МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ШКОЛА КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ПРОГРАМММНОГО ОБЕСПЕЧЕНИЯ

ОТЧЕТ

О РАЗРАБОТКЕ ПРОЕКТА ПО ДИСЦИПЛИНЕ

«ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ»

на тему: Разработка планировщика задач, реализующего методологию GTD, для студентов ТюмГУ

Выполнили студенты МОиАИС 4 курса:

Иванова Дарья Владимировна

Киселева Анастасия Андреевна

Руководитель проекта:

Ялдыгин Валерий Борисович

Тюмень-2024

Оглавление

[ВВЕДЕНИЕ 3](#_Toc185202652)

[Глава 1. Требования к проекту 5](#_Toc185202653)

[Глава 2. Теоретическое обоснование проекта 7](#_Toc185202654)

[Глава 3. Описание используемых технологий 8](#_Toc185202655)

[Глава 4. Теоретические разработки 11](#_Toc185202656)

[Глава 5. Архитектура приложения 12](#_Toc185202657)

[Глава 6. Функционал приложения 14](#_Toc185202658)

[Глава 7. Разработка приложения 16](#_Toc185202659)

[7.1 Разработка фронтенда web-приложения 16](#_Toc185202660)

[7.2 Разработка бэкенда web-приложения 21](#_Toc185202661)

[Глава 8. Тестирование проекта 26](#_Toc185202662)

[Заключение 28](#_Toc185202663)

[Список литературы 29](#_Toc185202664)

# ВВЕДЕНИЕ

В настоящее время достаточно остро встает проблема прокрастинации. По результатам опроса студентов в октябре 2022 года [1] было выявлено, что 35% опрошенных имеют высокий уровень прокрастинации, 41% – средний и 21% – низкий. Автор статьи подчеркивает, что студентам с высоким уровнем прокрастинации важно продумывать промежуточные задачи для достижения итоговой цели и научиться правильно расставлять приоритеты, правильно распределять время в течение дня.

В ходе исследования [2] был проведён онлайн-опрос с использованием психодиагностической методики К. Лэя «Шкала общей прокрастинации». В опросе приняли участие студенты различных факультетов БНТУ в возрасте от 17 до 20 лет. Среди них было 63 % юношей и 37 % девушек.

В процессе опроса учитывалась средняя оценка успеваемости студентов по результатам экзаменационных сессий. Также были проведены интервью, в ходе которых была получена информация о том, как студенты справляются с факторами, влияющими на их обучение.

Исследование проводилось в три этапа в период с сентября по декабрь 2023 года. Это позволило отследить динамику изменений. В ходе исследования было установлено, что более 50% респондентов имеют ярко выраженную склонность к откладыванию дел на потом, а у 36% эта склонность проявляется в умеренной форме. Кроме того, 65% студентов испытывают постоянное напряжение из-за экзаменов и других стрессовых ситуаций.

В своей статье автор предлагает рекомендации о ведении списка дел, где студент мог бы отмечать выполненные и невыполненные задачи. В качестве такого списка может выступать планировщик задач.

Ещё одно исследование подчеркивает, что для эффективного обучения студент должен обладать навыками самоорганизации, для этого рекомендуется применение стратегий управления временем, такие как определение приоритетности задач, разбиение задач на более мелкие этапы, использование планировщиков или цифровых календарей [3].

С помощью нашей системы студенты смогут держать список дел и заметки в одном месте, имея возможность обозначать как «мягкие», так и «жесткие» дедлайны, а также наделять задания метками контекста, в соответствии с методологией GTD. Кроме того, важно не только иметь возможность записывать свои дела и распределять их по дню или же неделе, но и правильно оценивать нагрузку.

Проект направлен на студентов Тюменского государственного университета. В рамках проекта планируется интеграция с системой modeus для автоматизации процесса добавления учебных встреч в планировщик. Это позволит студентам учитывать в расписании другие события и задачи, не связанные напрямую с занятиями.

# Глава 1. Требования к проекту

В соответствии с поставленной проблемой нами были выдвинуты требования к проекту:

* Функционал создания, изменения, удаления задач, проектов, событий. Привязка задач к проектам и событиям. Добавление проектов и событий в календарь.
* Интеграция с modeus для получения расписания студентов ТюмГУ.
* Функционал создания ссылки на свое расписание, которой можно поделиться с другим человеком, имея возможность скрыть подробности своих задач и событий.

