МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет _____ЦОО ФИСТ_____
Кафедра ____Информационные системы_____
Дисциплина _Интернет программирование_____

КУРСОВОЙ ПРОЕКТ (РАБОТА)

Тема

Разработка web приложения Таск-трекер

Выполнил студент _____ /____Ф.С Пудовкин____/
Подпись инициалы, фамилия

Курс _____5_____ Группа _____ЦИСТбв-51_____

Направление/ специальность ____информационные системы и технологии_

Руководитель _____
должность, ученая степень, ученое звание
_____
фамилия, имя, отчество

Дата сдачи:
« » _____ 2022 г.

Дата защиты:
« » _____ 2022 г.

Оценка: _____

Ульяновск
2022 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет ЦОО ФИСТ
Кафедра Информационные системы
Дисциплина Базы данных

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

студенту ЦИСТбв-51 Пудовкин Ф.С.
группа фамилия, инициалы

Тема проекта (работы)
Разработка web приложения Таск-трекер

Срок сдачи законченного проекта (работы) « » _____2022 г.

Исходные данные к проекту (работе)
Для управления задачами требуется разоработать web приложение, Использую языки программирования python, js, html, css. Сервер реализовать на фреймворке django, для общения с сервером реализовать rest api, в котором реализовать методы получения, создания, редактирования данных. Клиентскую часть реализовать на фреймворке vuejs, с помощью возможностей которого реализовать верстку, стилистическое оформление, и методы http клиента для общения с серверной часть.

Содержание пояснительной записки (перечень подлежащих разработке вопросов)
1. Описание поставленной задачи
2. Модель хранения данных представленная в ER-диаграмме
3. Описание структуры разработанной базы данных

Руководитель _____ / _____/
должность подпись инициалы, фамилия
« » _____2022 г.

Студент _____/**Ф.С Пудовкин**/
подпись инициалы, фамилия
« » _____ 2022 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ОТЗЫВ
руководителя на курсовой проект (работу)

студента _____Пудовкина Федора Сергеевича_____
фамилия, имя и отчество

Факультет___ЦОО ФИСТ__группа ЦИСТбв-51____курс____5__

Дисциплина_____Интернет программирование_____

Тема проекта (работы)
Разработка web приложения Таск-трекер
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Руководитель _____ /_____/
должность, учёная степень, ученое звание подпись инициалы, фамилия

«__»_____ 2022 г.

Оглавление

Введение

С появлением Web-технологии компьютер начинают использовать совершенно новые слои населения Земли. Можно выделить две наиболее характерные группы, находящиеся на разных социальных полюсах, которые были стремительно вовлечены в новую технологию, возможно, даже помимо их собственного желания. С одной стороны, это были представители элитарных групп общества - руководители крупных организаций, президенты банков, топ - менеджеры, влиятельные государственные чиновники и т. д. С другой стороны, это были представители широчайших слоев населения - домохозяйки, пенсионеры, дети.

При появлении технологии Web компьютеры повернулись лицом к этим двум совершенн
противоположным категориям потенциальных пользователей. Элиту объединяла одна черта – в силу высочайшей ответственности и практически стопроцентной занятости “большие люди” никогда не пользовались компьютером; типичной была ситуация, когда с компьютером работал секретарь. В какой-то момент времени они поняли, что компьютер им может быть полезен, что они могут результативно использовать то небольшое время, которое можно выделить на работу за компьютером. Они вдруг поняли, что компьютер - это не просто модная и дорогая игрушка, но инструмент получения актуальной информации для бизнеса. При этом им не нужно было тратить сколько-нибудь заметного времени, чтобы освоить технологию работы с компьютером (по сравнению с тем, как это было раньше).

Спектр социальных групп, подключающихся к сети Интернет и ищущих информацию в WWW, все время расширяется за счет пользователей, не относящихся к категории специалистов в области информационных технологий. Это врачи, строители, историки, юристы, финансисты, спортсмены, путешественники, священнослужители, артисты, писатели, художники. Список можно продолжать бесконечно. Любой, кто ощутил полезность и незаменимость Сети для своей профессиональной деятельности или увлечений, присоединяется к огромной армии потребителей информации во «Всемирной Паутине».

