Частное учреждение образования

«Колледж бизнеса и права»

ОТЧЕТ

ПО ПРОИЗВОДСТВЕННОЙ ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКИ

ОП Т.093026

Руководитель практики от колледжа (Т.Г.Багласова)

Руководитель практики от (А.О.Высочанский)

предприятия

ООО «ВЭБ Технологии»

Обучающийся (В.А.Черняков)

2024

**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ОП Т.093026

Разраб.

Черняков В.А.

Провер.

Багласова Т.Г.

Реценз.

Н. Контр.

Утверд.

*Система управления проектами «Cukierka»*

Лит.

Листов

КБиП

34

У

[1 Организационно-функциональная структура предприятия подразделения, характеристики основных видов деятельности 3](#_Toc157980246)

[2 Должностные обязанности 4](#_Toc157980247)

[3 Программное обеспечение, используемое на предприятии 5](#_Toc157980248)

[4 Реализация индивидуального задания 6](#_Toc157980249)

[4.1 Исследование предметной области 6](#_Toc157980250)

[4.2 Проектирование модели 6](#_Toc157980251)

[4.3 Организация данных 7](#_Toc157980252)

[4.4 Концептуальный прототип 8](#_Toc157980253)

[4.5 Реализация функций 9](#_Toc157980254)

[4.6 Функциональное тестирование 10](#_Toc157980255)

[Выводы 12](#_Toc157980256)

[Список используемых источников 13](#_Toc157980257)

[Приложение А (обязательное) Текст модулей веб-приложения 14](#_Toc157980258)

[Приложение Б (обязательное) Результаты работы приложения 34](#_Toc157980259)

# Организационно-функциональная структура предприятия подразделения, характеристики основных видов деятельности

ООО "ВЭБ Технологии" представляет собой инновационную технологическую компанию, специализирующуюся в разработке программного обеспечения для различных предприятий. Основными клиентами являются компании из различных отраслей, что обуславливает разнообразие выпускаемого софта.

Структура компании организована в соответствии с принципами функциональной дифференциации. На вершине иерархии находится Генеральный директор, подчиненные ему отделы ответственны за различные аспекты деятельности.

Основные виды деятельности ООО "ВЭБ Технологии" включают в себя:

- разработка программного обеспечения для корпоративных заказчиков;

- интеграция информационных систем;

- консультационные услуги по внедрению IT-решений;

- техническая поддержка и сопровождение программных продуктов.

Практика осуществляется в отделе разработки, который является ключевым звеном в производственном процессе компании.

Подразделение включает в себя следующие должности:

- руководитель отдела разработки, отвечает за общее управление и координацию разработки программного обеспечения;

- программисты (разработчики, занимаются написанием и тестированием кода для создания программных продуктов;

- тестировщики, осуществляют проверку функциональности программного обеспечения, выявляют и устраняют ошибки;

- аналитики, занимаются анализом требований заказчика, выявляют ключевые особенности проекта.

Отдел разработки является частью департамента по инновационным технологиям, который напрямую подчинен исполнительному директору компании. Департамент сосредотачивает в себе ключевые компетенции в области информационных технологий.

Отдел разработки занимается созданием и совершенствованием программного обеспечения в соответствии с требованиями заказчиков. Он активно взаимодействует с другими отделами компании, такими как отделы аналитики и тестирования, для обеспечения высокого качества и функциональности разрабатываемых продуктов.

# Должностные обязанности

Для выполнения возложенных на него функций инженер-программист (программист) обязан разрабатывать программы на основе анализа математических моделей и алгоритмов решения различных задач. Эти задачи могут относиться к научным, прикладным, экономическим и другим областям. Важной частью его обязанностей является обеспечение возможности выполнения алгоритмов и поставленных задач с использованием средств вычислительной техники, а также отладка и тестирование разработанных программ.

Он также обязан разрабатывать технологию решения задачи на всех этапах обработки информации, выбирать язык программирования для описания алгоритмов и структур данных. Определение информации, подлежащей обработке, ее объемов, структуры, макетов и схем ввода, обработки, хранения и вывода также входит в его обязанности.

Инженер-программист выполняет работу по подготовке программ к отладке, определяет объем и содержание данных контрольных примеров, осуществляет запуск отлаженных программ и ввод исходных данных. После этого проводит тестирование и корректировку разработанных программ на основе анализа выходных данных.

Он разрабатывает инструкции по работе с программами, оформляет техническую документацию, осваивает и применяет новые компьютерные технологии. Также инженер-программист определяет возможность использования готовых программных продуктов и осуществляет сопровождение внедренных программ и программных средств.

В его обязанности также входит разработка и внедрение систем автоматической проверки правильности программ, типовых и стандартных программных средств, а также составление технологии обработки информации. Он выполняет работу по унификации и типизации вычислительных процессов, принимает участие в создании каталогов и картотек стандартных программ, а также в разработке форм документов, подлежащих машинной обработке, и программ, расширяющих область применения вычислительной техники.

# Программное обеспечение, используемое на предприятии

На предприятии широко используется разнообразное программное обеспечение, ориентированное на эффективную разработку, тестирование и поддержку программных продуктов. Важной составляющей инфраструктуры является операционная система, система контроля версий, язык программирования, система управления базой данных, инструменты для тестирования и другие средства, обеспечивающие полный жизненный цикл программного продукта

Операционная система Windows является основным средством поддержки для работы с компьютерами на предприятии. Эта операционная система предоставляет удобное и функциональное окружение для разработчиков и других сотрудников, обеспечивая стабильность и совместимость с различными прикладными программами.

Система контроля версий Git используется для эффективного управления исходным кодом проектов. Она обеспечивает возможность отслеживания изменений, совместной работы разработчиков, а также управления версиями программного обеспечения, что является ключевым элементом современного программирования.

Прикладной язык программирования Java используется для создания масштабируемых и кроссплатформенных приложений. Этот язык предоставляет широкий спектр инструментов для разработки, обеспечивает высокую производительность и широкую совместимость с различными технологиями.

Система управления базой данных PostgreSQL является выбором для хранения и обработки данных на предприятии. Она отличается открытым исходным кодом, высокой производительностью и надежностью, что делает ее предпочтительным решением для проектов с различными масштабами и сложностью.

Инструмент Postman используется для тестирования API-интерфейсов, обеспечивая автоматизированное тестирование и проверку функциональности взаимодействия между различными компонентами программного обеспечения.

Система тестирования JUnit предоставляет средства для модульного тестирования кода на языке Java. Она позволяет создавать и запускать тесты, обеспечивая контроль качества разрабатываемых программных продуктов.

Таким образом, применение указанных программных средств на предприятии обеспечивает надежную и эффективную основу для разработки, тестирования и сопровождения программных продуктов, соответствуя высоким стандартам современной индустрии информационных технологий.

# Реализация индивидуального задания

## Исследование предметной области

В контексте месте прохождения практики было решено реализовать программный продукт для предприятия ООО "ВЭБ Технологии" с целью управления проектами в рамках бизнес-среды. Проектируемое приложение предназначено для эффективного ведения проектов, предполагающих активное взаимодействие сотрудников и систематизацию задач.