Заказчиком выступает Ялдыгин Валерий Борисович. В перечень требований заказчика вошли следующие пункты:

* Пользовательский интерфейс и логика работы должны быть реализованы в соответствии с методикой GTD.
* Подсказка при поиске по заметкам на основании истории поиска пользователя.

Для реализации проекта были поставлены следующие задачи:

* Исследовать разные группы людей и их потребностей для определения целевой аудитории.
* Провести анализ области технологий, реализующих ПО для планирования задач для студентов.
* Разработать архитектуру системы, включая клиентскую и серверную части приложения, которое будет предоставлять функционал согласно методике GTD.
* Разработать структуру базы данных, которая будет содержать информацию о проектах, задачах, событиях пользователя, включая дедлайны и метки контекста. Кроме того, база данных должна содержать данные пользователей, которые будут использованы для регистрации (имя, почта и хэш пароля).
* Разработать интерфейс программы, который будет предоставлять пользователю календарь и инструменты для составления списков и создания проектов.
* Реализовать механизм авторизации для студентов по имени, почте и паролю.
* Реализовать бизнес-логику приложения: CRUD-операции для сущностей и другие методы взаимодействия с ними.

Ограничения проекта:

* На создание прототипа отводится 3,5 месяца.
* В разработке участвует команда из двух человек.
* Планировщик предусмотрен для индивидуальной работы студента, функционал для командной работы не рассматривается.
* Проект не будет включать интеграцию с внешними календарями (например, Google Calendar, Microsoft Outlook): ограниченные временные ресурсы и сложности в интеграции с разными API.

Эти ограничения помогут сфокусироваться на ключевых аспектах проекта, улучшить его реализацию в заданных рамках и сделать работу над проектом более управляемой и целенаправленной.

# Глава 2. Теоретическое обоснование проекта

В основе логики приложения и требований к пользовательскому интерфейсу лежит методология Getting Things Done (GTD), предложенная Дэвидом Алленом.

Данная методология широко исследуется и признается полезной для управления задачами и временем. Исследования в области управления временем отмечают, что GTD снижает стресс и повышает концентрацию благодаря внедрению системного подхода к организации задач. Метод позволяет освободить ум от "перегрузки" информацией, что способствует улучшению креативности и продуктивности. Это подтверждается статьями в области психологии и личной эффективности [4-6].

# Глава 3. Описание используемых технологий

В клиентской части приложения будет реализована архитектура FSD (feature sliced design). Данная архитектура имеет несколько ключевых для нашего проекта преимуществ, как например масштабируемость, устойчивость к изменению и рефакторингу из-за принципов данной архитектуры.

Frontend предполагает использование следующих технологий:

1. React. Frontend-часть приложения будет реализована на React. Это библиотека для реализации нативного и web-интерфейса.
2. MUI, Material UI. Библиотека компонентов для React, позволяющая облегчить вёрстку страниц.
3. Tanstack React-Query. Представляет собой библиотеку для отправки запросов на сторонние сервисы, включая множество полезных функций как кэш для каждого запроса, автоматическая переотправка для актуализации данных на странице. Встроенные возможности данной библиотеки позволят в рамках нашего проекта не использовать дополнительные библиотеки для управления глобальным состоянием приложения.
4. React-form-hook. Библиотека для React, направленная на создание и взаимодействие форм. Наше приложение будет использовать разные формы, состояние и значения которых надо отслеживать (пример: создание задачи, контекста, прч.).
5. Material-react-table. Библиотека для React, расширение MUI.
6. Zustand. Для валидации вводимых пользователем значений.
7. React Calendar. Библиотека для React, необходимая для реализации разных видов календаря.
8. React router dom. Маршрутизация.
9. Typescript. Язык программирования, разработанный Microsoft. Добавляет JavaScript поддержку типизации.
10. Axios. Отправка запросов на удаленный сервер.

Backend предполагает использование следующих технологий:

1. C# (ASP.NET Core, Entity Framework Core), PostgreSQL.
2. ASP.NET Core будет использовать C#. В ряде независимых тестов производительности ASP.NET Core превосходит многие другие фреймворки, включая Node.js, Django и Ruby on Rails, а производительность будет важна в нашем проекте для работы с календарями, задачами. Это важно для отклика и быстроты работы нашего приложения.
3. Entity Framework Core. Это фреймворк, облегчающий рутинные задачи, связанные с базой данных. Механизм миграций позволяет вносить изменения в модель данных и автоматически синхронизировать эти изменения с базой данных, что значительно упрощает разработку и развертывание новых версий приложения. Кроме того, EF Core сочетается с ASP.NET Core, который мы выбрали для реализации проекта: оба инструмента поддерживают кросс-платформенную разработку и развертывание на разных операционных системах.