Web-технология полностью перевернула наши представления о работе с информацией, да и с компьютером вообще. Оказалось, что традиционные параметры развития вычислительной техники - производительность, пропускная способность, емкость запоминающих устройств - не учитывали главного «узкого места» системы - интерфейса с человеком. Устаревший механизм взаимодействия человека с информационной системой сдерживал внедрение новых технологий и уменьшал выгоду от их применения. И только когда интерфейс между человеком и компьютером был упрощен до естественности восприятия обычным человеком, последовал беспрецедентный взрыв интереса к возможностям вычислительной техники.

С развитием технологий гипертекстовой разметки в Интернете стало появляться всё больше сайтов, тематика которых была совершенно различной – от сайтов крупных компаний, повествующих об успехах компании и её провалах, до сайтов маленьких фирм, предлагающих посетить их офисы в пределах одного города.

Развитие Интернет-технологий послужило толчком к появлению новой ветки в Интернете – Интернет - форумов. Стали появляться сайты, и даже целые порталы, на которых люди со всех уголков планеты могут общаться, получать ответы на любые вопросы и, даже, заключать деловые сделки.

Постановка задачи

Необходимо реализовать курсовую работу трекер задач. Использую языки программирования python, js, html, css. Сервер реализовать на фреймворке django, для общения с сервером реализовать rest api, в котором реализовать методы получения, создания, редактирования данных. Клиентскую часть реализовать на фреймворке vuejs, с помощью возможностей которого реализовать верстку, стилистическое оформление, и методы http клиента для общения с серверной часть.

Верстка статических страниц приложения

Для простоты реализации верстки веб статических страниц используется язык гипер статической разметки html а для разметки стилей язык описания внешнего вида документа css. Для простоты верстки мы будем использовать фреймворк bootstrap, это набор инструментов который включает в себя шаблоны html + css для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

Для корневого шаблона реализуем следующую верстку

```
<!DOCTYPE html>

<html lang="ru">

  <head>

    <meta charset="utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width,initial-scale=1.0">

    <link rel="icon" href="<%= BASE_URL %>favicon.ico">

    <title><%= htmlWebpackPlugin.options.title %></title>

  </head>

  <body>

    <noscript>

      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't work properly without JavaScript enabled. Please enable it to continue.</strong>

    </noscript>

    <div class="container-fluid">

          <nav class="navbar navbar-expand-lg bg-light fixed-top">

              <div class="container-fluid">

                <a class="navbar-brand" href="#">Менеджер задач</a>

                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

                  <span class="navbar-toggler-icon"></span>

                </button>

                <div class="collapse navbar-collapse" id="navbarSupportedContent">

                  <ul class="navbar-nav me-auto mb-2 mb-lg-0">
```

```
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="/">Список задач</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="/add">Добавить задачу</a>
                </li>
              </ul>
            </div>
          </div>
        </nav>
  </div>
  <div class="container">
     <div id="app"></div>
  </div>

  <br/>
  <br/>
  <br/>
  <br/>
  <br/>
  <br/>
  <br/>
 <footer class="footer mt-auto py-3 bg-light">
  <div class="container">
     <span class="text-muted">Курсовая работа</span>
  </div>
 </footer>
 </body>
</html>
```

Для верстки формы добавления задачи

```
<div class="submit-form">
  <div v-if="!submitted">
    <div class="form-group">
      <label for="title">Заголовок</label>
      <input
        type="text"
        class="form-control"
        id="title"
        required
        v-model="task.title"
        name="title"
      />
    </div>
    <div class="form-group">
      <label for="description">Описание</label>
      <textarea
        class="form-control"
        id="description"
        required
        v-model="task.description"
        name="description"
      />
    </div>

    <button @click="saveTutorial" class="btn btn-success">Добавить задачу</button>
  </div>

  <div v-else>
    <h4>Вы успешно создали задачу</h4>
    <button class="btn btn-success" @click="newTutorial">Добавить новую задачу</button>
  </div>
```

```
</div>
```