Инструментальным средством данного приложения будет веб-сайт, обеспечивающий сотрудникам возможность создания и управления доской задач. Эта доска, представляющая собой специальную таблицу, позволяет каждому участнику процесса определить и структурировать задачи, которые требуется выполнить. Дополнительно, рассматривается вариант использования багтрекера, предназначенного для систематизации и учета выявленных ошибок.

Важным этапом в разработке данного программного продукта является проведение исследования предметной области. Это включает в себя анализ существующих задач, подлежащих автоматизации, и обоснование необходимости компьютерной обработки информации для повышения эффективности процессов. Также рассматриваются формы выходных документов, такие как договоры, счета, счет-фактуры и отчеты, предоставляя возможность визуализации структуры выходной информации.

Описывается текущий процесс решения задачи в организации, а также периодичность использования разрабатываемого программного продукта. В контексте упрощения решения задачи выделяется функциональность доски задач и багтрекера, предоставляющих наглядные и структурированные средства взаимодействия и мониторинга задач.

Помимо этого, проводится анализ существующих аналогов, выявляя особенности и преимущества разрабатываемого приложения. Этот этап позволяет лучше понять уникальные черты предлагаемого решения и определить его конкурентоспособность на рынке бизнес-приложений.

## Проектирование модели

На первом этапе проектирования необходимо определить объекты, которые будут взаимодействовать в рамках создаваемого приложения. Это включает в себя доску задач, баг-трекер, пользователей и другие элементы, составляющие функциональную структуру системы. Каждый объект будет иметь свои характеристики и роль в процессе управления проектами.

Для более наглядного представления функциональных требований системы, будет построена диаграмма вариантов использования. Эта диаграмма позволит систематизировать различные сценарии использования приложения, выделяя ключевые варианты взаимодействия между пользователями и системой. Данный инструментарий дает возможность лучше понять потребности пользователей и обеспечить соответствие приложения их ожиданиям.

Важным шагом в проектировании веб-приложения является разработка первоначальной структуры. Для этого используется организационный список или ментальная карта, предоставляя обзор основных разделов, функций и переходов между ними. Эта структура служит основой для дальнейшего развития и реализации веб-интерфейса, обеспечивая логичную и интуитивно понятную навигацию для пользователей.

Таким образом, процесс проектирования модели программного продукта включает в себя детализацию объектов, вариантов использования и первоначальной структуры, обеспечивая основу для последующей разработки и реализации.

## Организация данных

Логическая модель данных включает в себя описание структуры данных и их взаимосвязей на более высоком уровне абстракции. В контексте управления проектами, объектами логической модели могут быть, например, проекты, задачи, пользователи и т.д. Отношения между этими объектами представляют собой ключевые аспекты логической организации данных.

Физическая модель данных, в свою очередь, фокусируется на реальной структуре данных, их типах, индексации и отношениях на уровне базы данных. В контексте среды разработки, физическая модель может определять таблицы базы данных, их поля и связи между таблицами. Важным аспектом является оптимизация физической структуры для эффективного хранения и обработки данных.

Выходные данные, порождаемые системой, будут представлять собой информацию о проектах, статусах задач, пользователях и других аспектах управления проектами. Формат выходных данных будет зависеть от конкретных потребностей пользователей и возможностей системы. Это может включать в себя отчеты, графики, статистику и другие формы представления информации.

Таким образом, организация данных включает в себя построение логической и физической моделей данных, обеспечивающих эффективное и структурированное хранение информации, а также определение формата выходных данных для обеспечения полноценной информационной обратной связи с пользователями системы.

Для организации данных системы была использована система управления базами данных «Postgesql». В таблица 4.1 – 4.5 представлены организации таблиц данных.

Таблица 4.1 – Таблица данных «Users»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | UUID | Первичный ключ пользователя |
| username | VARCHAR | Имя пользователя (не может быть пустым) |
| email | VARCHAR | Электронная почта пользователя (не может быть пустой) |
| other | JSONB | Дополнительные данные пользователя в формате JSONB |

Таблица 4.2 – Таблица данных «Tokens»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | UUID | Первичный ключ токена |
| user\_id | UUID | Внешний ключ, ссылается на id пользователя в таблице Users |
| expires | TIMESTAMPTZ | Дата и время истечения срока действия токена (не может быть пустой) |
| other | JSONB | Дополнительные данные токена в формате JSONB |

Таблица 4.3 – Таблица данных «Projects»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | UUID | Первичный ключ проекта |
| name | VARCHAR | Название проекта (не может быть пустым) |
| applications | VARCHAR [] | Массив названий приложений, связанных с проектом (не может быть пустым) |
| other | JSONB | Дополнительные данные проекта в формате JSONB |

Таблица 4.4 – Таблица данных «**UsersToProjects**»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| user\_id | UUID | Внешний ключ, ссылается на id пользователя в таблице Users |
| project\_id | UUID | Внешний ключ, ссылается на id проекта в таблице Projects |
| user\_role | VARCHAR | Роль пользователя в проекте (не может быть пустой) |

Таблица 4.5 – Таблица данных «Bugs»

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Описание |
| id | UUID | Первичный ключ бага |
| name | VARCHAR | Название бага (не может быть пустым) |
| description | VARCHAR | Описание бага (не может быть пустым) |
| user\_id | UUID | Внешний ключ, ссылается на id пользователя в таблице Users |
| project\_id | UUID | Внешний ключ, ссылается на id проекта в таблице Projects |
| criticality | INT | Степень критичности бага (не может быть пустой) |
| priority | INT | Приоритет исправления бага (не может быть пустым) |
| status | VARCHAR | Статус бага (не может быть пустым) |
| other | JSONB | Дополнительные данные бага в формате JSONB |

## Концептуальный прототип

В системе меню концептуального прототипа выделяются ключевые разделы и функциональные блоки, обеспечивая легкий доступ пользователя к основным функциям приложения. Диалоговые окна представляют собой визуальные компоненты для ввода данных, отображения результатов и взаимодействия с системой. Элементы управления, такие как кнопки, поля ввода, выпадающие списки и т.д., акцентируют внимание на важных действиях пользователя.

Прототипы форм и диалоговых окон являются визуальным отображением концептуальных идей, предоставляя пользователям представление о том, как будет выглядеть интерфейс и какие возможности будут доступны. Для веб-приложений и сайтов прототипы включают в себя представление страниц и форм, определение расположения элементов, цветовой гаммы и структуры интерфейса. Один из прототипов представлен на рисунке 4.1.