PostgreSQL будет выступать в качестве основной базы данных для проекта, где будут хранится данные пользователей, включая описание задачи, дедлайны, приоритеты, контексты (по методике GTD). Данная СУБД была выбрана кроссплатформенность, поддержка разных типов данных, включая неструктурированные (JSON), что будет полезно в нашем проекте для обеспечения сохранности данных. Также PostgreSQL поддерживает шифрование базы данных на уровне базы данных, что очень важно для обеспечения безопасности чувствительных данных.

# Глава 4. Теоретические разработки

В ходе работы была смоделирована структура базы данных. Были использованы теоретические основы нормализации для обеспечения целостности и минимизации избыточности данных.

REST API обеспечит четкое разделение между frontend и backend-частями проекта [7]. Это позволит взаимодействовать с базой данных и алгоритмами машинного обучения через стандартный интерфейс запросов, не привязывая клиента к специфической реализации серверной части. В проекте будет несколько ключевых эндпоинтов для работы с данными, например:

* POST /tasks — создание новой задачи.
* GET /tasks/{id} — получение информации о задаче по идентификатору.
* PUT /tasks/{id} — обновление информации о задаче.
* GET /schedule/{user\_id} — получение расписания для пользователя.

Будут использоваться основные методы HTTP:

* GET — для получения данных.
* POST — для создания новых данных.
* PUT — для обновления существующих данных.
* DELETE — для удаления задач.

Запросы и ответы будут обмениваться данными в формате JSON, что является стандартом для REST API. Это упростит взаимодействие с данными как фронтенда.

# Глава 5. Архитектура приложения

Архитектура приложения включает в себя клиентскую и серверную части, базу данных, API.

Клиентскую часть включает следующий функционал:

* Отправка запросов к серверу (через REST API).
* Отображение данных, полученных с сервера, в виде, соответствующем установленным для календарей критериям.
* Формы для создания пользователей, задач, проектов.

На серверной части предполагается обработка запросов от клиента и взаимодействие с базой данных.

Основной функционал включает в себя:

* Обработка CRUD-запросов (Create, Read, Update, Delete) основных сущностей.
* Механизм авторизации/аутентификации пользователей по имени, почте и паролю.
* Управление связанными сущностями (пользователи, события, проекты, задачи).
* Функционал создания публичной ссылки на свое расписание.

База данных предполагает хранение данных о пользователях и об их расписании, обеспечение их целостности. Схема представлена на рис. 1.

