После того как всё будет готово, сконвертируйте docx в pdf и добавьте в начало документа сопроводительную документацию в следующем порядке:

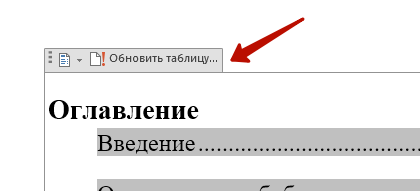
Титульный лист

Задание

Аннотация

Отзыв

Обновите оглавление после того как закончите писать курсовую



Оглавление

[Введение 3](#_Toc102319817)

[Определения, обобщения и сокращения 4](#_Toc102319818)

[Описание предметной области 5](#_Toc102319819)

[Описание прикладного процесса 5](#_Toc102319820)

[Формирование требований 5](#_Toc102319821)

[Проектирование 6](#_Toc102319822)

[Используемый стек технологий 6](#_Toc102319823)

[Системная архитектура 6](#_Toc102319824)

[Архитектура данных 7](#_Toc102319825)

[Программная архитектура 8](#_Toc102319826)

[Разработка 9](#_Toc102319827)

[Реализация серверного API 9](#_Toc102319828)

[Реализация пользовательского интерфейса 10](#_Toc102319829)

[Заключение 11](#_Toc102319830)

[Список использованной литературы 12](#_Toc102319831)

# Введение

Объектом разработки является система управления контентом на языке программирования TypeScript с использованием фреймворка Nest – для серверной части приложения и языка программирования JavaScript, без использования фреймворков – для клиентской части приложения. Выбранный фреймворк один из самых поддерживаемых и распространённых.

Целью работы является разработка веб-приложения и пользовательского интерфейса, анализ требования и моделирование процессов, средств автоматизации и архитектуры информационной системы.

В ходе работы были получены следующие результаты:

* Серверная часть системы, принимающая запросы.
* Клиентская часть системы, предоставляющая интерфейс пользователя.
* База данных для хранения информации о пользователях, файлах, а также служебной информации внутри системы.

# Определения, обобщения и сокращения

*Браузер* – …

*Фреймворк* - …

*Роль «администратор» внутри вашего приложения* - …

и т.д.

# Описание предметной области

## Описание прикладного процесса

Разрабатываемое веб-приложение, это система управления контентом. Благодаря таким системам можно, не имея навыков программирования, и не обращаясь к веб-разработчикам, самостоятельно управлять содержимым внутри сайта.

При работе с классическими системами управления контентом можно выделить такие типовые задачи как редактирование и удаление текущих, создание новых страниц или разделов.

## Формирование требований

В ходе анализа прикладного процесса был получен следующий список функциональных требований:

Модуль работы с разделами должен иметь:

* Возможность фильтрации по названию
* Возможность создать раздел
* Отображение всех существующих разделов

(описываем какие **операции** можно сделать внутри приложения в разрезе модулей)

Нефункциональные требования

Разрабатываемая система не является публичной, поэтому основным требованием выступает её закрытость, путем обязательного прохождения процесса авторизации и аутентификации.

Так же должна быть возможность работы с приложением напрямую, через общие программные интерфейсы, описанные по спецификации OpenAPI версии не ниже 3.0

# Проектирование

## Используемый стек технологий

Решение создания именно веб-приложения обусловлено тем, что необходимо было обеспечить доступ к системе с любого устройства, в любое время. Веб-приложение решает этот вопрос, а также снимает вопрос обновлений на стороне клиента.

Проект использует стек стандартных технологий, характерный для большинства веб-приложений: HTML, CSS, Javascript.

В качестве веб-сервера используется Express либо Fastify.