Для отображения и редактирования выбраной задачи

```
<div>
  <div v-if="currentTask" class="edit-form">
    <h4>Задача</h4>
    <form>
      <div class="form-group">
        <label for="title">Заголовок</label>
        <input type="text" class="form-control" id="title"
          v-model="currentTask.title"
        />
      </div>
      <div class="form-group">
        <label for="description">Описание</label>
        <input type="text" class="form-control" id="description"
          v-model="currentTask.description"
        />
      </div>
      <div class="form-group">
        <label for="state">Статус</label>
        <input type="text" class="form-control" id="state"
          v-model="currentTask.state"
        />
      </div>
    </form>
    <button type="submit" class="btn btn-primary"
      @click="updateTask"
    >
      Обновить
    </button>
    <p>{{ message }}</p>
  </div>
```

```
    <div v-else>
      <br />
      <p>Нажмите на задачу...</p>
    </div>
  </div>
```

Для вывода списка задач

```
<div class="list row row-cols-1 row-cols-sm-1 row-cols-md-1 g-3">
    <div class="col col-sm-1 col-md-3">
      <h4>Список задач</h4>
      <ul class="list-group">
        <li class="list-group-item"
          :class="{ active: index === currentIndex }"
          v-for="(task, index) in tasks"
          :key="index"
          @click="setActiveTutorial(task, index)"
        >
         {{ index + 1}}: {{ task.title }}
        </li>
      </ul>

    </div>
    <div class="col-sm-1 col-md-6">
      <div v-if="currentTask">
        <h4>Задача</h4>
        <div>
          <label><strong>Заголовок:</strong></label> {{ currentTask.title }}
        </div>
        <div>
          <label><strong>Описание:</strong></label> {{ currentTask.description }}
        </div>
        <div>
```

```
        <label><strong>Статус:</strong></label> {{ currentTask.state }}
      </div>
      <div>
        <label><strong>Время создания:</strong></label> {{ currentTask.created_at }}
      </div>
      <div>
        <label><strong>Время изменения:</strong></label> {{ currentTask.last_modified }}
      </div>

      <router-link :to="'/tasks/' + currentTask.id" type="button" class="btn
btn-primary">Изменить</router-link>
    </div>
    <div v-else>
      <br />
      <p>Пожалуйста выберете задачу...</p>
    </div>
  </div>
</div>
```

В верстке можно заметить не стандартные html параметры тегов.

К примеру:

```
v-for="(task, index) in tasks"
        :key="index"
        @click="setActiveTutorial(task, index)
```

Такие параметры взяты из инструментария фреймворка vue js, они необходимы для взаимодействия с java script и dom-деревом шаблона.

Разработка архитектуры web-приложения

Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная логика Представления реализуется в Django уровнем Шаблонов. Но для создания reat api, мы используем модуль django-rest-framework.

В django rest framework архитектура немного меняется, но не отходит далеко от модели MVC. Мы можем провести аналогию с mvc. Model (Модель), Serializers (Представление) и View (Контроллер). Понятия немного смешаются но смысл остаётся тем же.

Описание бизнес логики приложения

Модели данных приложения описываются в классах, следующим образом. После чего генерируется миграция приложения и применяется для базы данных. После создания моделей, можно описать виды и контроллеры

