СОДЕРЖАНИЕ

1. АКТУАЛЬНОСТЬ И ВОСТРЕБОВАННОСТЬ РАЗРАБАТЫВАЕМОГО ПРОДУКТА

В современном мире множество малых и средних компаний, которые занимаются разработкой различного рода программного обеспечения. В то время, как крупные компании давно и успешно используют для контроля и ведения разработки менеджеры задач, которые уже давно находятся на рынке и хорошо зарекомендовали себя, малые и средние компании и предприятия практически никогда не используют подобные технологии в своих проектах за счет их высокой стоимости и функционала, который для таких компаний может быть избыточными да и в целом менеджер задач может быть очень нагруженным с функциональной точки зрения, что требует временных затрат на обучение сотрудников.

Для подобных организаций имеет смысл иметь менеджер задач, который будет иметь базовый функционал, необходимый для ведения разработки и интуитивно-понятный интерфейс для взаимодействия с пользователем.

2. ЦЕЛЬ И ЗАДАЧИ ПРОЕКТА И ИСПОЛНИТЕЛИ

В рамках решения данной проблемы была поставлена следующая цель: разработка программного продукта для помощи в решении различных рабочих задач организации.

Для достижения поставленной цели было необходимо решить следующие задачи:

1. Разработать простой и интуитивно понятный пользовательский интерфейс;
2. Создать современный дизайн и использовать современные технологии в разработке;
3. Создать продукт, который будет работать на ПК типовой комплектации со свободным выходом в Интернет;
4. Реализовать запуск приложения без привязки к браузеру;
5. Реализовать базовый функционал, которого будет достаточно для помощи в решении рабочих задач.

Исполнителями данного проекта являются: Половинкин А. Е. и Лаптев А. В.

3. ОБЩИЕ СВЕДЕНИЯ О ПРОДЕЛАННОЙ РАБОТЕ

Ход выполнения работы был разделен на следующие этапы:

1. Определение формата реализации (Web-приложение, Desktop-приложение, др.);
2. Определение стека технологий, которые будут задействованы в разработке;
3. Разработка backend;
4. Разработка frontend;
5. Проверка интеграции frontend и backend;
6. Разработка дизайна Web-приложения;
7. Тестирование готового продукта.

На первом этапе необходимо было определиться с форматом реализации менеджера задач. В итоге был выбран формат Web-приложения, поскольку такой формат не требует места на накопителе, а также позволяет сразу же синхронизировать любые изменения в списке задач для пользователей.

Далее было необходимо определиться со стеком технологий, которые будут использованы для разработки. Готовое решение должно быть современным и отвечать текущим тенденциям в ...

По этим причинам были выбраны следующие технологии для использования во время разработки:

1. В качестве фреймворка для RESTfull API разработки был выбран FastAPI. FastAPI —- это современный, быстрый (высокопроизводительный) веб-фреймворк для создания API используя Python 3.6+, в основе которого лежит стандартная аннотация типов Python.

   FastAPI активно использует декораторы, аннотации типов и интроспекцию кода, что позволяет уменьшить количество шаблонного кода в веб-приложении.

Ключевые особенности FastAPI, которые повлияли на выбор данного фреймворка:

1.1. Скорость: Очень высокая производительность, на уровне NodeJS и Go (благодаря Starlette и Pydantic). Один из самых быстрых фреймворков Python;

1.2. Быстрота разработки: Позволяет увеличить скорость разработки примерно на 200–300%;

1.3. Меньше ошибок: Сокращено примерно на 40% количество ошибок, вызванных человеком (разработчиком). Этот и предыдущий пункт подтверждены лишь тестами внутренней команды разработчиков;

1.4. Интуитивно понятный: Отличная поддержка редактора. Автозавершение везде. Меньше времени на отладку;

1.5. Лёгкость: Разработан так, чтобы его было легко использовать и осваивать. Меньше времени на чтение документации;

1.6. Краткость: Сведено к минимуму дублирование кода. Каждый объявленный параметр – определяет несколько функций;

1.7. Надежность: На выходе получается готовый к работе код. С автоматической интерактивной документацией;

1.8. На основе стандартов: Основан на открытых стандартах API и полностью совместим с ними: OpenAPI (ранее известном как Swagger) и JSON Schema.

2. В качестве СУБД был выбран PostgreSQL. PostgreSQL – свободная объектно-реляционная система управления базами данных (СУБД).

Сильными сторонами PostgreSQL считаются:

2.1. Высокопроизводительные и надёжные механизмы транзакций и репликации;

2.2. Расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl;
2.3. Дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки модулей расширения на языке C;
2.4. Наследование;
2.5. Возможность индексирования геометрических (в частности, географических) объектов и наличие базирующегося на ней расширения PostGIS;
2.6. Встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;
2.7. Расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных).

PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса.

PostgreSQL поддерживает одновременную модификацию БД несколькими пользователями с помощью механизма Multiversion Concurrency Control (MVCC). Благодаря этому соблюдаются требования ACID и практически отпадает нужда в блокировках чтения.

3. Для создания пользовательских интерфейсов был выбран JavaScript-фреймворк с открытым исходным кодом – Vue.js. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

   Основные достоинства Vue.js:

   3.1. Можно использовать только знания JavaScript и HTML;

3.2. Возможно применение Typescript;

3.3. В Vue.js реализуется шаблон MVVM;

3.4. Vue.js предлагает возможность привязки данных на Javascript, так что вывод и ввод данных сопрягаются непосредственно с источником данных;

Vuetify – это библиотека пользовательского интерфейса, не требующая навыков дизайна, с красиво созданными вручную компонентами Vue.

4. Проверка данных и управление настройками с использованием аннотаций типа Python – для этих целей использовался Pydantic.

Pydantic использует некоторые интересные новые языковые функции, а также:

4.1. Отлично работает с IDE/линтером/мозгом: Нет нового микроязыка определения схемы для изучения. Если программисту известно, как использовать подсказки типов Python, ему известно, как использовать pydantic . Структуры данных —- это просто экземпляры классов, которые определяются с помощью аннотаций типов, поэтому автодополнение, linting, mypy , IDE (особенно PyCharm) должны правильно работать с проверенными данными;

4.2. Двойное использование: Класс BaseSettings pydantic позволяет использовать pydantic как в контексте «проверить данные этого запроса», так и в контексте «загрузить настройки моей системы». Основные отличия заключаются в том, что системные настройки можно считывать из переменных среды, и часто требуются более сложные объекты, такие как DSN и объекты Python;

4.3. Быстрый: Pydantic всегда серьезно относился к производительности, большая часть библиотеки скомпилирована с помощью cython, что дает ускорение 50%, обычно это так же быстро или даже быстрее, чем у большинства подобных библиотек;

4.4. Проверка сложных структур: Использование рекурсивных пидантических моделей , typingстандартных типов (например List, Tuple, Dictи т. д.) и валидаторов позволяет четко и легко определять, проверять и анализировать сложные схемы данных;

4.5. Расширяемый: Pydantic позволяет определять пользовательские типы данныхvalidator , или вы можете расширить проверку с помощью методов на модели, украшенной декоратором;

4.6. Интеграция классов данных: Кроме того BaseModel, pydantic предоставляет dataclassдекоратор, который создает классы данных Python с анализом и проверкой входных данных.