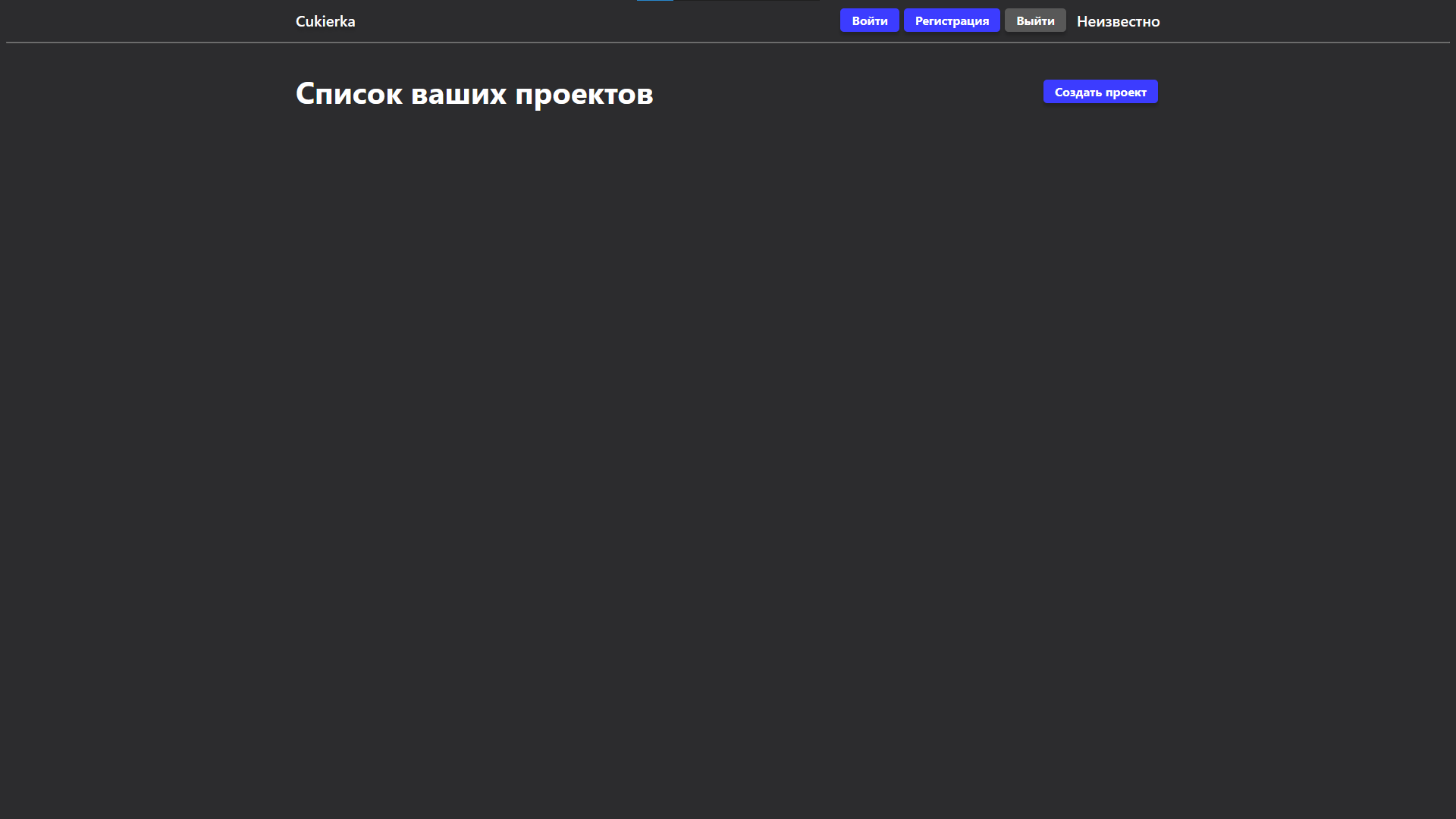


Рисунок 4.1 – Прототип главной страницы

Процесс разработки концептуального прототипа направлен на создание интуитивно понятного, удобного и привлекательного визуального интерфейса, способного эффективно взаимодействовать с пользователем. Каждый элемент прототипа тщательно обдумывается с учетом пользовательского опыта, удовлетворяя требованиям функциональности и эстетическим предпочтениям.

Таким образом, концептуальный прототип представляет собой важный этап в разработке программного продукта, обеспечивая предварительное представление о внешнем пользовательском интерфейсе и его функциональности.

## Реализация функций

Программные модули, представленные в тексте основных модулей, содержат комментарии, поясняющие их функциональность и указывающие на элементы управления, которые инициируют их выполнение. Это обеспечивает легкость восприятия кода разработчиками и обеспечивает понимание взаимосвязей между интерфейсом и функциональностью программы.

Реализация функций включает в себя взаимодействие с базой данных для эффективного хранения и извлечения данных, обновление информации на основе действий пользователя, а также визуализацию результатов в соответствии с задачами, решаемыми приложением.

Таким образом, в данном разделе представлены реализованные функции, которые непосредственно связаны с элементами управления, обеспечивая работу приложения и взаимодействие пользователя с системой управления проектами. Комментарии в тексте основных модулей содействуют легкости восприятия и дальнейшей поддержке разработанного программного продукта.

Пример реализации обработки запроса вызова API и последующей передачи управление функции:

public static JSONObject processMessage(String key, JSONObject message, App instance, String user) {

try {

Reflections reflections = new Reflections("by.vlobo.api.entity");

Set<Class<?>> annotatedClasses = reflections.getTypesAnnotatedWith(AApi.class);

System.out.println(key);

for (Class<?> clazz : annotatedClasses) {

if (IApiProcessor.class.isAssignableFrom(clazz)) {

IApiProcessor processor = (IApiProcessor) clazz.getDeclaredConstructor().newInstance();

;

return processor.process(message, instance, user);

}

}

} catch (InstantiationException | IllegalAccessException | NoSuchMethodException

| InvocationTargetException e) {

e.printStackTrace();

return IApiProcessor.CODE\_500\_INTERNAL\_SERVER\_ERROR;

}

return IApiProcessor.CODE\_500\_INTERNAL\_SERVER\_ERROR;

}

## Функциональное тестирование

Проведение тестирования сопровождается распечатками копий экранов, на которых отражены результаты выполнения определенных действий. Эти экраны служат визуальной демонстрацией работы программного продукта на различных этапах тестирования. Результаты работы программного средства, включая экранные формы, отчеты и другие выходные документы, предоставлены в приложении Б и таблице 4.6.

Тест-кейсы охватывают различные сценарии использования программы, включая основные функции управления проектами, взаимодействие с базой данных, а также корректное отображение информации на пользовательском интерфейсе. В процессе тестирования проверяется не только функциональность, но и устойчивость системы к возможным ошибкам и сценариям непредвиденного использования.

Таким образом, функциональное тестирование поддерживается тщательно разработанными тест-кейсами, визуальными материалами и результатами выполнения, обеспечивающими адекватную оценку работоспособности программного продукта.

Таблица 4.6 – Тест-кейсы

|  |  |  |
| --- | --- | --- |
| Модуль/функция | Шаги воспроизведения | Ожидаемый Результат |
| Вход в систему | 1. Открыть браузер.  2. Перейти на страницу входа.  3. Ввести корректные учетные данные.  4. Нажать кнопку "Войти". | Успешный вход в систему, перенаправление на главную страницу. |
| Регистрация нового пользователя | 1. Открыть страницу регистрации.  2. Заполнить обязательные поля (имя, электронная почта, пароль).  3. Нажать кнопку "Зарегистрироваться". | Успешная регистрация, отображение сообщения об успешной регистрации. |
| Создание нового проекта | 1. Войти в систему.  2. Перейти на страницу управления проектами.  3. Нажать кнопку "Создать проект".  4. Заполнить поля (название, описание).  5. Нажать кнопку "Сохранить". | Новый проект успешно создан, отображение его в списке проектов. |
| Добавление участника в проект | 1. Войти в систему с учетной записью администратора проекта.  2. Открыть страницу управления участниками проекта.  3. Нажать кнопку "Добавить участника".  4. Ввести данные нового участника.  5. Нажать кнопку "Добавить". | Участник успешно добавлен в проект, отображение его в списке участников проекта. |
| Создание нового бага | 1. Войти в систему.  2. Выбрать проект.  3. Перейти на страницу управления багами.  4. Нажать кнопку "Создать баг".  5. Заполнить поля (название, описание, критичность, приоритет).  6. Нажать кнопку "Создать". | Новый баг успешно создан, отображение его в списке багов проекта. |
| Завершение работы пользователя | 1. Выйти из системы, нажав кнопку "Выйти" или закрыв браузер. | Успешное завершение сеанса, перенаправление на страницу входа. |

# Выводы

В ходе прохождения технологической практики в ООО "ВЭБ Технологии" было активно использовано разнообразное программное обеспечение, отражающее современные требования к разработке и поддержке программных продуктов.