Модели

```
TASK_PRIORITY_FIELDS = ('state', '-priority', '-deadline')

class Project(models.Model):
    class Meta:
        verbose_name = _("project")
        verbose_name_plural = _("projects")

    title = models.CharField(_("title"), max_length=200)
    description = models.TextField(_("description"), max_length=2000, null=True, blank=True)
    created_at = models.DateTimeField(_("created at"), auto_now_add=True, editable=False)
    last_modified = models.DateTimeField(_("last modified"), auto_now=True, editable=False)


class State(enum.Enum):
    TO_DO = '00-to-do'
    IN_PROGRESS = '10-in-progress'
    BLOCKED = '20-blocked'
    DONE = '30-done'
    DISMISSED = '40-dismissed'
```

```
class Priority(enum.Enum):
    LOW = '00-low'
    NORMAL = '10-normal'
    HIGH = '20-high'
    CRITICAL = '30-critical'


class Task(models.Model):
    class Meta:
        verbose_name = _("Task")
        verbose_name_plural = _("Tasks")
    STATES = (
        (State.TO_DO.value, _('To Do')),
        (State.IN_PROGRESS.value, _('In Progress')),
        (State.BLOCKED.value, _('Blocked')),
        (State.DONE.value, _('Done')),
        (State.DISMISSED.value, _('Dismissed'))
    )

    PRIORITIES = (
        (Priority.LOW.value, _('Low')),
        (Priority.NORMAL.value, _('Normal')),
        (Priority.HIGH.value, _('High')),
        (Priority.CRITICAL.value, _('Critical')),
    )

    title = models.CharField(_("title"), max_length=200)
    project = models.ForeignKey(Project, blank=True, null=True, on_delete=models.PROTECT)

    description = models.TextField(_("description"), max_length=2000, null=True, blank=True)
    deadline = models.DateField(_("deadline"), null=True, blank=True)
    user = models.ForeignKey(User, related_name='tasks_assigned', verbose_name=_('assigned
to'),
```

```
                                on_delete=models.SET_NULL, null=True, blank=True)
    state = models.CharField(_("state"), max_length=20, choices=STATES,
default=State.TO_DO.value)
    priority = models.CharField(_("priority"), max_length=20, choices=PRIORITIES,
default=Priority.NORMAL.value)
    created_by = models.ForeignKey(User, related_name='users_created',
verbose_name=_('created by'),
                                   on_delete=models.SET_NULL, null=True)
    created_at = models.DateTimeField(_("created at"), auto_now_add=True, editable=False)
    last_modified = models.DateTimeField(_("last modified"), auto_now=True, editable=False)


    class Meta:
        indexes = [
            models.Index(fields=TASK_PRIORITY_FIELDS, name='tasks_task_priority_idx'),
        ]
```

Представления

```
class TaskSerializer(serializers.ModelSerializer):
    class Meta:
        model = Task
        fields = '__all__'
class CreateTaskSerializer(serializers.ModelSerializer):
    class Meta:
        model = Task
        exclude = ['last_modified', 'created_at', 'created_by']
class UpdateTaskSerializer(serializers.ModelSerializer):
    class Meta:
        model = Task
        # fields = '__all__'
        exclude = ['last_modified', 'created_at', 'created_by']
class ChangeStatusTaskSerializer(serializers.ModelSerializer):
    class Meta:
        model = Task
```

```
        fields = ['state']
class CreateProjectSerializer(serializers.ModelSerializer):
    class Meta:
        model = Project
        exclude = ['last_modified', 'created_at', 'created_by']
class UpdateProjectSerializer(serializers.ModelSerializer):
    class Meta:
        model = Project
        exclude = ['last_modified', 'created_at', 'created_by']
```