5. Для синхронизации объектов Python и записей реляционной базы данных была выбрана библиотека SQLAlchemy – это программная библиотека на языке Python для работы с реляционными СУБД с применением технологии ORM. SQLAlchemy позволяет описывать структуры баз данных и способы взаимодействия с ними на языке Python без использования SQL.

   SQLAlchemy – это набор инструментов Python SQL и Object Relational Mapper, который дает разработчикам приложений всю мощь и гибкость SQL.

   Работает back-end для баз данных: MySQL, PostgreSQL, SQLite, Oracle и других, между которыми можно переключаться изменением конфигурации.

   Основные возможности:

   5.1. Использование ORM не является обязательным;

   5.2. Устоявшаяся архитектура;

   5.3. Возможность использовать SQL, написанный вручную;

   5.4. Поддержка транзакций;

   5.5. Создание запросов с использованием функций и выражений Python;

5.6. Модульность и расширяемость;
5.7. Дополнительная возможность раздельного определения объектного отображения и классов;
5.8. Поддержка составных индексов;
5.9. Поддержка отношений между классами, в том числе «один-ко-многим» и «многие-ко-многим»;
5.10. Поддержка ссылающихся на себя объектов;
5.11. Предварительная и последующая обработка данных (параметров запроса, результата) и другие.

Использование SQLAlchemy для автоматической генерации SQL-кода имеет несколько преимуществ по сравнению с ручным написанием SQL:

5.1. Безопасность. Параметры запросов экранируются, что делает атаки типа внедрение SQL-кода маловероятными;
5.2. Производительность. Повышается вероятность повторного использования запроса к серверу базы данных, что может позволить ему в некоторых случаях применить повторно план выполнения запроса;
5.3. Переносимость. SQLAlchemy, при должном подходе, позволяет писать код на Python, совместимый с несколькими back-end СУБД. Несмотря на стандартизацию языка SQL, между базами данных имеются различия в его реализации, абстрагироваться от которых и помогает SQLAlchemy.

После этого началась разработка backend части приложения. Backend решения представлен несколькими взаимодействующими контейнерами Docker. В одном контейнере развёрнута БД, где хранятся сами задачи и информация связанная с ними. В другом развернута API для работы с БД.

После того, как были реализованы frontend и backend. API отвечающая за взаимодействие web интерфейса и базы данных представляет из себя на-

```yaml
version: "4"
services:
  web:
    build: .
    command: bash -c 'while !</dev/tcp/db/5432; do sleep 1; done; uvicorn app.main:app --host 0.0.0.0'
    ports:
      - 8008:8000
    environment:
      - DATABASE_URL=postgresql://mgknn:tst@db:5432/postgres
    depends_on:
      - db
    volumes:
      - .:/app
  db:
    image: postgres:13.3
    environment:
      POSTGRES_DB: "postgres"
      POSTGRES_USER: "mgknn"
      POSTGRES_PASSWORD: "tst"
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data/
volumes:
  postgres_data:
```

Рис. 3.1 Файл Docker для запуска контейнеров.

бор асинхронных методов. Каждый из методов выполняет свою функцию: от добавления новой задачи, до удалению уже существующей. При этом данные проходят проверку на соответствие типов, что исключает возможность занесения некорректных данных в таблицы базы данных.

Frontend проекта разворачивается с помощью npm. Кнопки, таблицы и другие элементы разработанного веб интерфейса путем вызов методов API позволяют вести работу с задачами: добавлять, удалять, редактировать и т.д. Помимо функционала для frontend составляющей проекта был выработан и стиль, который применялся для всех элементов интерфейса. Была подобрана цветовая палитра, определено положение всех кнопок и областей, выбраны иконки.

После завершения основной разработки было произведено ручное тестирование функционала всего приложения:

1. Создание новой задачи;
2. Редактирование существующей задачи;
3. Удаление задачи;

```python
app = FastAPI()


@app.on_event("startup")
async def startup():
    models.Base.metadata.drop_all(bind=engine)  # await
    models.Base.metadata.create_all(bind=engine)


@app.get("/tasks", response_model=List[TaskSchemaExpanded], status_code=200)
async def read_tasks(db: Session = Depends(create_session)):
    tasks = db.query(Task).all()
    return tasks

@app.post('/tasks/add', response_model=TaskSchemaExpanded, status_code=201)
async def create_task(task: TaskSchema, db: Session = Depends(create_session)):
    new_task = Task(**task.dict())
```

Рис. 3.2 API проекта.

4. Изменение статуса задачи.

Весь функционал отрабатывал стабильно и без ложных срабатываний, никаких багов в ходе тестирования найдено не было.

4. РЕЗУЛЬТАТЫ ПРОЕКТА

Конечный программный продукт представляет собой web-приложение — менеджер задач, обладающий необходимым базовым функционалом для помощи в ведении разработки. В частности, он обеспечивает следующие возможности:

В качестве СУБД был выбран PostgreSQL. Работа с БД осуществляется из Python с помощью SQLAclhemy. С помощью этого инструмента при первичном запуске beckend’а происходит создание таблицы для наших будущих задач. Этим же инструментом, когда мы получаем соответствующие запросы, идет редактирование, удаление и создание задач.

У данного продукта есть свои сильные и слабые стороны.

К достоинствам можно отнести следующее:

1. Использование быстрых и современных фреймворков и расширений к ним.
2. Автоматизация ряда задач, которые включают в себя работу с БД.
3. Простоту в использовании, за счет удобного интерфейса и навигации в нем.
4. Малую потребность в ресурсах, поскольку данное решение не является перегруженным функционалом.
5. Возможность расширения функционала.

Программную реализацию этого механизма можно увидеть в приложении.

1. Отслеживание текущих задач на главном экране с их валлидацией по времени — более новые задачи добавляются в конец, толкая более старые вверх перечня, что обеспечивает меньшую путаницу и помогает держать в поле зрения задачи, которые требуют больше времени для решения;
2. Создание новой задачи;

3. Удаление (закрытие) задачи;
4. Редактирование задачи.

Менеджер задач работает на всех устройствах, включая смартфоны, и не имеет никаких проблем с масштабированием интерфейса под разные соотношения сторон и диагонали. Функционал также не зависит от выбранного устройства.

Ниже будет представлено более подробное описание каждого из них.

Главный экран

Главный экран представляет собой минималистично оформленную страницу, на которой в заголовке обозначены две кнопки, с помощью которых можно обновить количество задач (динамического обновления пока что не присутствует) и создать новую задачу соответственно.

Ниже представлен перечень задач, в сортировке от более старой к более новой.

Для каждого статуса задачи предусмотрена разная сила подсветки. Предусмотрены следующие статусы:

1. Задача создана;
2. Задача в работе;
3. Задача завершена.

На рис. 1. показано то, как выглядит главный экран приложения при нескольких созданных активных и завершенных задачах.

Удаление (закрытие) задачи

Напротив каждой задачи с правой стороны есть две кнопки, которые отвечают за редактирование задачи и удаление. Чтобы удалить задачу достаточно нажать на красную урну. Никаких подтверждений при этом не требуется, удаление происходит в одно действие.

