

Планировщик задач

Авторы:

Группа ИВБО-01-20

1. Ермишин М. Е.
2. Чурсинов А. А.
3. Соколов А. Ю.
4. Букановская Н. Д.
5. Ильина Д. А.

Состав команды и роли

Команда состоит из 5 человек:

1. Ермишин М. Е.
2. Чурсинов А. А.
3. Соколов А. Ю.
4. Букановская Н. Д.
5. Ильина Д. А.



Ермишин Максим - Разработчик, потому что он обладает навыками программирования и разработки ПО. А так же менеджер проекта - управляет проектом, следит за его выполнением в соответствии с планом, руководит командой. Навыки: управление проектами, коммуникация, планирование, аналитическое мышление, решение проблем.



Ильина Дарья - технический писатель, создаёт документацию к продукту, включая инструкции, руководства и т.д. Навыки: понимание технологий, навыки написания, техническое понимание продукта.



Чурсинов Артемий - тестировщик, осуществляет тестирование продукта на разных уровнях и выявляет ошибки и проблемы. Навыки: тестирование, поиск и устранение ошибок, аналитическое мышление.



Состав команды и роли

Команда состоит из 5 человек:

1. Ермишин М. Е.
2. Чурсинов А. А.
3. Соколов А. Ю.
4. Букановская Н. Д.
5. Ильина Д. А.



Соколов Александр - дизайнер, создаёт дизайн интерфейса продукта. Навыки: работа с графическими редакторами, понимание требований пользователей, умение создавать привлекательный и удобный дизайн.



Букановская Наталья - аналитик, анализирует требования пользователей, определяет потребности рынка, обеспечивает правильное направление разработки. Навыки: аналитическое мышление, понимание рынка и потребностей пользователей.

Описание проекта

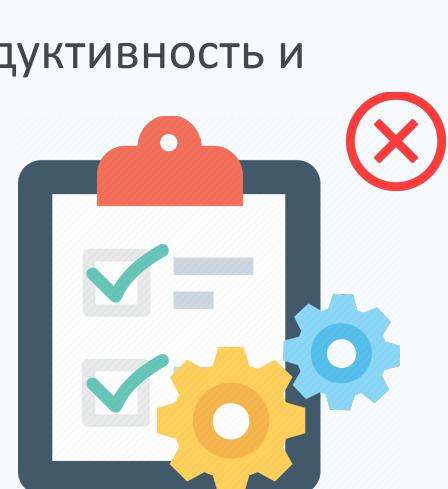
"Планировщик задач" - это удобное веб-приложение, которое позволяет эффективно организовать свой рабочий день.

С помощью данного сервиса можно создавать задачи, определять их приоритет и сроки выполнения, а также следить за их выполнением.

В планировщике можно установить уведомления о ближайших событиях и важных сроках, что позволяет не забывать о дедлайнах.

Планировщик задач имеет удобный интерфейс, легко настраивается и прост в использовании.

Он поможет пользователю стать более организованным, увеличить продуктивность и достичь своих целей.



Цели и требования



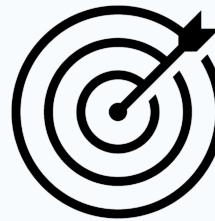
Цели проекта:

1. Создание приложения для планирования и управления задачами.
2. Обеспечение удобства и простоты использования приложения.
3. Максимальная надежность и стабильность работы приложения.

Требования к функциональности:

1. Создание и хранение задач.
2. Возможность установки сроков выполнения задач.
3. Возможность оплаты премиум версии приложения.
4. Оповещение пользователя о предстоящих сроках задач.
5. Возможность создания списка задач на определенный период времени.
6. Возможность редактирования и удаления задач.
7. Возможность установки напоминаний для задач.

Цели и требования



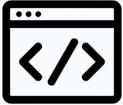
Требования к интерфейсу:

1. Интуитивно понятный и простой интерфейс.
2. Легкий доступ к основным функциям приложения.
3. Удобство работы с большим количеством задач.