Программное обеспечение, сопровождавшее процесс практики, включало операционную систему Windows, систему контроля версий Git, язык программирования Java, систему управления базой данных PostgreSQL, инструмент Postman для тестирования API, а также систему тестирования JUnit. Этот набор инструментов предоставил обширные возможности для полного цикла разработки, начиная от написания кода и его версионного контроля до тестирования и управления данными.

Практика также привнесла новые знания и навыки в области разработки программного обеспечения. В процессе работы с языком программирования Java были углублены знания по объектно-ориентированному программированию, а использование системы управления базой данных PostgreSQL расширило понимание организации и хранения данных. Опыт работы с инструментами тестирования, такими как Postman и JUnit, позволил освоить методологии тестирования и обеспечения качества разрабатываемого софта.

Важной частью практики стала реализация поставленной задачи. В ходе технологической практики была успешно реализована задача, связанная с разработкой программного модуля на языке Java и его интеграцией с базой данных PostgreSQL. Результаты работы были внедрены в рабочую среду, что подтверждает успешное взаимодействие разработанных компонентов с общей системой предприятия.

Таким образом, технологическая практика в ООО "ВЭБ Технологии" предоставила ценный опыт работы с современными технологиями разработки программного обеспечения и позволила значительно расширить знания и навыки в этой области. Реализация поставленной задачи внесла практический вклад в профессиональное развитие и подтвердила успешное освоение полученных знаний.

# Список используемых источников

1 Багласова, Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2017. – 30 c.

2 Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.

# Приложение А (обязательное) Текст модулей веб-приложения

package by.vlobo;

import com.sun.net.httpserver.HttpServer;

import by.vlobo.api.IApiProcessor;

import com.sun.net.httpserver.HttpHandler;

import com.sun.net.httpserver.HttpExchange;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.net.InetSocketAddress;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import org.json.JSONException;

import org.json.JSONObject;

public class Server {

public Server(App app) {

try {

HttpServer server = HttpServer.create(new InetSocketAddress(8080), 0);

server.createContext("/", new ServerHandler(app));

server.setExecutor(null);

server.start();

} catch (JSONException e) {

e.printStackTrace();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}

class ServerHandler implements HttpHandler {

private App app;

public ServerHandler(App app) {

this.app = app;

}

@Override

public void handle(HttpExchange t) throws IOException {

String path = t.getRequestURI().getPath();

System.out.println(path);

for (Pattern pattern : app.getRoadmap().keySet()) {

Matcher matcher = pattern.matcher(path);

if (matcher.matches()) {

HashMap<String, String> vars = new HashMap<>();

for (int i = matcher.groupCount(); i > 0; i--) {

// Увы, нельзя обратно получить название группы, поэтому пока такой костыль. В

// будущем мб заменю

vars.put(new String("" + i), matcher.group(i));

}

String road = app.getRoadmap().get(pattern);

switch (road) {

case "#":

apiHandle(t, app, vars);

break;

case "$":

cdnHandle(t, vars);

break;

default:

wwwHandle(t, app, vars, road);

break;

}

break;

}

}

String notFoundResponse = "File not found";

t.getResponseHeaders().set("Content-Type", "text/html; charset=UTF-8");

Tools.sendString(t, 404, notFoundResponse);

}

public void wwwHandle(HttpExchange t, App instance, HashMap<String, String> vars, String road) {

try {

String response = Tools.linkHtml(road, vars);

t.getResponseHeaders().set("Content-Type", "text/html; charset=UTF-8");

Tools.sendString(t, 200, response);

} catch (JSONException | IOException e) {

e.printStackTrace();

}

}

public void apiHandle(HttpExchange t, App instance, HashMap<String, String> vars) {

String user = t.getRequestHeaders().getFirst("Authorization");

try {

JSONObject response;

try {

if (user != null) {

String uuidPattern = "([0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12})|"

+

"([0-9a-fA-F]{32})";

Matcher matcher = Pattern.compile(uuidPattern).matcher(user);

if (matcher.find()) {

user = instance.getDatabase().getUserByToken(matcher.group(0)).getString("user\_id");

} else {

user = null;

}

}

JSONObject message = new JSONObject(new String(t.getRequestBody().readAllBytes()));

response = Tools.processMessage(vars.get("1"), message, instance, user);

} catch (JSONException e) {

e.printStackTrace();

response = IApiProcessor.CODE\_400\_BAD\_REQUEST;

}

System.out.println(response);

String responseStr = response.toString(4);

t.getResponseHeaders().set("Content-Type", "application/json; charset=UTF-8");

Tools.sendString(t, response.getInt("code"), responseStr);

} catch (JSONException | IOException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

}

public void cdnHandle(HttpExchange t, HashMap<String, String> vars) throws IOException {

String filePath = "./" + vars.get("1");

System.out.println(filePath);

File file = new File(filePath);

if (file.isDirectory()) {

ArrayList<String> fileNames = new ArrayList<>();

for (File f : file.listFiles()) {

fileNames.add(

"<a href=\"./" +

f.getName() +

(f.isDirectory() ? "/" : "") +

"\">" + f.getName() + "</a>");

}

String response = String.join("<br>", fileNames);

Tools.sendString(t, 200, response);

} else if (file.exists() && file.isFile()) {

t.getResponseHeaders().set("Content-Type", "text/plain; charset=UTF-8");

Tools.sendFile(t, 200, file);

} else {

String notFoundResponse = "File not found";

Tools.sendString(t, 404, notFoundResponse);

}

}

}

package by.vlobo;

import com.sun.net.httpserver.HttpExchange;

import by.vlobo.api.AApi;

import by.vlobo.api.IApiProcessor;

import java.io.File;

import java.io.IOException;

import java.io.OutputStream;

import java.lang.reflect.InvocationTargetException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.time.ZonedDateTime;

import java.time.format.DateTimeFormatter;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Map;

import java.util.Set;

import java.util.UUID;

import java.util.regex.Pattern;

import java.util.stream.Collectors;

import org.json.JSONArray;

import org.json.JSONObject;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

import org.postgresql.util.PGobject;

import org.reflections.Reflections;

public class Tools {

public static String hashPassword(String plainPassword) {

try {

MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");

byte[] hashBytes = messageDigest.digest(plainPassword.getBytes());

// Преобразование байтов хеша в шестнадцатеричную строку

StringBuilder stringBuilder = new StringBuilder();

for (byte b : hashBytes) {

stringBuilder.append(String.format("%02x", b));

}

return stringBuilder.toString();

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

// Обработка ошибки

return null;

}

}

// Метод для проверки соответствия введенного пароля хешу

public static boolean checkPassword(String userInputPassword, String hashedPassword) {