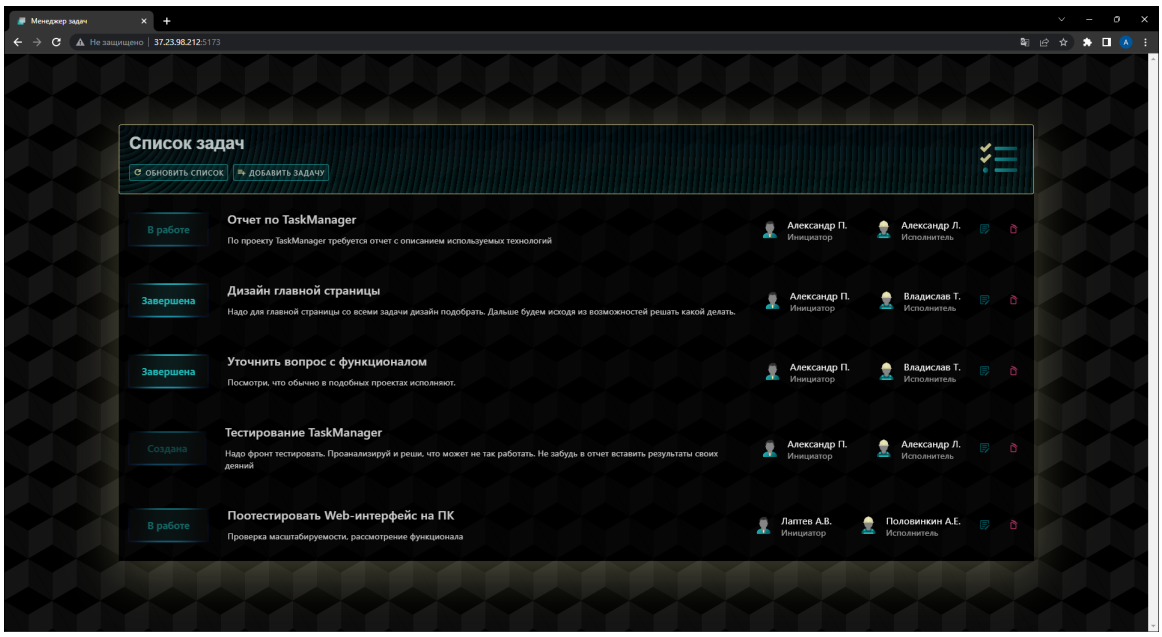

Рис. 4.1 Главное окно приложения

Редактирование задачи

Для редактирования задачи нужно нажать соответствующую кнопку со списком (синего цвета). После этого откроется окно с задачей, в которую можно вносить изменения по всем доступным полям. Поля, доступные для редактирования:

1. Название задачи
2. Описание задачи
3. Инициатор задачи
4. Исполнитель задачи
5. Статус задачи

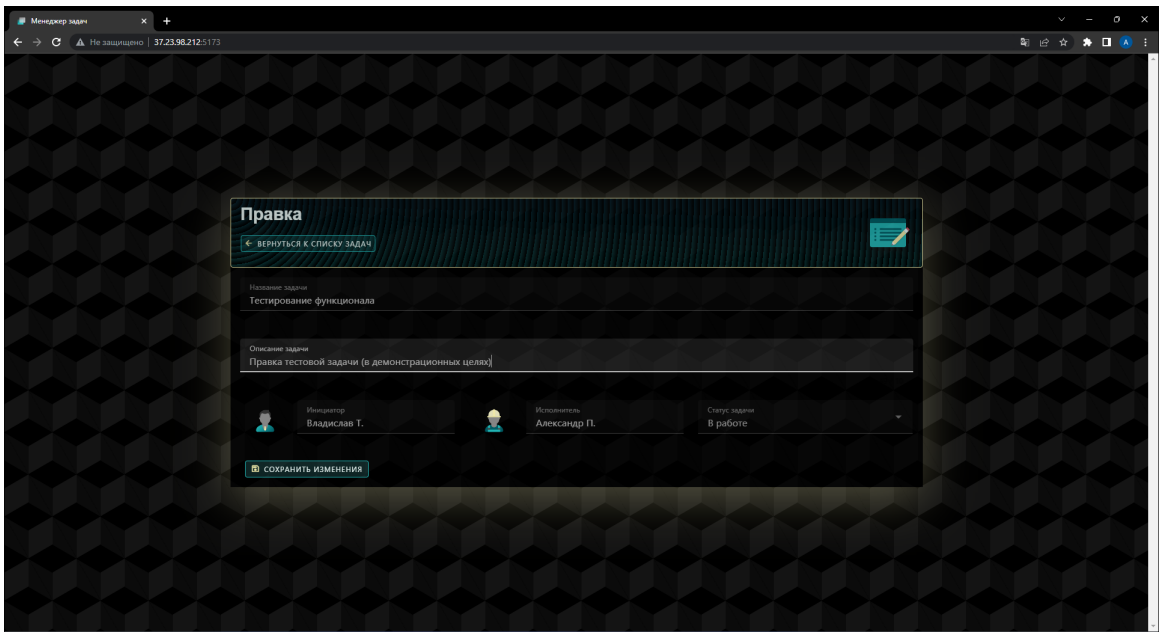

Рис. 4.2 Окно для редактирования выбранной задачи

Также в окне присутствуют две кнопки: для сохранения изменений и для возврата к главной странице.

Создание новой задачи

Чтобы создать новую задачу нужно, как уже упоминалось, нужно нажать соответствующую кнопку в меню на главном экране. После этого появится окно, которое похоже на окно редактирования задачи. Вид этого она показан на рис. 3.

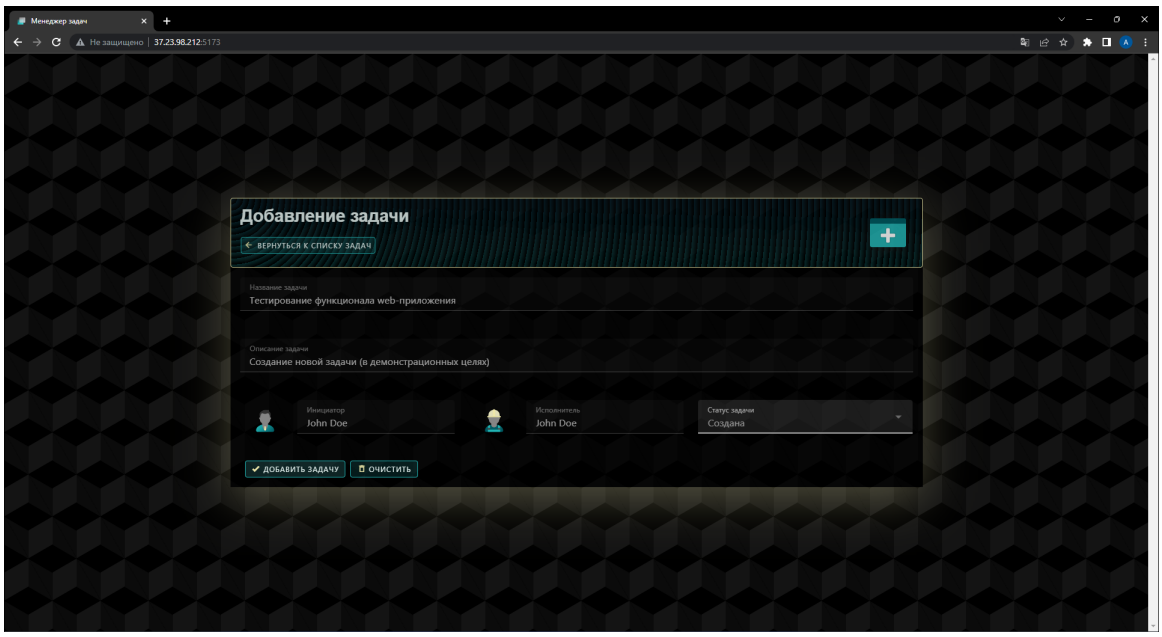

Рис. 4.3 Окно для создания новой задачи

Здесь также нужно заполнить те же самые поля, единственным отличием является наличие кнопки «Очистить», которая очищает все введенные данные и вместо кнопки сохранения изменений появляется кнопка «Добавить задачу».

