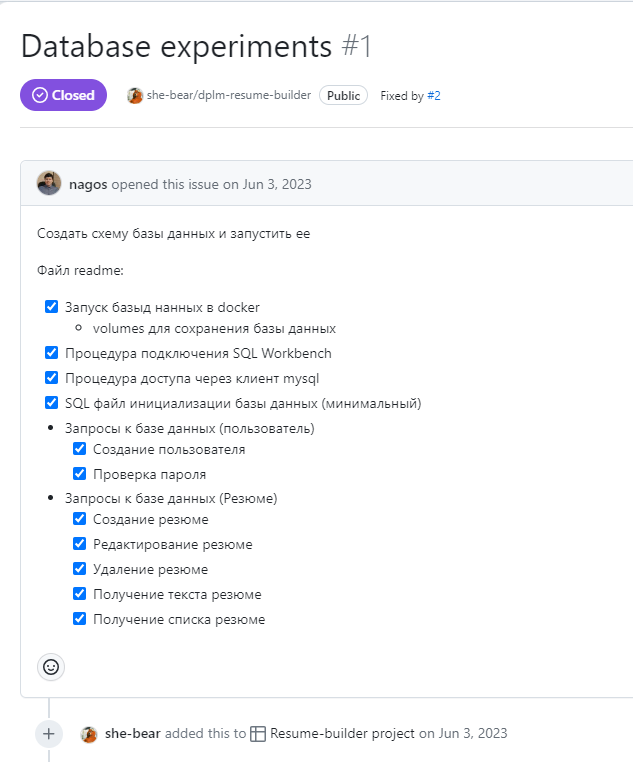


Рисунок N. Пример Issue из GitHub Projects

## Запуск mysql в Docker

Для запуска mysql в Docker необходимо:

Запустить приложение Docker Desktop на Windows,

Создать контейнер Docker:

Для создания контейнера в Docker необходимо разработать команду запуска с нужными для выполнения задачи параметрами.

Базовая команда для создания docker-контейнера выглядит следующим образом:

|  |
| --- |
| docker run *<имя контейнера>* |

Согласно документации по запуску mysql в docker, формируем и проверяем команды:

* Запустить mysql с доступом без пароля, удалить контейнер после окончания работы):

|  |
| --- |
| sudo docker run -d --rm -e MYSQL\_ALLOW\_EMPTY\_PASSWORD=true -p 3306:3306  mysql |

* Запусть с root-паролем (пароль задаем произвольный, для отладки):

|  |
| --- |
| sudo docker run --name=mysql -e MYSQL\_ROOT\_PASSWORD=123 -d -p 3306:3306  mysql |

Данная команда нужна для того, чтобы был доступ к контейнеру из командной строки:

|  |
| --- |
| docker exec -it mysql -p |

* Запустить контейнер и считать переменные среды из файла (имя файла произвольное, путь - относительно места запуска):

|  |
| --- |
| sudo docker run --name=mysql --env-file experiments/database/mysql.env -d -p 3306:3306 mysql |

Опции:

|  |
| --- |
| -d - запуск в режиме демона  --rm - удалить контейнер после использования (команда docker stop приведёт к УДАЛЕНИЮ контейнера!)  -e - установить переменные среды  -p - проброс портов (хост:контейнер)  --env-file - путь к файлу с переменными среды |

1. Работа из командной строки:

Работа с docker-контейнером MySQL из командной строки понадобится для развертывания, проверки и тестирования, т.к. на этапе экспериментов пока отсутствует другой доступ к данным внутри контейнера.

Согласно инструкции MySQL:

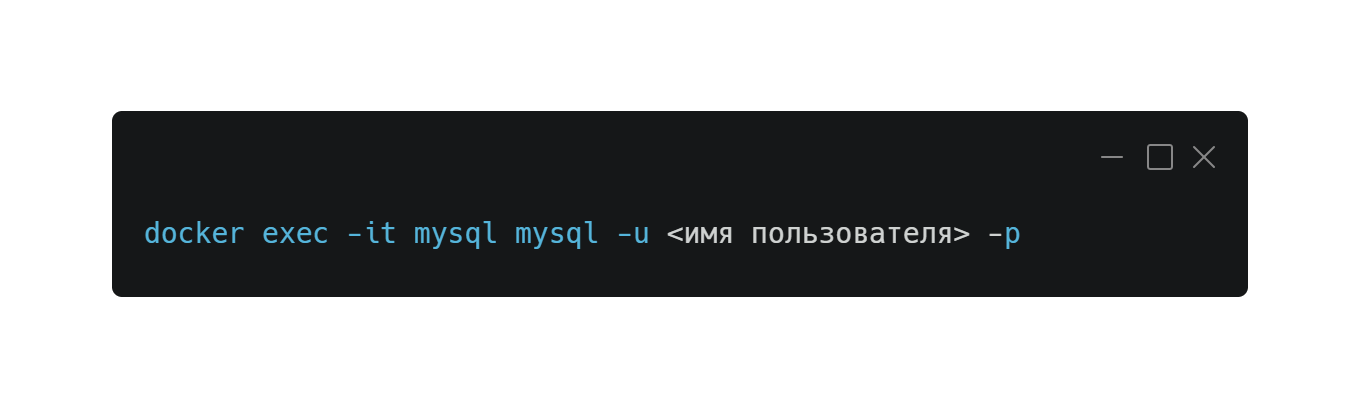
|  |
| --- |
| docker exec -it mysql |

Войти и запросить пароль для root-пользователя (для случая с MYSQL\_ROOT\_PASSWORD=123):

|  |
| --- |
| docker exec -it mysql -p |

Войти и запросить пароль для заданного пользователя (вариант с env-файлом, будет рассмотрен далее, в пп. 4):

|  |
| --- |
| docker exec -it mysql mysql -u *<имя пользователя>* -p |



|  |
| --- |
|  |

# Развитие проекта

1. Добавление функциональности для роли “Администратор сайта”:
   1. реализация каскадного удаления
2. Масштабирование системы

# Список литературы

1. Документация по GitHub: <https://docs.github.com/ru>
2. Документация по GitHub Projects: https://docs.github.com/en/issues/planning-and-tracking-with-projects/learning-about-projects/about-projects
3. Работа с VSCode, возможности, основные понятия: <https://code.visualstudio.com/docs>
4. Документация по Notion: https://www.notion.so/help/guides/category/documentation
5. Архитектура системы: понятие, виды: <https://gb.ru/blog/arhitektuta-sistemy/>
6. User Stories: <https://www.visual-paradigm.com/learning/handbooks/agile-handbook/user-story.jsp>
7. Пользовательские истории с примерами и шаблоном: [https://www.atlassian.com/ru/agile/project-management/user-stories#](https://www.atlassian.com/ru/agile/project-management/user-stories)
8. Модель «сущность-связь»: <https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model>
9. Запуск mysql в docker: <https://hub.docker.com/_/mysql>
10. Установка переменных среды: <https://docs.docker.com/engine/reference/run/#env-environment-variables>
11. Список опций для docker run: https://docs.docker.com/engine/reference/commandline/run/