String userInputHash = hashPassword(userInputPassword);

return hashedPassword.equals(userInputHash);

}

public static String randomUUID2() {

// Генерация UUID на основе имени (namespace-based UUID)

String namespace = UUID.randomUUID().toString();

String name = "cukierka";

UUID nameBasedUUID = UUID.nameUUIDFromBytes((namespace + name).getBytes());

return nameBasedUUID.toString();

}

public static JSONObject toJSONObject(ResultSet rs) {

try {

int columns = rs.getMetaData().getColumnCount();

JSONObject jsonObject = new JSONObject();

for (int i = 1; i <= columns; i++) {

String columnName = rs.getMetaData().getColumnName(i);

Object value = rs.getObject(i);

if (value instanceof org.postgresql.util.PGobject) {

PGobject pgObject = (PGobject) value;

jsonObject.put(columnName, new JSONObject(pgObject.getValue()));

} else {

jsonObject.put(columnName, value.toString());

}

}

return jsonObject;

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public static String formatDateTime(ZonedDateTime zonedDateTime) {

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ssXXX");

return zonedDateTime.format(formatter);

}

public static HashMap<Pattern, String> readRoadmap(JSONObject jsonObject) {

HashMap<String, String> hashMap = new HashMap<String, String>();

readRoadmapR(jsonObject, hashMap, "^");

HashMap<Pattern, String> patternHashMap = new HashMap<>();

for (Map.Entry<String, String> entry : hashMap.entrySet()) {

String key = entry.getKey();

String value = entry.getValue();

Pattern pattern = Pattern.compile(key);

patternHashMap.put(pattern, value);

}

return patternHashMap;

}

private static void readRoadmapR(JSONObject jsonObject, HashMap<String, String> result, String s) {

switch (jsonObject.getInt("type")) {

case 0: {// ничего

s += jsonObject.getString("i") + "\\/";

break;

}

case 1: {// обычный файл

s += jsonObject.getString("i") + "\\/";

result.put(s + "?$", jsonObject.getString("index"));

break;

}

case 2: {// api

s += String.format("(?<%s>[0-9a-zA-Z\\-]+)\\/", jsonObject.getString("i"));

result.put(s + "?$", "#");

break;

}

case 3: {// переменная

s += String.format("(?<%s>[0-9a-zA-Z\\-]+)\\/", jsonObject.getString("i"));

result.put(s + "?$", jsonObject.getString("index"));

break;

}

case 4: {// cdn

s += String.format("(?<%s>.\*)", jsonObject.getString("i"));

result.put(s + "$", "$");

break;

}

default:

return;

}

if (jsonObject.has("next")) {

JSONArray nextValue = jsonObject.optJSONArray("next");

for (Object object : nextValue) {

JSONObject nextObject = (JSONObject) object;

readRoadmapR(nextObject, result, new String(s));

}

}

}

public static JSONObject processMessage(String key, JSONObject message, App instance, String user) {

try {

Reflections reflections = new Reflections("by.vlobo.api.entity");

Set<Class<?>> annotatedClasses = reflections.getTypesAnnotatedWith(AApi.class);

System.out.println(key);

for (Class<?> clazz : annotatedClasses) {

if (IApiProcessor.class.isAssignableFrom(clazz)) {

IApiProcessor processor = (IApiProcessor) clazz.getDeclaredConstructor().newInstance();

;

return processor.process(message, instance, user);

}

}

} catch (InstantiationException | IllegalAccessException | NoSuchMethodException

| InvocationTargetException e) {

e.printStackTrace();

return IApiProcessor.CODE\_500\_INTERNAL\_SERVER\_ERROR;

}

return IApiProcessor.CODE\_500\_INTERNAL\_SERVER\_ERROR;

}

public static JSONObject addJsonObject(JSONObject jo1s, JSONObject jo2s) {

JSONObject jo = new JSONObject();

JSONObject jo1 = new JSONObject(jo1s.toString());

JSONObject jo2 = new JSONObject(jo2s.toString());

for (String key : jo1.keySet()) {

jo.put(key, jo1.get(key));

}

for (String key : jo2.keySet()) {

jo.put(key, jo2.get(key));

}

return jo;

}

public static void sendString(HttpExchange t, int code, String str) throws IOException {

byte[] bstr = str.getBytes("UTF-8");

t.sendResponseHeaders(code, bstr.length);

try (OutputStream os = t.getResponseBody()) {

os.write(bstr);

}

}

public static void sendFile(HttpExchange t, int code, File file) throws IOException {

t.sendResponseHeaders(code, file.length());

try (OutputStream os = t.getResponseBody()) {

Files.copy(file.toPath(), os);

}

}

public static String linkHtml(String firstHtml, HashMap<String, String> vars) throws IOException {

HashSet<String> cssFiles = new HashSet<String>();

HashSet<String> jsFiles = new HashSet<String>();

HashMap<String, Float> jsMain = new HashMap<String, Float>();

Document doc = processInclude(firstHtml, cssFiles, jsFiles, jsMain);

StringBuilder cssCommon = new StringBuilder();

StringBuilder jsCommon = new StringBuilder();

for (String fileName : cssFiles) {

try {

String cssContent = Files.readString(Path.of("www/" + fileName));

cssCommon.append(cssContent).append("\n");

} catch (IOException e) {

e.printStackTrace();

}

}

jsCommon.append(String.format("let global = %s;\n", new JSONObject(vars).toString()));

for (String fileName : jsFiles) {

try {

String jsContent = Files.readString(Path.of("www/" + fileName));

jsCommon.append(jsContent).append("\n");

} catch (IOException e) {

e.printStackTrace();

}

}

// document.addEventListener("DOMContentLoaded", function () {});

for (String fun : jsMain.entrySet()

.stream()

.sorted(Map.Entry.comparingByValue())

.map(Map.Entry::getKey)

.collect(Collectors.toList())) {

jsCommon.append(fun).append("\n");

}

doc.head().append("<style>\n" + cssCommon.toString() + "\n</style>");

doc.body().append("<script>\n" + jsCommon.toString() + "\n</script>");

// System.out.println(doc.toString());

return doc.toString();

}

private static Document processInclude(

String filePath,

HashSet<String> cssFiles,

HashSet<String> jsFiles,

HashMap<String, Float> jsMain)

throws IOException {

File inputFile = new File("www/" + filePath);

Document doc = Jsoup.parse(inputFile, "UTF-8", "");

Elements requiredEls = doc.select("required");

for (Element includeEl : requiredEls.select("[css]")) {

String srcValue = includeEl.attr("src");

cssFiles.add(srcValue);

includeEl.remove();

}

for (Element includeEl : requiredEls.select("[js]")) {

String srcValue = includeEl.attr("src");

jsFiles.add(srcValue);

includeEl.remove();

}

for (Element includeEl : requiredEls.select("[jsmain]")) {

String srcValue = includeEl.attr("src");

Float priorValue = Float.parseFloat(includeEl.attr("jsmain"));

jsMain.put(srcValue, priorValue);

includeEl.remove();

}

Elements includeEls = doc.select("include[src]");

for (Element includeEl : includeEls) {

String srcValue = includeEl.attr("src");

String includedContent = processInclude(srcValue, cssFiles, jsFiles, jsMain).body().html();

includeEl.after(includedContent);

includeEl.remove();