ПРИЛОЖЕНИЕ 2

Листинг 4.1 Схема БД

```python
from sqlalchemy.schema import Column
from sqlalchemy.types import Text, JSON
from sqlalchemy.dialects.postgresql import UUID
import uuid

from app.database import Base

class Task(Base):
    __tablename__ = "Tasks"
    task_uuid = Column(UUID(as_uuid=True),
     ↪  primary_key=True, default=uuid.uuid4)
    task_description = Column(Text())
    task_name = Column(Text())
    task_params = Column(JSON)
```

Листинг 4.2 Vue

```
<template>
  <v-container v-if="!AddTask && !EditTask"
   ↪  style="padding: 0px;">
  <v-card
      color="rgb(33 178 178)"
      theme="dark"
      variant="outlined"
  >
```

```
8       <div class="d-flex flex-no-wrap
          ↪  justify-space-between">
9         <div>
10          <v-card-title class="text-h5" >
11            Список задач
12          </v-card-title>
13
14  <!--          <v-card-subtitle>Выберете задачу и вы
    ↪  сможете просмотреть, отредактировать или удалить
    ↪  её</v-card-subtitle>-->
15
16          <v-card-actions
             ↪  v-if="!this.$vuetify.display.mobile">
17            <v-btn
18                class="ms-2"
19                variant="outlined"
20                size="small"
21                v-on:click="update_task_list()"
22                id = "v-btn_on_header"
23            ><v-icon  id="task_on_header"
               ↪  >mdi-refresh</v-icon>Обновить
               ↪  список</v-btn>    <!--
               ↪  v-on:click="update_task_list()"--> <!--
               ↪  prepend-icon="mdi-refresh"-->
24            <v-btn
25                class="ms-2"
26                variant="outlined"
27                size="small"
```

```
28                v-on:click="add_new_task()"
29                id = "v-btn_on_header"
30              ><v-icon id="task_on_header"
                ↪  >mdi-playlist-plus</v-icon>Добавить
                ↪  задачу</v-btn>
31            </v-card-actions>
32
33
34            <v-card-actions
                ↪  v-if="this.$vuetify.display.mobile">
35              <v-btn
36                  class="ms-2"
37                  variant="outlined"
38                  size="large"
39                  v-on:click="update_task_list()"
40                  id = "v-btn_on_header"
41              ><v-icon  id="task_on_header"
                ↪  >mdi-refresh</v-icon></v-btn>   <!--
                ↪  v-on:click="update_task_list()"--> <!--
                ↪  prepend-icon="mdi-refresh"-->
42              <v-btn
43                  class="ms-2"
44                  variant="outlined"
45                  size="large"
46                  v-on:click="add_new_task()"
47                  id = "v-btn_on_header"
48              ><v-icon id="task_on_header"
                ↪  >mdi-playlist-plus</v-icon></v-btn>
```

```
<!--            <v-btn-->
<!--                class="ms-2"-->
<!--                variant="outlined"-->
<!--                size="large"-->
<!--                id = "v-btn_on_header"-->
<!--            ><v-icon id="task_on_header"
    ↪  >mdi-translate-variant</v-icon></v-btn>-->
          </v-card-actions>
        </div>

        <v-avatar
            class="ma-3"
            size="6em"
            rounded="0"
        >
          <v-img id="todo_svg" src="/todo.svg"></v-img>
        </v-avatar>
      </div>
    </v-card>



<!--       <v-list lines="three">-->
<!--       <v-list-item-->
<!--           v-for="task in TasksArray"-->
<!--
    ↪  v-on:click="select_task(task['task_uuid'])"-->
<!--           :key="task['task_name']"-->
```

```
<!--           :title="task['task_name']"-->
<!--           :subtitle="task['task_description']">
  ↪ &lt;!&ndash;  append-icon="mdi-delete-empty-
  ↪ outline"&ndash;&gt;-->
<!--        <template v-slot:prepend="{ isActive }">-->

<!--          <v-list-item__content>-->
<!--         <v-list-item-
  ↪ title>{{task['task_name']}}</v-list-item-title>-->
<!--          <v-list-item-
  ↪ subtitle>{{task['task_description']}}</v-list-
  ↪ item-subtitle>-->
<!--            </v-list-item__content>-->
<!--          <v-list-item-action start>-->

<!--            <v-btn stacked
  ↪ append-icon="mdi-text-box-edit-outline"
  ↪ variant="outlined"></v-btn>-->
<!--            <v-btn stacked
  ↪ append-icon="mdi-delete-empty-outline"
  ↪ variant="outlined"></v-btn>-->
<!--          </v-list-item-action>-->
<!--        </template>-->
<!--      </v-list-item>-->
<!--    </v-list>-->


  <v-table id="tasks_table" max-height="70vh">
```

```
94    <v-toolbar
95        height="20%"
96        class="mb-0"
97        dark
98        prominent
99        v-for="task in TasksArray"
100   >
101
102
103 <!--       <div v-
    ↪  if="task['task_params']['task_status']==='Assigned'"
    ↪  class="square" id="status_Assigned">-->
104 <!--       </div>-->
105 <!--       <div v-
    ↪  if="task['task_params']['task_status']==='In_Process'"
    ↪  class="square" id="status_In_Process">-->
106 <!--       </div>-->
107 <!--       <div v-
    ↪  if="task['task_params']['task_status']==='Finalized'"
    ↪  class="square" id="status_Finalized">-->
108 <!--       </div>-->
109 <!--       <div v-
    ↪  if="task['task_params']['task_status']==='Closed'"
    ↪  class="square" id="status_Closed">-->
110 <!--       </div>-->
```

```
111     <div v-
        ↪  if="task['task_params']['task_status']==='Assigned'
        ↪  && !this.$vuetify.display.mobile" class="card"
        ↪  id="Assigned">
112       <h2>Создана</h2>
113     </div>
114     <div v-
        ↪  if="task['task_params']['task_status']==='In_Proces
        ↪  && !this.$vuetify.display.mobile" class="card"
        ↪  id="In_Process">
115       <h2>В работе</h2>
116     </div>
117     <div v-
        ↪  if="task['task_params']['task_status']==='Finalized
        ↪  && !this.$vuetify.display.mobile" class="card"
        ↪  id="Finalized">
118       <h2>Завершается</h2>
119     </div>
120     <div v-
        ↪  if="task['task_params']['task_status']==='Closed'
        ↪  && !this.$vuetify.display.mobile" class="card"
        ↪  id="Closed">
121       <h2>Завершена</h2>
122     </div>
123
124     <v-card
125         :title="task['task_name']"
126         :text="task['task_description']"
```

```
127             id="toc"
128         ><div class="square"></div></v-card>
129         <v-spacer></v-spacer>
130
131 <!--     <v-card-->
132 <!--
    ↪  :subtitle="task['task_params']['task_status']"-->
133 <!--             style="min-width: 9vw; max-width: 9vw;
    ↪  background-color: #009E73"-->
134 <!--     ></v-card>-->
135
136
137
138 <!--     <v-card-->
139 <!--             class="mx-auto"-->
140 <!--             style="background-image: none;
    ↪  background-color: transparent"-->
141 <!--     >-->
142 <!--       <template v-slot:prepend>-->
143 <!--         <v-avatar-->
144 <!--             class="ma-3"-->
145 <!--             size="2.3em"-->
146 <!--             rounded="0"-->
147 <!--         >-->
148 <!--           <v-img src="/init.svg"></v-img>-->
149 <!--         </v-avatar>-->
150 <!--       </template>-->
151 <!--     </v-card>-->
```

```
152
153       <v-list-item
154           v-if="!this.$vuetify.display.mobile"
155           :title="task['task_params']['task_initiator']"
156           :subtitle="'Инициатор'"
157           class="pa-2 ma-2"
158           >
159         <template v-slot:prepend>
160           <v-avatar
161               class="ma-3"
162               size="2.3em"
163               rounded="0"
164           >
165             <v-img src="/init.svg"></v-img>
166           </v-avatar>
167         </template>
168       </v-list-item>
169
170       <v-list-item
171           v-if="!this.$vuetify.display.mobile"
172           :title="task['task_params']['task_executor']"
173           :subtitle="'Исполнитель'"
174           class="pa-2 ma-2"
175           >
176               <template v-slot:prepend>
177                 <v-avatar
178                     class="ma-3"
179                     size="2.3em"
```

```
180                     rounded="0"
181                 >
182                   <v-img src="/exec.svg"></v-img>
183                 </v-avatar>
184               </template>
185     </v-list-item>
186
187
188     <v-btn icon id="btn-card-task-edit"
        ↪  v-on:click="edit_task(task)">
189       <v-icon>mdi-text-box-edit-outline</v-icon>
190     </v-btn>
191     <v-btn icon id="btn-card-task-delete"
        ↪  v-on:click="delete_task(task['task_uuid'])">
192       <v-icon>mdi-delete-empty-outline</v-icon>
193     </v-btn>
194   </v-toolbar>
195   </v-table>
196   </v-container>
197
198   <v-container v-if="AddTask" id="container_add_Task"
      ↪  style="padding: 0px;" >
199     <v-card
200         color="rgb(33 178 178)"
201         theme="dark"
202         variant="outlined"
203     >
```

```
204         <div class="d-flex flex-no-wrap
             ↪  justify-space-between">
205           <div>
206             <v-card-title
                 ↪  v-if="!this.$vuetify.display.mobile"
                 ↪  class="text-h5" >
207               Добавление задачи
208             </v-card-title>
209             <v-card-title
                 ↪  v-if="this.$vuetify.display.mobile"
                 ↪  class="text-h5" >
210               Добавление
211             </v-card-title>
212             <v-card-actions>
213               <v-btn
214                   v-if="!this.$vuetify.display.mobile"
215                   class="ms-2"
216                   variant="outlined"
217                   size="small"
218                   v-on:click="add_new_task()"
219                   id = "v-btn_on_header"
220               ><v-icon id="task_on_header"
                     ↪  >mdi-arrow-left</v-icon>Вернуться к
                     ↪  списку задач</v-btn>
221               <v-btn
222                   v-if="this.$vuetify.display.mobile"
223                   class="ms-2"
224                   variant="outlined"
```

```
225                 size="small"
226                 v-on:click="add_new_task()"
227                 id = "v-btn_on_header"
228             ><v-icon id="task_on_header"
                ↪  >mdi-arrow-left</v-icon>К списку
                ↪  задач</v-btn>
229           </v-card-actions>
230         </div>
231         <v-avatar
232             class="ma-3"
233             size="6em"
234             rounded="0"
235         >
236           <v-img id="todo_svg"
                ↪  src="/add_task.svg"></v-img>
237         </v-avatar>
238       </div>
239     </v-card>
240
241     <v-form>
242       <v-container>
243         <v-row>
244           <v-col
245               cols="12"
246               md="12"
247           >
248             <v-text-field
249                 v-model="task_name"
```

```
250                 :rules="nameRules"
251                 :counter="10"
252                 label="Название задачи"
253                 required