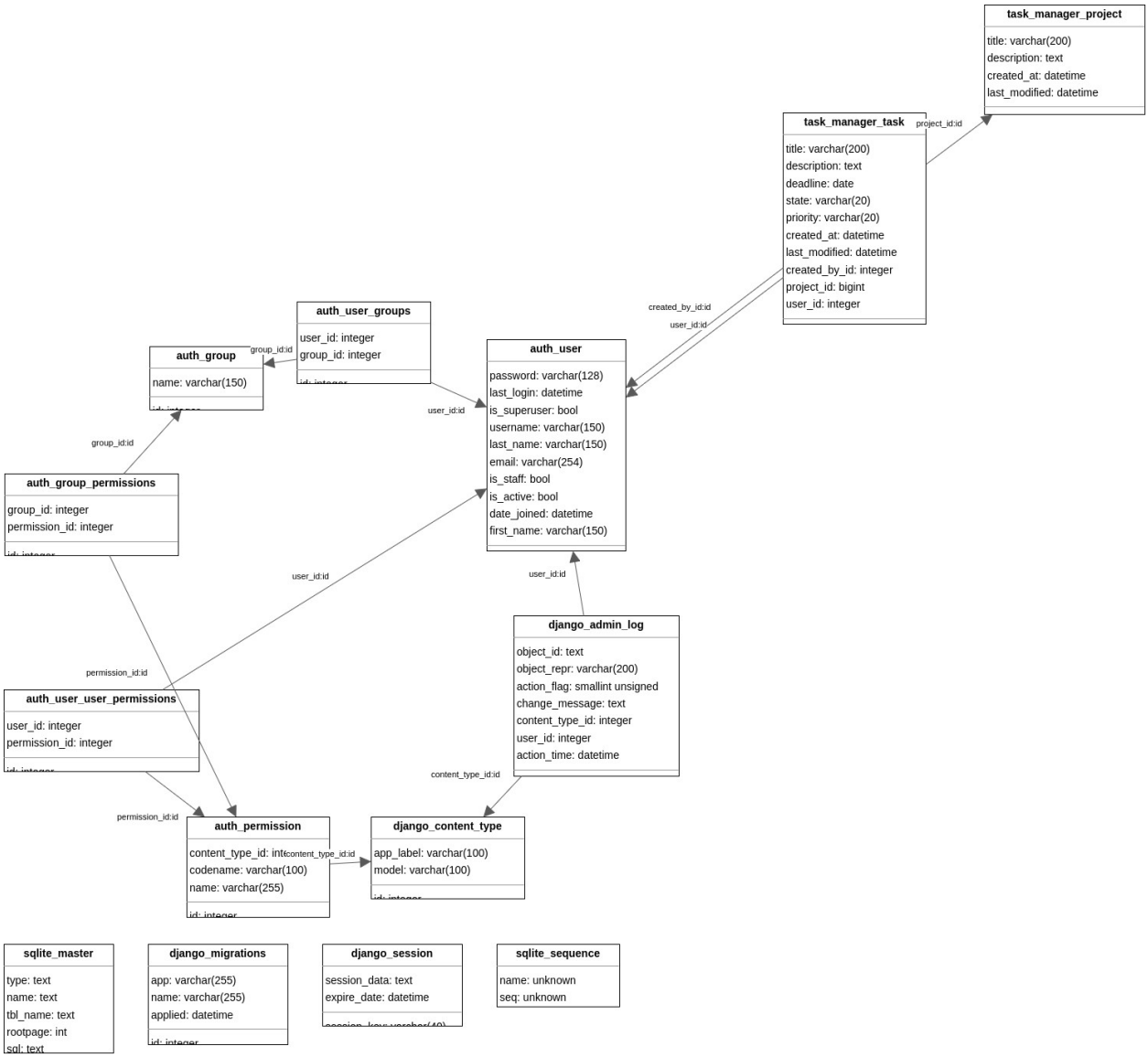
Контроллеры

```
class TaskDetailAPIView(RetrieveAPIView):
    queryset = Task.objects.all()
    serializer_class = TaskSerializer
    lookup_field = 'id'
class TaskAPIView(ListAPIView):
    queryset = Task.objects.all()
    serializer_class = TaskSerializer
class CreateTaskAPIView(CreateAPIView):
    queryset = Task.objects.all()
    serializer_class = CreateTaskSerializer
    def perform_create(self, serializer):
        serializer.save(
            created_at=timezone.now())
class UpdateTaskAPIView(RetrieveUpdateAPIView):
    queryset = Task.objects.all()
    serializer_class = UpdateTaskSerializer
    lookup_field = 'id'
    def perform_update(self, serializer):
        serializer.save(last_modified=timezone.now())
class CreateProjectAPIView(CreateAPIView):
    queryset = Project.objects.all()
```

```
    serializer_class = CreateProjectSerializer
    def perform_create(self, serializer):
        serializer.save(created_at=timezone.now())
class UpdateProjectAPIView(RetrieveUpdateAPIView):
    queryset = Project.objects.all()
    serializer_class = UpdateProjectSerializer
    lookup_field = 'id'

    def perform_update(self, serializer):
        serializer.save(last_modified=timezone.now())
class ChangeStatusTaskAPIView(RetrieveUpdateAPIView):
    queryset = Task.objects.all()
    serializer_class = ChangeStatusTaskSerializer
    lookup_field = 'id'
    def perform_update(self, serializer):
        serializer.save(last_modified=timezone.now())
```

Полная диаграмма классов вместе с служебными таблицами django фреймворка.



Разработка контроллеров для работы клиентской части приложения

Во vuejs для в основном описываются функции для работы с dom деревом, http клиентами.