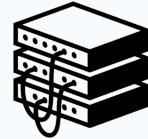
Изменения требований:

1. Изначально в планировщике задач не предусматривалась функция установки приоритетов задач, но после обсуждения с заказчиком было решено добавить эту функцию для более эффективного управления задачами.
2. Также были внесены изменения в функцию создания списка задач на определенный период времени. Изначально предлагалось делать это вручную, но после обратной связи от пользователей было добавлено автоматическое создание списка задач на ближайшую неделю.

Архитектура программного продукта



Архитектура приложения "Планировщик задач" включает в себя трехуровневую архитектуру: фронтенд, бэкенд и базу данных.



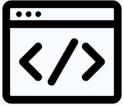
Выбор трехуровневой архитектуры обоснован несколькими факторами. Во-первых, такая архитектура обеспечивает легкое масштабирование и расширение проекта. Если в будущем потребуется добавить новые функции, то изменения в одном из звеньев не повлияют на работу других.



Во-вторых, трехуровневая архитектура помогает сохранить логику работы приложения в целостности. Код бэкенда может быть переиспользован в других приложениях, а база данных централизованно хранит все необходимые данные.



Архитектура программного продукта



В-третьих, трехуровневая архитектура облегчает оптимизацию производительности приложения. Например, можно использовать кеширование повторно используемых данных, оптимизировать обмен данными между звеньями, использовать распределенные системы обработки задач.

Если бы приложение было разработано на основании другой архитектуры, могли возникнуть затруднения. Например, большая часть бизнес-логики могла быть реализована в пользовательском интерфейсе, что усложнило бы ее переиспользование и масштабирование. Использование монолитной архитектуры без разделения приложения на звенья также могло бы сделать его менее гибким и увеличить трудность обеспечения производительности. Более сложные архитектуры, такие как сервис-ориентированная или микросервисная, могли бы усложнить разработку и потребовать больших расходов на инфраструктуру.



Наиболее значимые риски проекта

Один из серьезных рисков проекта "Планировщик задач" - это потеря данных пользователей. Если произойдет сбой системы или крах сервера, возможно, что все сохраненные задачи и другие пользовательские данные будут утрачены. Чтобы бороться с этим риском, разработчики должны регулярно резервировать данные и подготовить запасной сервер для быстрого восстановления.



Другим риском является неправильная работа системы из-за ошибки в программном обеспечении. Это может привести к некорректной обработке данных пользователей, таким как неправильное отображение задач или ошибки в расписании. Для уменьшения этого риска разработчики должны тестировать систему на ошибки и баги, прежде чем выпустить ее на общественную платформу.

Также риском является обновление или изменение операционной системы, которая может быть несовместима с приложением. Если это произойдет, пользователи могут потерять доступ к своим задачам или через приложение может стать недоступным. Чтобы избежать этого, разработчики должны регулярно проверять совместимость и обновлять программное обеспечение при необходимости.



Наиболее значимые риски проекта



Риском также является конкуренция с другими подобными программами. Если другие программы могут предложить лучший функционал и опыт пользователю, могут быть потеряны пользователи нашего приложения "Планировщик задач". Разработчики должны следить за конкурентами, анализировать их продукты, чтобы узнать, чего пользователи хотят, прежде чем предоставить более эффективное решение.



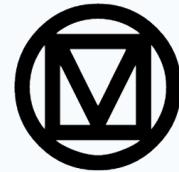
В случае наступления этих рисков разработчики должны незамедлительно принимать действия, предложенные в плане реагирования. Например, при потере или повреждении данных пользователей, подразумевается быстрое восстановление данных из резервной копии и сообщение об этом пользователям. Если возникают проблемы с программным обеспечением, оно должно быть незамедлительно исправлено или вернуто к предыдущей стабильной версии. Если в приложении не хватает функционала, нужно провести обновление или создать функцию в следующем релизе для сохранения пользователей.



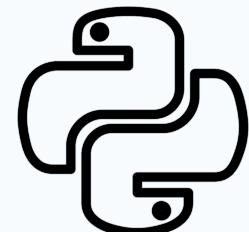
Описание стека технологий

Для реализации сервиса "Планировщик задач" были выбраны
нижеперечисленные программные решения.