254               ></v-text-field>
255             </v-col>
256           </v-row>
257
258           <v-row>
259             <v-col
260                 cols="12"
261                 md="12"
262             >
263               <v-text-field
264                   v-model="task_description"
265                   :rules="nameRules"
266                   :counter="10"
267                   label="Описание задачи"
268                   required
269               ></v-text-field>
270             </v-col>
271           </v-row>
272
273           <v-row>
274             <v-col
275                 cols="12"
276                 md="1"
```

```
277                 style="text-align: center; padding-right:
                      ↪  0px;"
278               >
279                 <v-avatar
280                     class="ma-3"
281                     size="3em"
282                     rounded="0"
283                 >
284                   <v-img src="/init.svg"></v-img>
285                 </v-avatar>
286               </v-col>
287               <v-col
288                   cols="12"
289                   md="3"
290               >
291                 <v-text-field
292                     v-model="task_initiator"
293                     :rules="nameRules"
294                     :counter="10"
295                     label="Инициатор"
296                     required
297                 ></v-text-field>
298               </v-col>
299
300
301               <v-col
302                   cols="12"
303                   md="1"
```

```
304                 style="text-align: center; padding-right:
                      ↪  0px;"
305               >
306                 <v-avatar
307                     class="ma-3"
308                     size="3em"
309                     rounded="0"
310                 >
311                   <v-img src="/exec.svg"></v-img>
312                 </v-avatar>
313               </v-col>
314               <v-col
315                   cols="12"
316                   md="3"
317               >
318
319                 <v-text-field
320                     v-model="task_executor"
321                     :rules="nameRules"
322                     :counter="10"
323                     label="Исполнитель"
324                     required
325                 ></v-text-field>
326               </v-col>
327
328               <v-col
329                   cols="12"
330                   md="4"
```

```
331             >
332 <!--             "['Assigned', 'In_Process',
    ↪  'Finalized', 'Closed' ]"-->
333             <v-select
334                 v-model="task_status"
335                 :items="['Создана', 'В работе',
                     ↪  'Завершается', 'Завершена' ]"
336                 label="Статус задачи"
337             ></v-select>
338           </v-col>
339
340         </v-row>
341         <v-btn
342             class="ms-2"
343             variant="outlined"
344             size="small"
345             v-on:click="create_new_task()"
346             id = "v-btn_on_header"
347         ><v-icon id="task_on_header"
             ↪  >mdi-check-bold</v-icon>Добавить
             ↪  задачу</v-btn>
348         <v-btn
349             class="ms-2"
350             variant="outlined"
351             size="small"
352             v-on:click="clear_task_params()"
353             id = "v-btn_on_header"
```

```
354          ><v-icon id="task_on_header" >mdi-delete-
              ↪  outline</v-icon>Очистить</v-btn>
355        </v-container>
356
357      </v-form>
358
359    </v-container>
360
361    <v-container v-if="EditTask" id="container_add_Task"
       ↪  style="padding: 0px;" >
362      <v-card
363          color="rgb(33 178 178)"
364          theme="dark"
365          variant="outlined"
366      >
367        <div class="d-flex flex-no-wrap
           ↪  justify-space-between">
368          <div>
369            <v-card-title class="text-h5" >
370              Правка
371            </v-card-title>
372            <v-card-actions>
373              <v-btn
374                  v-if="!this.$vuetify.display.mobile"
375                  class="ms-2"
376                  variant="outlined"
377                  size="small"
378                  v-on:click="show_edit_task()"
```

```
379                 id = "v-btn_on_header"
380             ><v-icon id="task_on_header"
                ↪  >mdi-arrow-left</v-icon>Вернуться к
                ↪  списку задач</v-btn>
381             <v-btn
382                 v-if="this.$vuetify.display.mobile"
383                 class="ms-2"
384                 variant="outlined"
385                 size="small"
386                 v-on:click="show_edit_task()"
387                 id = "v-btn_on_header"
388             ><v-icon id="task_on_header"
                ↪  >mdi-arrow-left</v-icon>К списку
                ↪  задач</v-btn>
389           </v-card-actions>
390         </div>
391         <v-avatar
392             class="ma-3"
393             size="6em"
394             rounded="0"
395         >
396           <v-img id="todo_svg" src="/edit.svg"></v-img>
397         </v-avatar>
398       </div>
399     </v-card>
400
401     <v-form>
402       <v-container>
```

```
403          <v-row>
404            <v-col
405                cols="12"
406                md="12"
407            >
408              <v-text-field
409                  v-model="task_name"
410                  :rules="nameRules"
411                  :counter="10"
412                  label="Название задачи"
413                  required
414              ></v-text-field>
415            </v-col>
416          </v-row>
417
418          <v-row>
419            <v-col
420                cols="12"
421                md="12"
422            >
423              <v-text-field
424                  v-model="task_description"
425                  :rules="nameRules"
426                  :counter="10"
427                  label="Описание задачи"
428                  required
429              ></v-text-field>
430            </v-col>
```

```
431             </v-row>
432
433             <v-row>
434               <v-col
435                   cols="12"
436                   md="1"
437                   style="text-align: center; padding-right:
                        ↪  0px;"
438               >
439                 <v-avatar
440                     class="ma-3"
441                     size="3em"
442                     rounded="0"
443                 >
444                   <v-img src="/init.svg"></v-img>
445                 </v-avatar>
446               </v-col>
447               <v-col
448                   cols="12"
449                   md="3"
450               >
451                 <v-text-field
452                     v-model="task_initiator"
453                     :rules="nameRules"
454                     :counter="10"
455                     label="Инициатор"
456                     required
457                 ></v-text-field>
```

```
458             </v-col>
459
460
461             <v-col
462                 cols="12"
463                 md="1"
464                 style="text-align: center; padding-right:
                      ↪  0px;"
465             >
466               <v-avatar
467                   class="ma-3"
468                   size="3em"
469                   rounded="0"
470               >
471                 <v-img src="/exec.svg"></v-img>
472               </v-avatar>
473             </v-col>
474             <v-col
475                 cols="12"
476                 md="3"
477             >
478
479               <v-text-field
480                   v-model="task_executor"
481                   :rules="nameRules"
482                   :counter="10"
483                   label="Исполнитель"
484                   required
```

```
485             ></v-text-field>
486           </v-col>
487
488           <v-col
489               cols="12"
490               md="4"
491           >
492             <!--              "['Assigned', 'In_Process',
                ↪  'Finalized', 'Closed' ]"-->
493             <v-select
494                 v-model="task_status"
495                 :items="['Создана', 'В работе',
                    ↪  'Завершается', 'Завершена' ]"
496                 label="Статус задачи"
497             ></v-select>
498           </v-col>
499
500         </v-row>
501         <v-btn
502             class="ms-2"
503             variant="outlined"
504             size="small"
505             v-on:click="update_task()"
506             id = "v-btn_on_header"
507         ><v-icon id="task_on_header" >mdi-content-
            ↪  save-outline</v-icon>Сохранить
            ↪  изменения</v-btn>
508       </v-container>
```

```
509
510       </v-form>
511
512     </v-container>
513   </template>
514
515   <script>
516   import axios from 'axios'
517   export default {
518     name: "TaskList",
519     data() {
520       return {
521         TasksArray: [],
522         AddTask: false,
523         task_name: "",
524         task_description: "",
525         task_initiator: "",
526         task_executor: "",
527         task_status: "",
528         task_uuid: "",
529         EditTask: false,
530       }
531     },
532     mounted() {
533       console.log(this.$vuetify.display.mobile)
534       axios({
535         method: "get",
536         url: `/api/tasks`,
```

```
537       })
538           .then(response => {
539             console.log(response.data);
540             // console.log(response);
541             this.TasksArray = response.data;
542           });
543       // fetch("/api/tasks")
544       //      .then((response) =>response.json())
545       //      .then((json) => {
546       //            console.log(json)});
547     },
548     methods: {
549       show_edit_task() {
550         this.update_task_list()
551         this.EditTask = !this.EditTask
552       },
553       edit_task(selected_task) {
554         console.log(" ")
555         console.log("Edit")
556         console.log(selected_task)
557         this.task_name = selected_task['task_name']
558         this.task_description =
            ↪  selected_task['task_description']
559         this.task_executor =
            ↪  selected_task['task_params']['task_executor']
560         this.task_initiator =
            ↪  selected_task['task_params']['task_initiator']
561         this.task_uuid = selected_task['task_uuid']
```

```
562         if
             ↪  (selected_task['task_params']['task_status']==='As
             ↪  {
563           this.task_status ='Создана'
564           // "['Assigned', 'In_Process', 'Finalized',
               ↪  'Closed' ]"
565           // "['Создана', 'В работе', 'Завершается',
               ↪  'Завершена' ]"
566         }
567         else if
             ↪  (selected_task['task_params']['task_status']==='In
             ↪  {