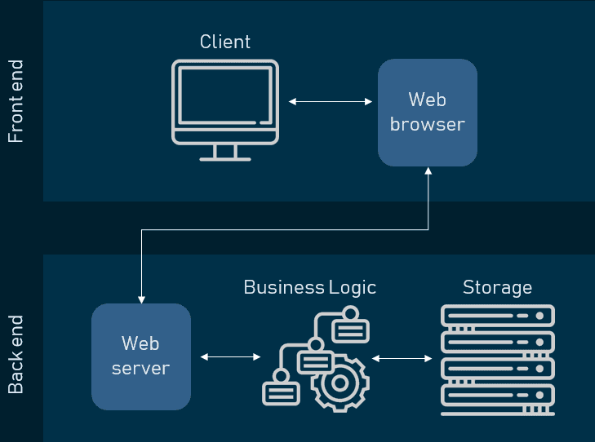
В качестве базы данных используется PostgreSQL, ввиду того, что данная СУБД является самой стабильной в данной связке. Библиотекой для работы с данными была выбрана Prisma или TypeORM.

В проекте используется свободная распределённая система управления версиями Git, хранилищем исходных кодов является крупнейший веб-сервис GitHub, а в качестве хостинга используется облачный сервис Heroku.

## Системная архитектура

Рисуем Deployment Diagram в нотации UML и отображаем на ней все сервисы с которыми мы взаимодействуем, в случае если у вас нет внешних зависимостей, рисуем три ноды Client => Server => Database. Указываем протоколы на стрелках. (HTTPS, ODBC (для бд), и т.д.)

Далее описываем словами то, что отобразили на схеме.

  
Рисунок 1. Системная архитектура приложения.

## Архитектура данных

Вставьте артефакты моделирования из readme.md полученные в рамках третьей лабораторной работы. Опишите какую информацию содержит каждая сущность.

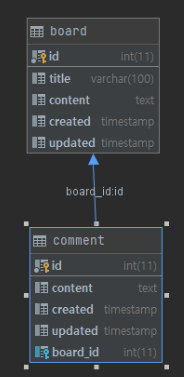


Рисунок 2. Схема таблиц базы данных

Board – Предоставляет информацию о доске

Comment – Предоставляет информацию о комментариях на доске

## Программная архитектура

Распишите каждый модуль, какие сущности в нём есть и какую роль они выполняют.

Таблица 1. Отношения модулей и классов

|  |  |  |
| --- | --- | --- |
| **Название модуля** | **Название класса** | **Назначение класса** |
| EventModule | EventController | Сервис (Контроллер) |
| EventRequestDto | Модель передачи данных |