1. Для уровня представления: KivyMD – популярный фреймворк для языка
программирования Python, реализующая набор виджетов в стиле Google
Material Design, для использования с фреймворком Kivy.



2. Для уровня приложения (сервера): Python - это высокоуровневый язык
программирования, который широко используется для создания приложений и
веб-сервисов.



Описание стека технологий



3. Фреймворки:

mysql.connector – это модуль для работы с базами данных MySQL на языке Python;



4. Для уровня данных: MySQL - является одной из наиболее популярных и широко используемых реляционных баз данных в мире. Она обеспечивает высокую надежность и производительность, что делает ее идеальным выбором для приложений, которые должны обрабатывать большое количество данных.



Диаграммы процессов проекта

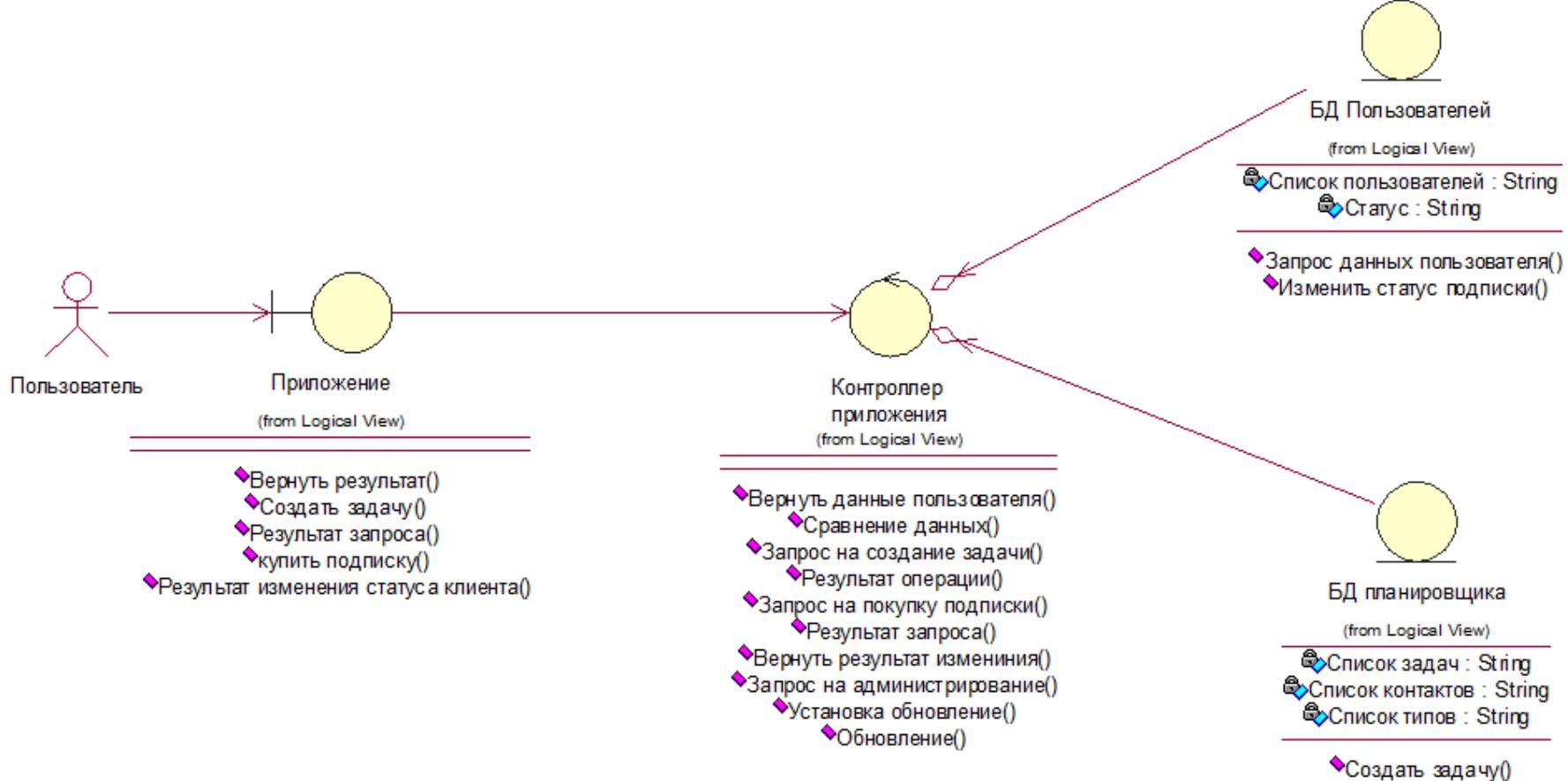
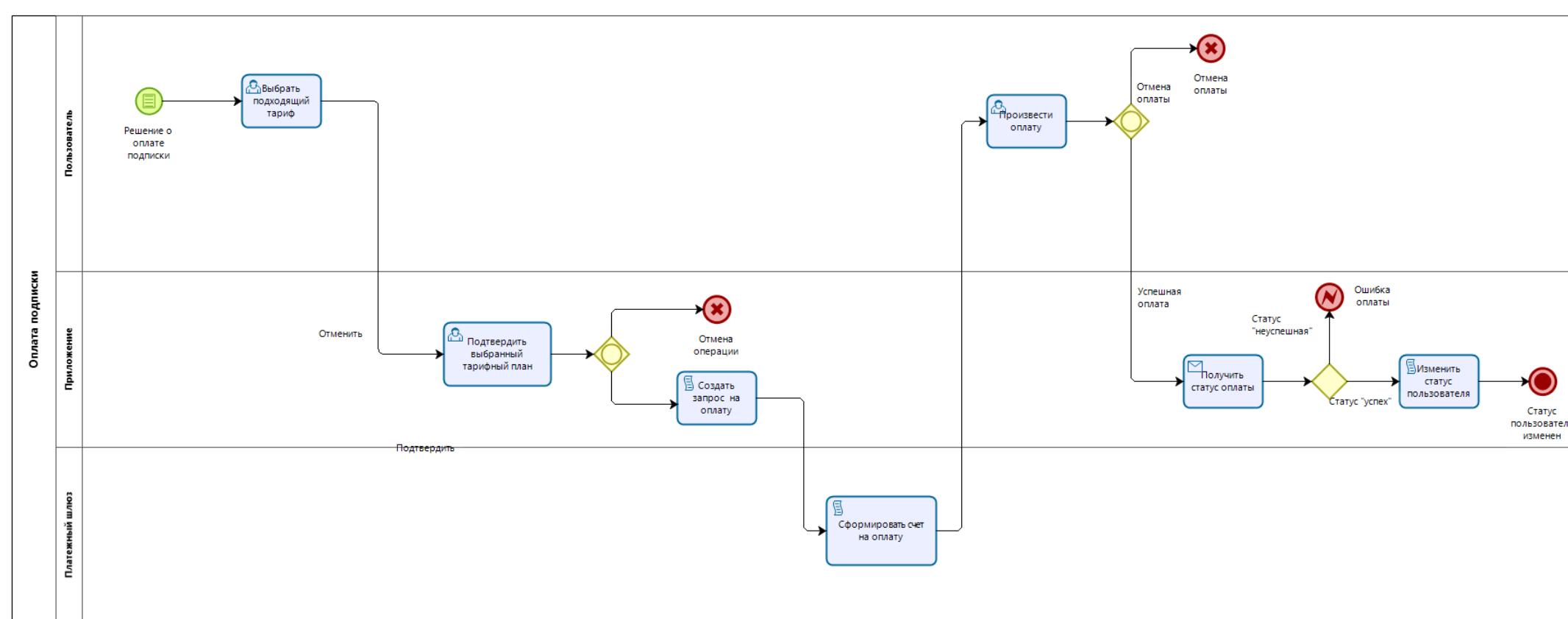