568           this.task_status ='В работе'
569         }
570         else if
             ↪  (selected_task['task_params']['task_status']==='Fi
             ↪  {
571           this.task_status ='Завершается'
572         }
573         else if
             ↪  (selected_task['task_params']['task_status']
             ↪  ==='Closed') {
574           this.task_status ='Завершена'
575         }
576         else {
577           this.task_status ='Создана'
578         }
579         this.show_edit_task()
```

```
580         // this.$emit('pointselected', point_name);
581         // this.$router.replace({ name: "SettingUp" });
582       },
583       update_task() {
584         let task_status_real
585         if (this.task_status==='Создана') {
586           task_status_real = 'Assigned'
587           // "['Assigned', 'In_Process', 'Finalized',
                ↪  'Closed' ]"
588           // "['Создана', 'В работе', 'Завершается',
                ↪  'Завершена' ]"
589         }
590         else if  (this.task_status==='В работе') {
591           task_status_real = 'In_Process'
592         }
593         else if  (this.task_status==='Завершается') {
594           task_status_real = 'Finalized'
595         }
596         else if  (this.task_status==='Завершена') {
597           task_status_real = 'Closed'
598         }
599         else {
600           task_status_real = 'Assigned'
601         }
602         axios.put('/api/tasks/'+this.task_uuid, {
603           task_name: this.task_name,
604           task_description: this.task_description,
605           task_params: {
```

```
606             task_initiator: this.task_initiator,
607             task_executor: this.task_executor,
608             task_status: task_status_real,
609           }
610         })
611             .then((response) => {
612               console.log(response);
613               this.show_edit_task()
614             }, (error) => {
615               console.log(error);
616             });
617       },
618       delete_task(selected_task_uuid) {
619         console.log(" ")
620         console.log("Delete")
621         console.log(selected_task_uuid)
622         axios({
623           method: "delete",
624           url: `/api/tasks/`+selected_task_uuid,
625         })
626             .then(response => {
627               axios({
628                 method: "get",
629                 url: `/api/tasks`,
630               })
631                   .then(response => {
632                     this.TasksArray = response.data;
633                   });
```

```
634             });
635       },
636       update_task_list() {
637         axios({
638           method: "get",
639           url: `/api/tasks`,
640         })
641             .then(response => {
642               console.log(response.data);
643               this.TasksArray = response.data;
644             });
645       },
646       clear_task_params (){
647         this.task_name = "";
648         this.task_description = "";
649         this.task_initiator= "";
650         this.task_executor = "";
651         this.task_status = "";
652       },
653       add_new_task() {
654         this.update_task_list()
655         this.AddTask = !this.AddTask
656         this.clear_task_params()
657         console.log(" ")
658         console.log("Add")
659       },
660       create_new_task() {
661         let task_status_real
```

```
662         if (this.task_status==='Создана') {
663           task_status_real = 'Assigned'
664           // "['Assigned', 'In_Process', 'Finalized',
                ↪  'Closed' ]"
665           // "['Создана', 'В работе', 'Завершается',
                ↪  'Завершена' ]"
666         }
667         else if  (this.task_status==='В работе') {
668           task_status_real = 'In_Process'
669         }
670         else if  (this.task_status==='Завершается') {
671           task_status_real = 'Finalized'
672         }
673         else if  (this.task_status==='Завершена') {
674           task_status_real = 'Closed'
675         }
676         else {
677           task_status_real = 'Assigned'
678         }
679         axios.post('/api/tasks/add', {
680           task_name: this.task_name,
681           task_description: this.task_description,
682           task_params: {
683             task_initiator: this.task_initiator,
684             task_executor: this.task_executor,
685             task_status: task_status_real,
686           }
687         })
```

```
688             .then((response) => {
689               console.log(response);
690               this.add_new_task()
691             }, (error) => {
692               console.log(error);
693             });
694       }
695     }
696 }
697 </script>
698 <style scoped>
699
700 #container_add_Task{
701   min-width: 60vw;
702   margin: 0px;
703 }
704 #tasks_table {
705   background: transparent;
706 }
707 #status_Assigned {
708   height: 100%;
709   width: calc(3vw);
710   background-color: #21B2B2;
711 }
712 #status_In_Process {
713   height: 100%;
714   width: 3%;
715   background-color: #009E73;
```

```
716 }
717 #status_Finalized {
718   height: 100%;
719   width: 3%;
720   background-color: #06614C;
721 }
722 #status_Closed {
723   height: 100%;
724   width: 3%;
725   background-color: #182C25;
726 }
727
728 .v-card {
729   /*box-shadow: rgb(192 187 137 / 60%) 0px 30px 60px
      ↪  -12px, rgb(194 189 139 / 37%) 0px 18px 36px
      ↪  -18px;*/
730   /*box-shadow: rgba(238, 232, 169, 0.33) 0px 20px
      ↪  80px;*/
731   border-color: #EEE8A9;
732   background-color: #000000;
733   opacity: 0.8;
734   background-image:  repeating-radial-gradient( circle
      ↪  at 0 0, transparent 0, #000000 10px ),
      ↪  repeating-linear-gradient( rgba(33, 178, 247,
      ↪  0.15), rgba(33, 178, 178, 0.3));
735 }
736
737 button{
```

```
738   margin-top: 2%;
739 }
740
741 .v-btn {
742   color: #e6f4f1;
743   border-color: #21B2B2;
744 }
745
746 .v-card .v-card-title  {
747   padding-top: 4%;
748   color: #e6f4f1;
749   font-size: 1.8rem !important;
750   font-weight: bold !important;
751   /*font-size: calc(0.002vw + 0.002vh + 1.5vmin);*/
752 }
753 .v-card .v-card-subtitle{
754   color: #e6f4f1;
755   font-size: 1rem !important;
756 }
757
758 .v-card-item__content .v-card-title  {
759 }
760
761 .v-toolbar {
762   background: rgba(0, 0, 0, 0.63);
763   color: white;
764 }
765
```

```
766
767 span .v-btn__content {
768   color: #e6f4f1;
769 }
770
771 #toc{
772   background-color: transparent;
773   background-image: none;
774   border-color: transparent;
775   box-shadow: none;
776   padding-top: 15px;
777   padding-bottom: 15px;
778   color: white;
779 }
780 /*#toc.v-card-title {*/
781 /*  font-size: 1rem !important;*/
782 /*}*/
783
784 #btn-card-task-edit{
785   margin-top: 0%;
786   /*color: #324B4B;*/
787   color: #005c64;
788 }
789
790 #btn-card-task-delete {
791   margin-top: 0%;
792   color: #A73160;
793 }
```

```
794
795 #v-btn_on_header  {
796   background-color: rgb(33 178 178 / 15%);
797 }
798
799 #task_on_header {
800   margin-right: 3%;
801   color: #EEE8A9;
802 }
803 .v-btn--icon .v-icon {
804   --v-icon-size-multiplier: 0.85;
805 }
806
807 #todo_svg{
808   /*filter: drop-shadow( 5px 5px 8px rgba(33, 178,
      ↪  178, 0.37));*/
809   filter: drop-shadow( 1px 1px 9px rgb(22, 22, 22));
810   height: 80%;
811 }
812
813 .v-table__wrapper {
814   max-height: calc(60vh) !important;
815 }
816
817 .card {
818   min-width: 120px;
819   width: 7vw;
820   height: 6vh;
```

```
821   background: transparent;
822   position: relative;
823   display: flex;
824   place-content: center;
825   place-items: center;
826   overflow: hidden;
827   border-radius: 0px;
828   margin-right: 1%;
829   margin-left: 1%;
830 }
831
832 #Assigned h2 {
833   z-index: 1;
834   color: rgba(33, 178, 178, 0.3);
835   font-size: 1.1em;
836   font-weight: bold;
837 }
838
839 #Assigned::before{
840   -webkit-filter: blur(15px);
841   content: '';
842   position: absolute;
843   width: 300px;
844   /*background-image: linear-gradient(180deg, rgb(33,
      ↪  178, 178), rgba(42, 169, 169, 0.8));*/
845   background-image:radial-gradient(circle, rgba(33,
      ↪  178, 178, 0.3), rgba(34, 179, 179, 1));
846   height: 130%;
```

```
847   animation: rotBGimg 6s linear infinite;
848   transition: all 0.2s linear;
849 }
850
851 #In_Process h2 {
852   z-index: 1;
853   color: rgba(33, 178, 178, 0.5);
854   font-size: 1.1em;
855   font-weight: bold;
856 }
857
858 #In_Process::before{
859   -webkit-filter: blur(15px);
860   content: '';
861   position: absolute;
862   width: 300px;
863   /*background-image: linear-gradient(180deg, rgb(33,
      ↪  178, 178), rgba(42, 169, 169, 0.8));*/
864   background-image:radial-gradient(circle, rgba(33,
      ↪  178, 178, 0.5), rgba(33, 178, 178, 1));
865   height: 130%;
866   animation: rotBGimg 6s linear infinite;
867   transition: all 0.2s linear;
868 }
869
870 #Finalized h2 {
871   z-index: 1;
872   color: rgba(33, 178, 178, 0.65);
```

```
873   font-size: 1.1em;
874   font-weight: bold;
875 }
876
877 #Finalized::before{
878   -webkit-filter: blur(15px);
879   content: '';
880   position: absolute;
881   width: 300px;
882   /*background-image: linear-gradient(180deg, rgb(33,
      ↪  178, 178), rgba(42, 169, 169, 0.8));*/
883   background-image:radial-gradient(circle, rgba(33,
      ↪  178, 178, 0.8), rgba(33, 178, 178, 1));
884   height: 130%;
885   animation: rotBGimg 6s linear infinite;
886   transition: all 0.2s linear;
887 }
888
889 #Closed h2 {
890   z-index: 1;
891   color: rgba(33, 178, 178, 1);
892   font-size: 1.1em;
893   font-weight: bold;
894 }
895
896 #Closed::before{
897   -webkit-filter: blur(15px);
898   content: '';
```

```
899   position: absolute;
900   width: 300px;
901   /*background-image: linear-gradient(180deg, rgb(33,
      ↪  178, 178), rgba(42, 169, 169, 0.8));*/
902   background-image:radial-gradient(circle, rgba(33,
      ↪  178, 178, 1), rgba(33, 178, 178, 1));
903   height: 130%;
904   animation: rotBGimg 6s linear infinite;
905   transition: all 0.2s linear;
906 }
907
908
909 @keyframes rotBGimg {
910   from {
911     transform: rotate(0deg);
912   }
913
914   to {
915     transform: rotate(360deg);
916   }
917 }
918
919 .card::after {
920   content: '';
921   position: absolute;
922   background: rgba(9, 9, 9, 0.88);
923   box-shadow: 0 8px 32px 0 rgba( 31, 38, 135, 0.37 );
924   backdrop-filter: blur( 10px );
```

```
925   -webkit-backdrop-filter: blur( 10px );
926   border-radius: 10px;
927   inset: 3px;
928   border-radius: 0px;
929 }
930 /* .card:hover:before {
931   background-image: linear-gradient(180deg, rgb(81,
      ↪  255, 0), purple);
932   animation: rotBGimg 3.5s linear infinite;
933 } */
934
935 .v-form {
936   background: rgba(0, 0, 0, 0.63);
937 }
938
939
940 </style>
```