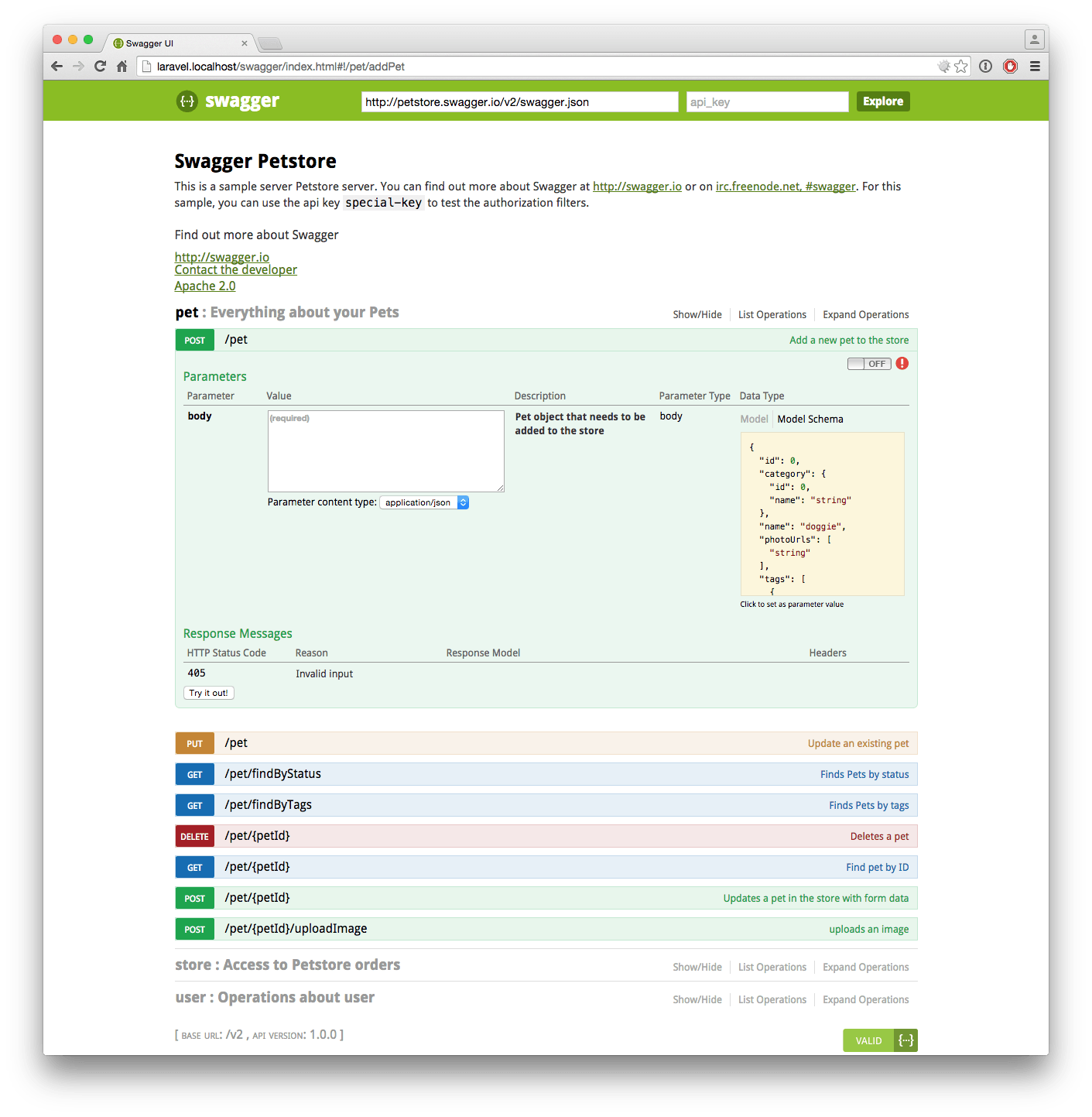
Таблица 2. Описание классов

|  |  |
| --- | --- |
| **Название класса** | **Описание класса** |
| EventService | Класс, используемый для работы с событиями. Осуществляет поиск сущностей, отображение полного списка событий, а также позволяет удалить то или иное событие из общего каталога. |

# Разработка

## Реализация серверного API

В качестве описания программного интерфейса был выбран инструмент, поддерживающий стандарт OAS 3.0 – Swagger. Далее представлена полученная документация API полученная автоматически по директивам, указанным в декораторах различных методов и структурах данных внутри разрабатываемой информационной системы.

Рисунок N. Программный интерфейс серверного API.

## Реализация пользовательского интерфейса

Вставляем скриншоты, нумеруем и подписываем картинки снизу. Описываем словами что изображено на рисунке словами из функциональных требований (т.е. нужно показать, что все функции, которые мы указали ранее действительно реализованы и их видно на скриншотах)

# Заключение

В ходе выполнения курсовой работы был проведён анализ работы классических систем управления сайтами, исходя из которого были выявлены и сформированы требования к разрабатываемому веб-приложению.

Исходя из выбранной архитектуры и наложенных ограничений были сформированы требования к используемым технологиям внутри модулей. Была спроектирована архитектура данных, программная и системная архитектура в виде набора диаграмм в нотации UML.

Опираясь на выше изложенные требования и стек технологий было разработано веб-приложение и пользовательский интерфейс в рамках дисциплины «Web-программирование».

Таким образом, все поставленные ранее цели были выполнены.

Разработанное приложение является результатом данной курсовой работы.

# Список использованной литературы

1. Мартин Фаулер - Архитектура корпоративных программных приложений. Издательский дом "Вильяме'. 2006 г.

2. Флэнаган, Дэвид. JavaScript. Полное руководство, 7-е изд. : Пер. с англ. — СПб. : ООО “Диалектика”, 2021. — 720 с .

3. Янг А., Мек Б., Кантелон М. Node.js в действии. 2-е изд. — СПб.: Питер, 2018. — 432 с.

4. Браун И.Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. 2-е издание. — СПб.: Питер, 2021. — 336 с.

5. Современный учебник JavaScript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/. – Дата доступа: 04.05.2021.