}

return doc;

}

}

package by.vlobo;

public class Main {

public static void main(String[] args) throws Exception {

Database.preset();

App app = new App();

app.start();

}

}

package by.vlobo;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Properties;

import org.json.JSONArray;

import org.json.JSONObject;

// Этот класс должен служить прослойкой между бэком, что бы я спокойно вносил правки в бд и тут, а само приложение не трогал.

public class Database {

private Connection connection;

public static void preset() {

try {

Class.forName("org.postgresql.Driver");

} catch (ClassNotFoundException e) {

System.err.println("Dont found class name");

e.printStackTrace();

}

}

public Database(String user, String password, String host, String dbName) {

try {

String url = "jdbc:postgresql://" + host + "/" + dbName;

Properties props = new Properties();

props.setProperty("user", user);

props.setProperty("password", password);

connection = DriverManager.getConnection(url, props);

} catch (SQLException e) {

System.out.println("Не удалось подключиться к БД");

// e.printStackTrace();

}

}

public Database(String user, String password) {

this(user, password, "localhost", "cukierka");

}

public JSONObject basecheck() {

try {

Statement st = connection.createStatement();

ResultSet rs = st.executeQuery("SELECT \n" + //

" table\_name as \"Table\",\n" + //

" column\_name as \"Column\"\n" + //

"FROM \n" + //

" information\_schema.columns\n" + //

"WHERE \n" + //

" table\_catalog = 'cukierka'\n" + //

" AND table\_schema NOT LIKE 'pg\_%'\n" + //

" AND table\_schema != 'information\_schema'\n" + //

"ORDER BY \n" + //

" table\_name, ordinal\_position;\n" + //

"");

JSONArray jsonArray = new JSONArray();

while (rs.next()) {

jsonArray.put(Tools.toJSONObject(rs));

}

return new JSONObject().put("answer", jsonArray);

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getUserByToken(String token) {

try {

String source = "SELECT user\_id FROM Tokens WHERE id = '%s';";

String sql = String.format(source, token);

ResultSet rs = connection.createStatement().executeQuery(sql);

if (rs.next()) {

return Tools.toJSONObject(rs);

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getUserInfoByUsername(String username) {

try {

PreparedStatement preparedStatement = connection.prepareStatement("SELECT \* FROM Users WHERE username = ?");

preparedStatement.setString(1, username);

ResultSet rs = preparedStatement.executeQuery();

if (rs.next()) {

return Tools.toJSONObject(rs);

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getUserInfo(String user) {

try {

String source = "SELECT \* FROM Users WHERE id = '%s';";

String sql = String.format(source, user);

ResultSet rs = connection.createStatement().executeQuery(sql);

if (rs.next()) {

return Tools.toJSONObject(rs);

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject addUser(String id, String name, String email, JSONObject other) {

try {

String source = "INSERT INTO Users (id, username, email, other) VALUES ('%s', '%s', '%s', '%s');";

String sql = String.format(source, id, name, email, other.toString());

if (connection.createStatement().executeUpdate(sql) == 1) {

return new JSONObject();

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject addProject(String id, String name, JSONObject other) {

try {

String source = "INSERT INTO Projects (id, name, applications, other) VALUES ('%s', '%s', ARRAY[]::VARCHAR[], '%s');";

String sql = String.format(source, id, name, other.toString());

if (connection.createStatement().executeUpdate(sql) == 1) {

return new JSONObject().put("id", id);

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject addBug(

String id,

String bugName,

String bugDescription,

String userId,

String projectId,

String criticality,

String priority,

String status,

JSONObject other) {

try {

String source = "INSERT INTO Bugs (id, name, description, user\_id, project\_id, criticality, priority, status, other) VALUES ('%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s');";

String sql = String.format(source, id, bugName, bugDescription, userId, projectId, criticality, priority, status, other.toString());

if (connection.createStatement().executeUpdate(sql) == 1) {

return new JSONObject();

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject addUsersToProjects(String user, String project) {

return addUsersToProjects(user, project, "default");

}

public JSONObject addUsersToProjects(String user, String project, String role) {

try {

String source = "INSERT INTO UsersToProjects (user\_id, project\_id, user\_role) VALUES ('%s', '%s', '%s');";

String sql = String.format(source, user, project, role);

if (connection.createStatement().executeUpdate(sql) == 1) {

return new JSONObject();

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getProjects(String user) {

try {

String source = "SELECT project\_id FROM UsersToProjects WHERE user\_id = '%s';";

String sql = String.format(source, user);

ResultSet rs = connection.createStatement().executeQuery(sql);

JSONArray jsonArray = new JSONArray();

while (rs.next()) {

jsonArray.put(Tools.toJSONObject(rs));

}

return new JSONObject().put("arr", jsonArray);

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getBugs(String project) {

try {

String source = "SELECT \* FROM Bugs WHERE project\_id = '%s';";

String sql = String.format(source, project);

ResultSet rs = connection.createStatement().executeQuery(sql);

JSONArray jsonArray = new JSONArray();

while (rs.next()) {

jsonArray.put(Tools.toJSONObject(rs));

}

return new JSONObject().put("arr", jsonArray);

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getBugInfo(String id) {

try {

String source = "SELECT \* FROM Bugs WHERE id = '%s';";

String sql = String.format(source, id);

System.out.println(sql);

ResultSet rs = connection.createStatement().executeQuery(sql);

if (rs.next()) {

return Tools.toJSONObject(rs);

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getMembers(String user) {

try {

String source = "SELECT user\_id FROM UsersToProjects WHERE project\_id = '%s';";

String sql = String.format(source, user);

ResultSet rs = connection.createStatement().executeQuery(sql);

JSONArray jsonArray = new JSONArray();

while (rs.next()) {

jsonArray.put(Tools.toJSONObject(rs));

}

return new JSONObject().put("arr", jsonArray);

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getProjectInfo(String project) {

try {

String source = "SELECT \* FROM Projects WHERE id = '%s';";

String sql = String.format(source, project);

ResultSet rs = connection.createStatement().executeQuery(sql);

if (rs.next()) {

return Tools.toJSONObject(rs);

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject getPermission(String user, String project) {

try {

String source = "SELECT \* FROM UsersToProjects WHERE user\_id = '%s' AND project\_id = '%s';";

String sql = String.format(source, user, project);

System.out.println(sql);

ResultSet rs = connection.createStatement().executeQuery(sql);

if (rs.next()) {

return Tools.toJSONObject(rs);

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject checkPassword(String user, String password) {

try {

String source = "SELECT \* FROM Users WHERE username = '%s' AND other->>'password' = '%s';";

String sql = String.format(source, user, password);

System.out.println(sql);

if (connection.createStatement().executeQuery(sql).next()) {

return new JSONObject();

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

public JSONObject addToken(String id, String user, String expires, JSONObject other) {

try {

String source = "INSERT INTO Tokens VALUES ('%s', '%s', '%s', '%s');";

String sql = String.format(source, id, user, expires, other.toString());

System.out.println(sql);

if (connection.createStatement().executeUpdate(sql) == 1) {

return new JSONObject().put("token", id);

} else {

return null;

}

} catch (SQLException e) {

e.printStackTrace();

return null;