Листинг 4.3 API

```
1 from fastapi import FastAPI, Depends, HTTPException,
   ↪  status
2 from uuid import UUID
3 from typing import List
4 from sqlalchemy.orm import Session
5
6 import app.models as models
7 from app.database import engine, create_session
8 from app.schema import TaskSchema, TaskSchemaExpanded
```

```
from app.models import Task


app = FastAPI()


@app.on_event("startup")
async def startup():
    models.Base.metadata.drop_all(bind=engine)  # await
    models.Base.metadata.create_all(bind=engine)


@app.get("/tasks",
    response_model=List[TaskSchemaExpanded],
    status_code=200)
async def read_tasks(db: Session =
    Depends(create_session)):
    tasks = db.query(Task).all()
    return tasks


@app.post('/tasks/add',
    response_model=TaskSchemaExpanded,
    status_code=201)
async def create_task(task: TaskSchema, db: Session =
    Depends(create_session)):
    new_task = Task(**task.dict())
    db.add(new_task)
    db.commit()
```

```
31      return new_task
32
33
34  @app.put("/tasks/{task_uuid}",
    ↪   response_model=TaskSchemaExpanded,
    ↪   status_code=200)
35  async def update_task(task_uuid: UUID, task:
    ↪   TaskSchema, db: Session =
    ↪   Depends(create_session)):
36      updating_task = db.query(Task).get(task_uuid)
37      if updating_task is not None:
38          updating_task.task_name = task.task_name
39          updating_task.task_description =
             ↪  task.task_description
40          updating_task.task_params =
             ↪  task.task_params.dict()
41          # for field, value in
             ↪  task.task_params.dict().items():
42          #     setattr(updating_task, field, value)
43          db.commit()
44          db.refresh(updating_task)
45      else:
46          raise HTTPException(status.HTTP_404_NOT_FOUND,
             ↪  detail="Task with UUID does not exist ")
47      return updating_task
48
49
50  @app.delete("/tasks/{task_uuid}")
```

```
def delete_task(task_uuid: UUID, db: Session =
    Depends(create_session)):
    task = db.get(Task, task_uuid)
    if not task:
        raise HTTPException(status_code=404,
            detail="Task not found")
    db.delete(task)
    db.commit()
    return {"Task deleted successfully": True}
```