Диаграмма вариантов использования

Диаграммы процессов проекта



Диаграммы процессов проекта

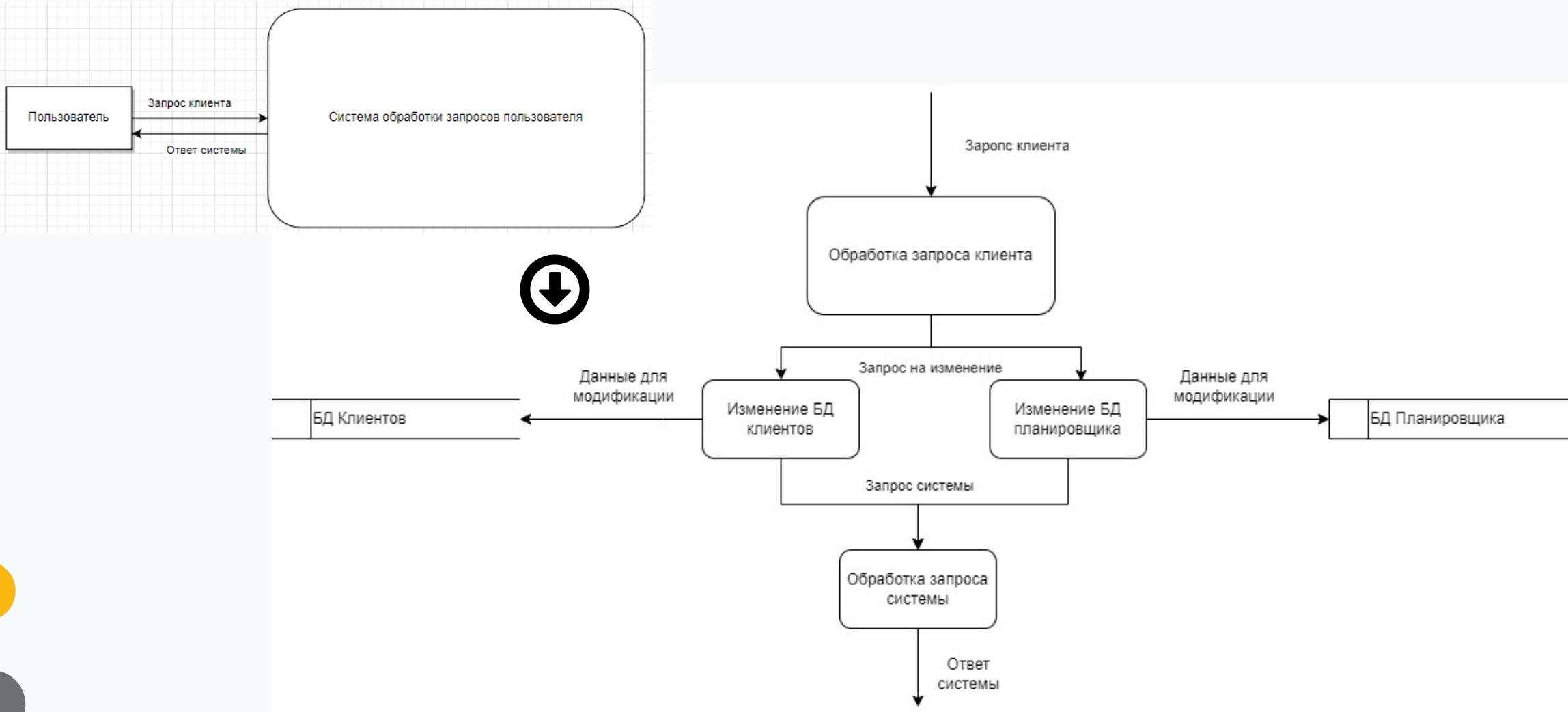


Диаграмма в нотации DFD

Описание функционала приложения

Приложение "Планировщик задач" позволяет конечному пользователю управлять временем и планировать свои задачи, чтобы увеличить продуктивность. Реализованный функционал включает:

1. Создание списка задач: пользователь может добавить новые задачи в список, указать для них названия, сроки выполнения и теги для удобства поиска.
2. Управление задачами: пользователь может отмечать выполненные задачи, удалять их или изменять сроки выполнения.
3. Покупка полной версии приложения: пользователь имеет возможность купить полную версию приложения.