}

}

}

package by.vlobo;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.util.HashMap;

import java.util.regex.Pattern;

import org.json.JSONException;

import org.json.JSONObject;

import org.json.JSONTokener;

public class App {

private Server server;

private Database database;

private HashMap<Pattern, String> roadmap;

public App() {

try {

JSONObject jsonObject = new JSONObject(new JSONTokener(new FileReader("./www/roadmap.json")));

roadmap = Tools.readRoadmap(jsonObject);

System.out.println("Roadmap собран");

} catch (JSONException e) {

e.printStackTrace();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

database = new Database("VloBo", " 1");

}

public void start() {

server = new Server(this);

}

public HashMap<Pattern, String> getRoadmap() {

return roadmap;

}

public Server getServer() {

return server;

}

public Database getDatabase() {

return database;

}

}

package by.vlobo.api;

import org.json.JSONObject;

import by.vlobo.App;

public interface IApiProcessor {

public static final JSONObject CODE\_200\_OK = new JSONObject().put("code", 200).put("status", "OK");

//public static final JSONObject CODE\_201\_CREATED = new JSONObject().put("code", 201).put("status", "Created");

//public static final JSONObject CODE\_204\_NO\_CONTENT = new JSONObject().put("code", 204).put("status", "No Content");

//public static final JSONObject CODE\_301\_MOVED\_PERMANENTLY = new JSONObject().put("code", 301).put("status", "Moved Permanently");

//public static final JSONObject CODE\_302\_FOUND = new JSONObject().put("code", 302).put("status", "Found");

//public static final JSONObject CODE\_304\_NOT\_MODIFIED = new JSONObject().put("code", 304).put("status", "Not Modified");

public static final JSONObject CODE\_400\_BAD\_REQUEST = new JSONObject().put("code", 400).put("status", "Bad Request");

public static final JSONObject CODE\_401\_UNAUTHORIZED = new JSONObject().put("code", 401).put("status", "Unauthorized");

public static final JSONObject CODE\_403\_FORBIDDEN = new JSONObject().put("code", 403).put("status", "Forbidden");

public static final JSONObject CODE\_404\_NOT\_FOUND = new JSONObject().put("code", 404).put("status", "Not Found");

//public static final JSONObject CODE\_405\_METHOD\_NOT\_ALLOWED = new JSONObject().put("code", 405).put("status", "Method Not Allowed");

public static final JSONObject CODE\_500\_INTERNAL\_SERVER\_ERROR = new JSONObject().put("code", 500).put("status", "Internal Server Error");

public static final JSONObject CODE\_501\_NOT\_IMPLEMENTED = new JSONObject().put("code", 501).put("status", "Not Implemented");

public static final JSONObject CODE\_503\_SERVICE\_UNAVAILABLE = new JSONObject().put("code", 503).put("status", "Service Unavailable");

JSONObject process(JSONObject message, App instance, String user);