Классы и методы классов описываются обычно в общих пакетах.

Для реализации http клиента к серверной части напишем класс с использованием библиотеки axios.

```
import axios from "axios";

export default axios.create({
  baseURL: "http://localhost:8000/api",
  headers: {
    "Content-type": "application/json"
  }
});
```

Также добавим класс и методы класса для вызова api запросов. Каждый метод класса вызывает свой собственный запрос к сервису, д

```
import http from "../http-client-common";

class TaskDataService {
  getAll() {
    return http.get("/task/?format=json");
  }

  get(id) {
    return http.get(`/task/${id}`);
  }

  create(data) {
```

```
    return http.post("/task/add/", data);
  }

  update(id, data) {
    return http.put(`/task/edit/${id}`, data);
  }

  delete(id) {
    return http.delete(`/tasks/${id}`);
  }

}

export default new TaskDataService();
```

Разработка динамического интерфейса приложения.

В vuejs код приложения хранится в компонентах. Каждый компонент имеет 3 директивы template, script, style. Соответственно template отвечает за шаблон html верстку, script за Js код отвечающий за манипуляции интерфейсом и style отвечает за описание стилей.

В дерективе script методы для манипуляций с интерфесом описываются в дерективе methods

Пример компонент addTask.vue, темплейт мы можем увидеть код верстки, с вставками шаблонизатора vuejs и вставками вызов js методов, в скрипте описывается методы data внутри него возвращаются структуры данных для шаблонизатора, в дерективе методы описывается функция для в которой вызывается класс и метод http клиента.

```
<template>
  <div class="submit-form">
    <div v-if="!submitted">
      <div class="form-group">
        <label for="title">Заголовок</label>
        <input
          type="text"
          class="form-control"
          id="title"
          required
          v-model="task.title"
          name="title"
        />
      </div>
      <div class="form-group">
        <label for="description">Описание</label>
        <textarea
          class="form-control"
          id="description"
          required
```

```
        v-model="task.description"
        name="description"
      />
    </div>

    <button @click="saveTutorial" class="btn btn-success">Добавить задачу</button>
  </div>

  <div v-else>
    <h4>Вы успешно создали задачу</h4>
    <button class="btn btn-success" @click="newTutorial">Добавить новую задачу</button>
  </div>
 </div>
</template>

<script>
import DataService from "../services/DataService";
export default {
 name: "add-task",
 data() {
  return {
   task: {
    id: null,
    title: "",
    description: "",
    // published: false
   },
   submitted: false
  };
 },
 methods: {
  saveTutorial() {
```

```
      let data = {
        'title': this.task.title,
        'description': this.task.description
      };
      DataService.create(data)
        .then(response => {
          this.task.id = response.data.id;
          console.log(response.data);
          this.submitted = true;
        })
        .catch(e => {
          console.log(e);
        });
    },

    newTutorial() {
      this.submitted = false;
      this.task = {};
    }
  }
};
</script>

<style>
.submit-form {
  max-width: 300px;
  margin: auto;
}
</style>
```

Разработка стилей страниц

Для стилистического оформления используется язык css и framework bottstrap.

Для достижения своих целей мы не будет использовать набор классов представленных во фреймворке. Примеры стилей можно найти на сайте https://getbootstrap.com/docs/5.0/getting-started/introduction/

Заключение

Изучены основы web программирования. Проведено исследование предметной области. На основе результатов; созданы модели данных; имеющиеся контроллеры данных выполняющие функцию по обработке и выполнению операций оптимизированы под поставленные требования; созданы новые веб-формы и элементы дизайна для ввода вывода и обработки данных.

Результатом выполненной работы стали полноценные информационные системы позволяющие создавать задачи и манипулировать ими интуитивно-понятные конечному пользователю готовые к расширению и дальнейшей модернизации.

Литература

1. Марк Лутц Изучаем Python 1, 2 том
2. William S. Vincent Django for beginners
3. Элизабет Робсон, Эрик Фримен Изучаем HTML, XHTML и CSS (Head First)
4. Ольга Фролова Learning Vue.js 2.
5. Марк Массе REST API Design Rulebook