Описание функционала приложения

5. Специальные тэги: приложение имеет готовые тэги для задач, как "срочно", "важно", "для дома" и т.п

6. Мультиплатформенность: приложение доступно на разных платформах, таких как iOS, Android и Web-версии.

7. Синхронизация: все задачи и изменения вносятся в базу данных приложения, которая автоматически синхронизируется на всех устройствах, где установлено данное приложение.

Приложение "Планировщик задач" является незаменимым помощником для организации своего рабочего времени и повышения продуктивности в жизни.

Тестирование

Процесс тестирования разработанного приложения "Планировщик задач" состоит из нескольких этапов:

1. Планирование тестирования:

- Анализ функциональных требований;
- Определение целей тестирования;
- Разработка тестовой документации;
- Выбор методов и инструментов тестирования.

2. Начальное тестирование:

- Проверка установки и конфигурации ПО;
- Тестирование основных функций приложения;
- Выявление критических ошибок.

3. Регрессионное тестирование:

- Проверка исправленных ошибок;
- Проверка работоспособности после внесения изменений.

4. Тестирование на соответствие требованиям:

- Проверка соответствия функциональных требований;
- Проверка соответствия требований по безопасности и производительности.

Тестирование

5. Тестирование на нагрузку:

- Оценка производительности приложения (скорость запуска, время отклика приложения и др.);
- Определение предельной нагрузки на приложение.

6. Тестирование на надежность:

- Проверка работоспособности приложения в условиях сбоев и непредвиденных ситуаций.

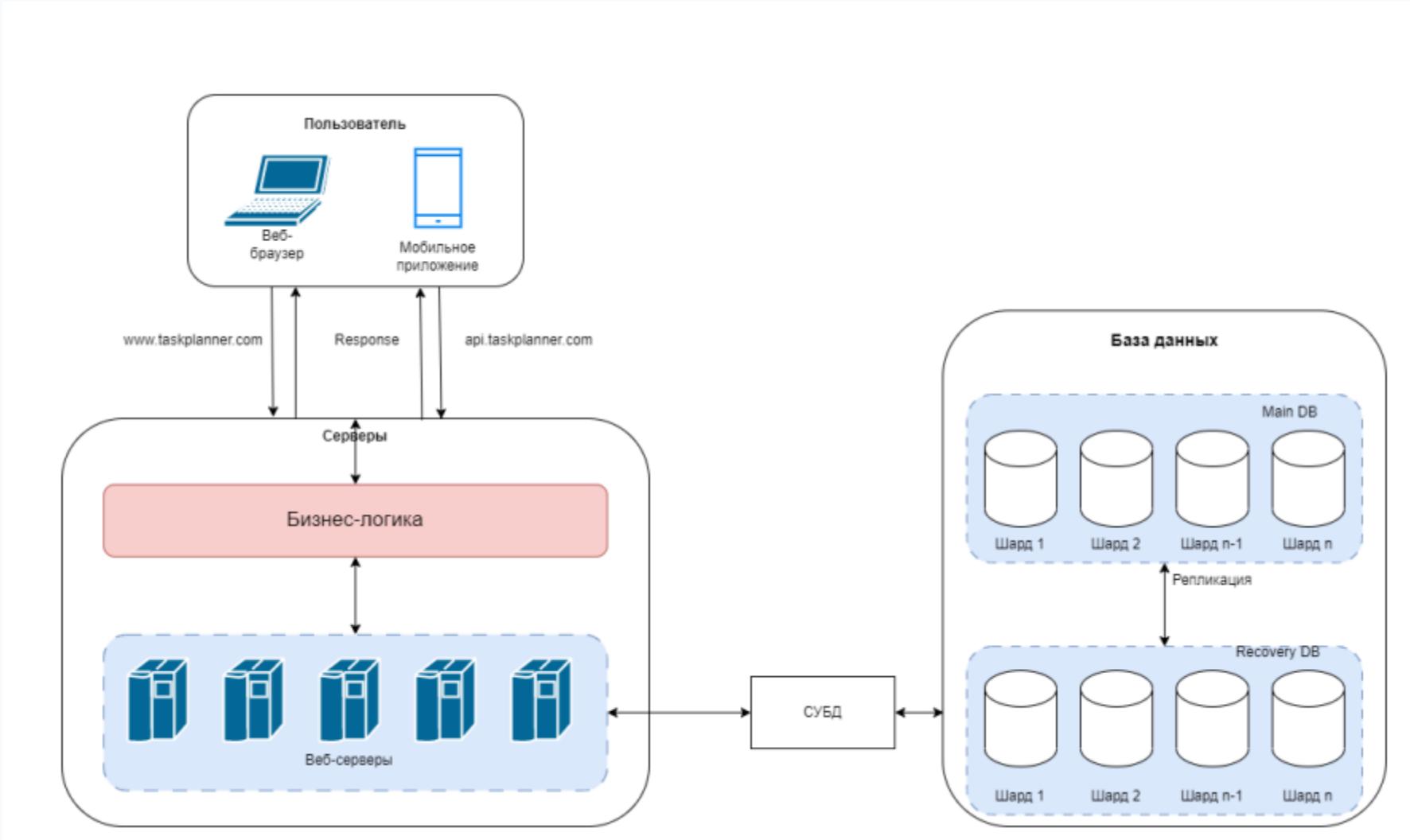
При тестировании приложения "Планировщик задач" применялись следующие методы тестирования:

- Метод белого ящика (проверка кода и алгоритмов);
- Метод функционального тестирования (проверка соответствия функциональных требований);
- Метод тестирования на нагрузку (оценка производительности приложения);
- Метод тестирования на надежность (проверка работоспособности приложения в условиях сбоев и непредвиденных ситуаций).

Так же для тестирования использовались методика BDD.

Развертывание

Архитектура, планируемая для развертывания



Развертывание



Для аренды серверов для архитектуры, планируемой для развертывания, было решено использовать Amazon AWS. Используем Amazon t2.medium. Аренда обойдётся в $8760 * 0.031 * 2 = 543$ долларов в год (Один инстанс для БД, второй для приложения).

Название	Виртуальные ЦПУ	Оперативная память (ГиБ)	Кредитов ЦПУ в час	Цена по требованию в час*	Инстанс, зарезервированный на 1 год, фактический почасовой тариф*	Инстанс, зарезервированный на 3 года, фактический почасовой тариф*
t2.nano	1	0,5	3	0,0058 USD	0,003 USD	0,002 USD
t2.micro	1	1,0	6	0,0116 USD	0,007 USD	0,005 USD
t2.small	1	2,0	12	0,023 USD	0,014 USD	0,009 USD
t2.medium	2	4,0	24	0,0464 USD	0,031 USD	0,021 USD
t2.large	2	8,0	36	0,0928 USD	0,055 USD	0,037 USD
t2.xlarge	4	16,0	54	0,1856 USD	0,110 USD	0,074 USD
t2.2xlarge	8	32,0	81	0,3712 USD	0,219 USD	0,148 USD

Развертывание

Для обработки входящих http запросов используем Nginx. Также используем его для Балансировки нагрузки в будущем

Также используем хранилище Amazon S3(0.023 доллара за 1 Гб).
Будем использовать его для хранения резервных копий и логов.



Для реализации CI конвейера необходимо создать дополнительные окружения: staging, QA и prod.

В качестве CI конвейера используем GitLab+GitLab runner. Это обеспечит быстрый выход обновлений и бесшовный переход веб-приложения на новую версию. Развернем его на нашем сервере.

В качестве средств мониторинга используем Node exporter+Prometheus+Grafana.



Документация разработчика

[Документация разработчика](#) была сгенерирована с помощью приложения для автогенерации документации Doxygen.