Листинг 4.4 Валидация данных

```
from pydantic import BaseModel, Extra
from uuid import uuid4, UUID
from datetime import datetime
from typing import Literal


class TaskParamsModel(BaseModel, extra=Extra.forbid):
    task_initiator: str        # инициатор
    task_executor: str        # исполнитель
    task_status: Literal['Assigned', 'In_Process',
        'Finalized', 'Closed']
    # task_deadline: datetime.date  # крайний срок
    # task_param_2:   int


class TaskSchema(BaseModel):
    task_name: str
```

```
    task_description: str
    task_params: TaskParamsModel
    class Config:
        orm_mode = True


class TaskSchemaExpanded(TaskSchema):
   task_uuid: UUID = uuid4()
   class Config:
       orm_mode = True
```

Листинг 4.5 Docker compose

```
version: "4"
services:
  web:
    build: .
    command: bash -c 'while !</dev/tcp/db/5432; do
     ↪ sleep 1; done; uvicorn app.main:app --host
     ↪ 0.0.0.0'
    ports:
      - 8008:8000
    environment:
      - DATABASE_URL=
      postgresql://mgknn:tst@db:5432/postgres
    depends_on:
      - db
    volumes:
      - .:/app
```

```
15   db:
16     image: postgres:13.3
17     environment:
18       POSTGRES_DB: "postgres"
19       POSTGRES_USER: "mgknn"
20       POSTGRES_PASSWORD: "tst"
21     ports:
22       - "5432:5432"
23     volumes:
24       - postgres_data:/var/lib/postgresql/data/
25 volumes:
26   postgres_data
```