}

# Приложение Б (обязательное) Результаты работы приложения

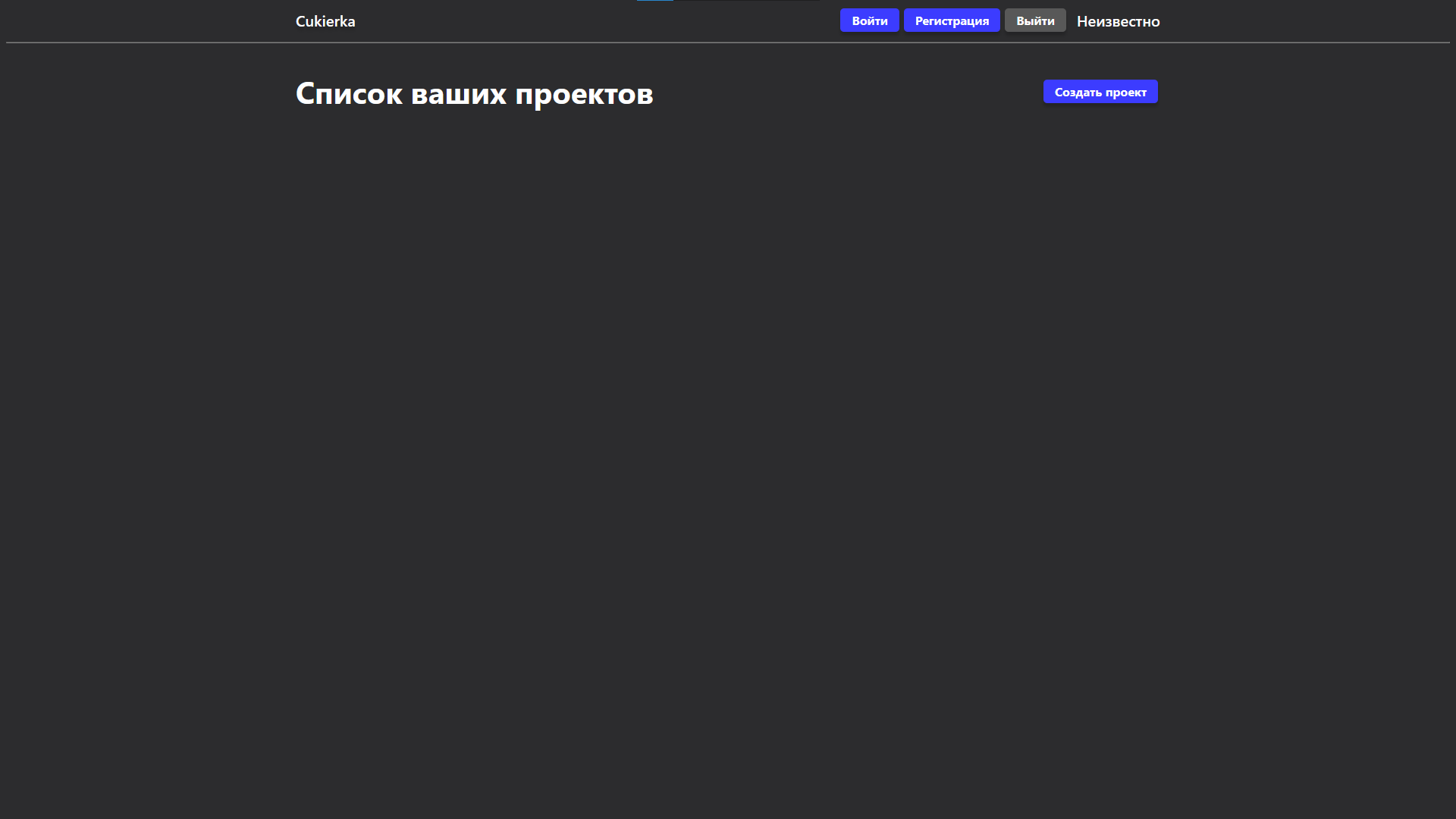


Рисунок Б.1 – Главное меню

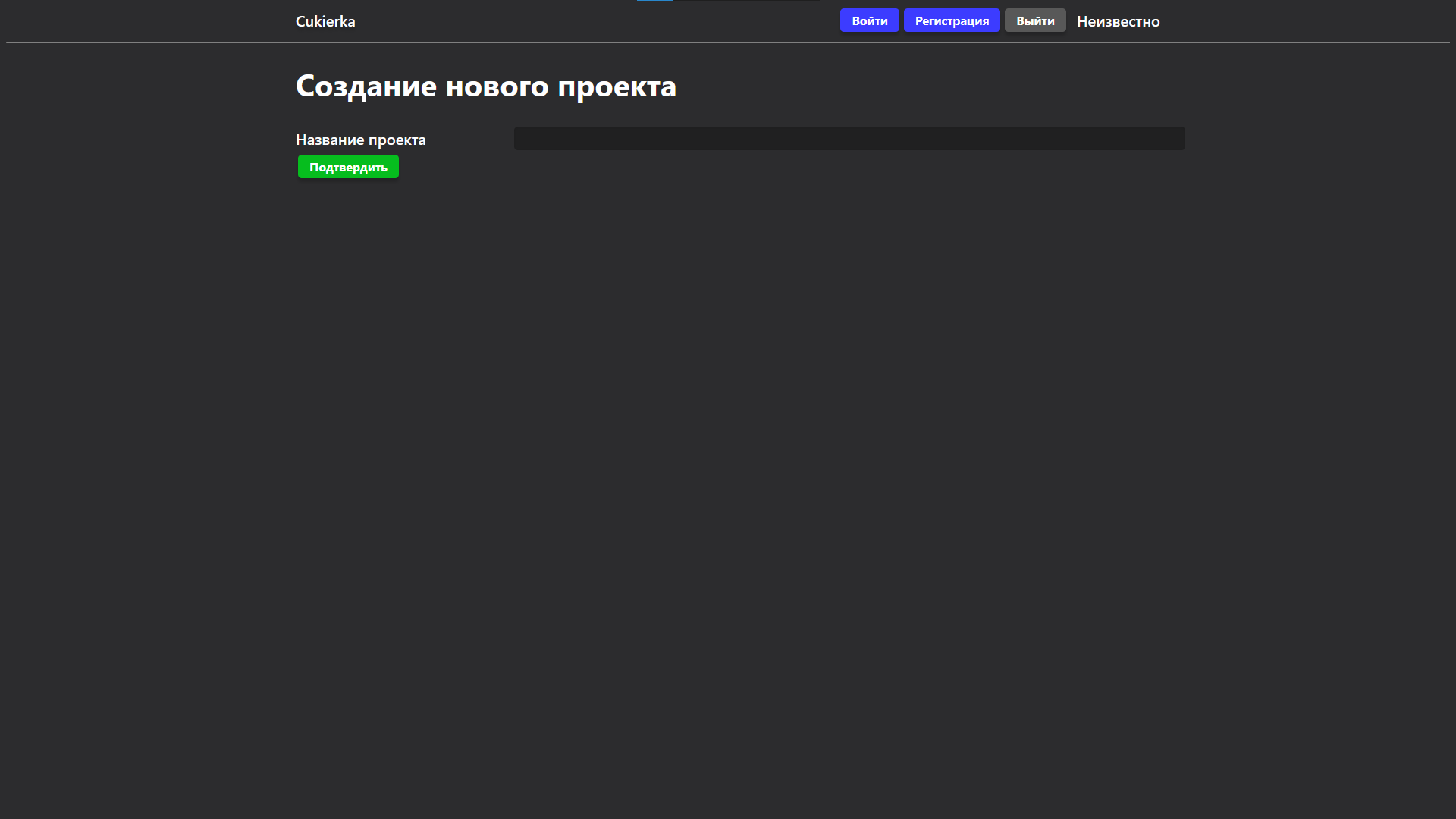


Рисунок Б.2 – Окно создание проекта

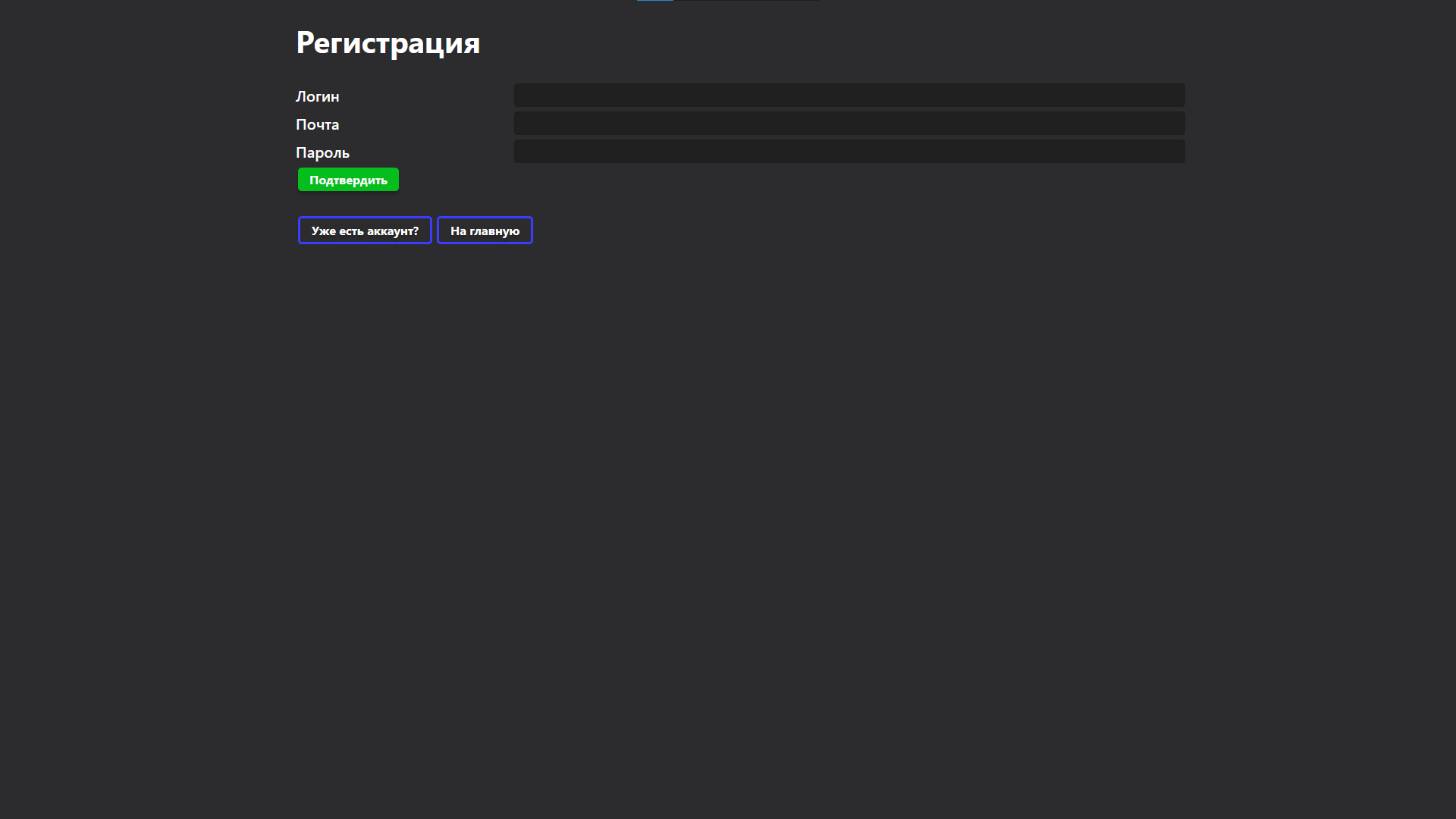


Рисунок Б.3 – Окно регистрации