Task_Scheduler

Main Page Related Pages Namespaces Classes Files

GUI.MainApp Class Reference

Inheritance diagram for GUI.MainApp:

```
graph TD; MDApp --> GUIMainApp
```

Public Member Functions

```
def delete_table(self, screen)
def build(self)
def checked(self, instance_table, current_row)
def row_checked(self, instance_table, instance_row)
```

Member Function Documentation

• build()

```
def GUI.MainApp.build ( self )
```

• checked()

```
def GUI.MainApp.checked ( self,
                           instance_table,
                           current_row
                         )
```

• delete_table()

```
def GUI.MainApp.delete_table ( self,
                               screen
                             )
```

Generated by doxygen 1.9.6

Task_Scheduler

Main Page Related Pages Namespaces Classes Files

config Namespace Reference

Variables

```
str host = "127.0.0.1"
str user = "root"
str password = "1111"
str db_name = "task"
str MAIN_TABLE = "tasks"
str select_all = "SELECT * FROM "
str describe = "DESCRIBE "
str delete = "DELETE from "
```

Variable Documentation

• db_name

```
str config db_name = "task"
```

• delete

```
str config delete = "DELETE from "
```

• describe

```
str config describe = "DESCRIBE "
```

• host

Generated by doxygen 1.9.6

Документация пользователя

Документация пользователя была создана на [GitHub Wiki](#)

The screenshot shows the GitHub Wiki page for the 'Task_Scheduler' repository. The main content area features a heading 'Welcome to the Task_Scheduler wiki!' followed by a paragraph of text and a bulleted list. Below this is a section titled 'Установка приложения' (Application Installation) with a note about cloning the repository. A sidebar on the right lists 'Pages' including 'Home' and 'Руководство по использованию приложения'. At the bottom, there's a footer with links to GitHub's Terms, Privacy, and Status pages.

This screenshot displays the 'Руководство по использованию приложения' (User Guide) page from the same GitHub Wiki. It includes sections for 'Начальный экран' (Initial Screen) and 'Окно добавления новых задач' (New Task Addition Window). A large image shows the application's interface with a 'Новая задача' (New Task) button highlighted. The sidebar on the right contains links to other pages like 'Home' and 'Руководство по использованию приложения'.

Достоинства приложения/системы

1. Удобство использования: Планировщик задач обладает простым и понятным интерфейсом, легкоусвоемом пользователями с любым опытом работы с подобными приложениями. В отличие от некоторых аналогов, он не перегружен функциями и не требует дополнительных настроек и научения.
2. Функциональность: Планировщик задач позволяет создавать задачи, устанавливать приоритеты и даты выполнения, а также детализировать информацию о задаче с помощью комментариев. Он также обладает возможностью создания регулярных задач и уведомлений их о выполнении. Однако, по сравнению с некоторыми аналогами, у приложения может быть меньше функций.
3. Совместимость: Планировщик задач предназначен для работы на различных операционных системах, что обеспечивает его простоту и универсальность. Он также совместим с многими другими приложениями и сервисами, что позволяет увеличить его использование и удобство работы для пользователя.

Достоинства приложения/системы

4. Простота использования и установки: Планировщик задач имеет свою собственную страницу на GitHub, где размещен исходный код, установщик, а также руководство пользователя.
5. Открытость и бесплатность: Планировщик задач - программное обеспечение с открытым исходным кодом, что позволяет использовать и модифицировать приложение абсолютно свободно.

Недостатки приложения/системы

1. Распознание речи: приложение не обладает функцией признания речи и управления голосом, что может быть не так удобно на устройствах, где это предусмотрено.
2. Ранняя версия: приложение не обладает значительной функциональностью и находится в предрелизной версии. Нужный функционал будет добавлен позже.
3. Не обеспечивает достаточную гибкость для решения необходимых задач и изменений в процессе выполнения.
4. Отсутствие интеграции с другими приложениями и системами.
5. Не всегда обеспечивает удобное представление информации и ее визуализацию.