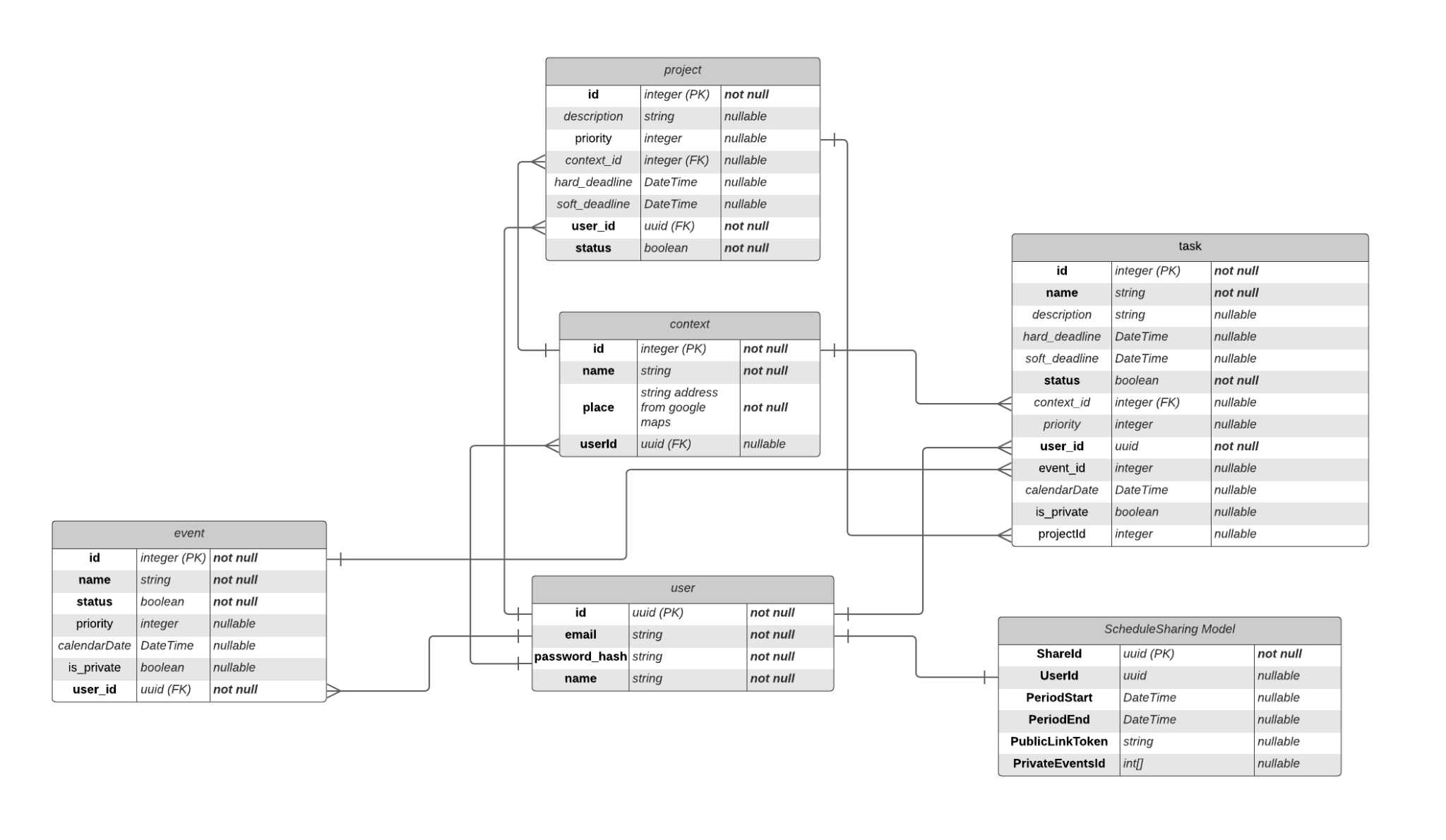


Рисунок 1 – схема базы данных

API (RestAPI, Representational State Transfer Application Programming Interface) – интерфейс между сервером и клиентом, ключевой элемент архитектуры, обеспечивающий взаимодействие между frontend и backend-частями системы.

# Глава 6. Функционал приложения

Функционал системы:

* Регистрация для новых пользователей и авторизация для существующих.
* Возможность создавать проекты и задачи, назначать им срок выполнения (дедлайн) — «мягкий» или «жесткий» — и метку контекста (наименование).
* Отправка уведомлений о пропущенных дедлайнах, о предстоящих важных делах.
* Возможность делиться своим расписанием посредством общедоступной ссылки. Функционал отправки своего расписания другому человеку должен включать в себя возможность выбора начала и конца периода.
* Возможность добавить для задания геометку (метка контекста), в зависимости от которой впоследствии будет отправляться уведомление о наличии задачи.

Требования к пользовательскому интерфейсу:

* Цветовое кодирование: необходима возможность визуально выделять разные составляющие, такие как проект, контексты, ссылки, дела по их важности, дедлайны и прочее. Исследования [8-9] показывают, что разделение по цветам позволяет быстрее ориентироваться в приложении. Было доказано, что цвет привлекает внимание пользователей быстрее, чем черно-белая или однотонная информация, что делает его эффективным инструментом для визуальной организации данных.
* Календари: годовой, месячный, недельный. Годовой и месячный календарь включает в себя только color coding задач и событий. В еженедельном календаре отображаются задачи, дедлайны с описанием и доп. события, которые могут быть не привязаны ни к какому контексту.
* Дедлайны: «мягкие» (желаемая дата выполнения), «жёсткие». Визуально должны быть отличимы друг от друга посредством цвета, шрифта, положения в календаре.
* Списки: дел, делегированных дел, событий.

# Глава 7. Разработка приложения

## 7.1 Разработка фронтенда web-приложения

В ходе работы над проектом был разработан интерфейс для недельного календаря, макет представлен на рис. 2.

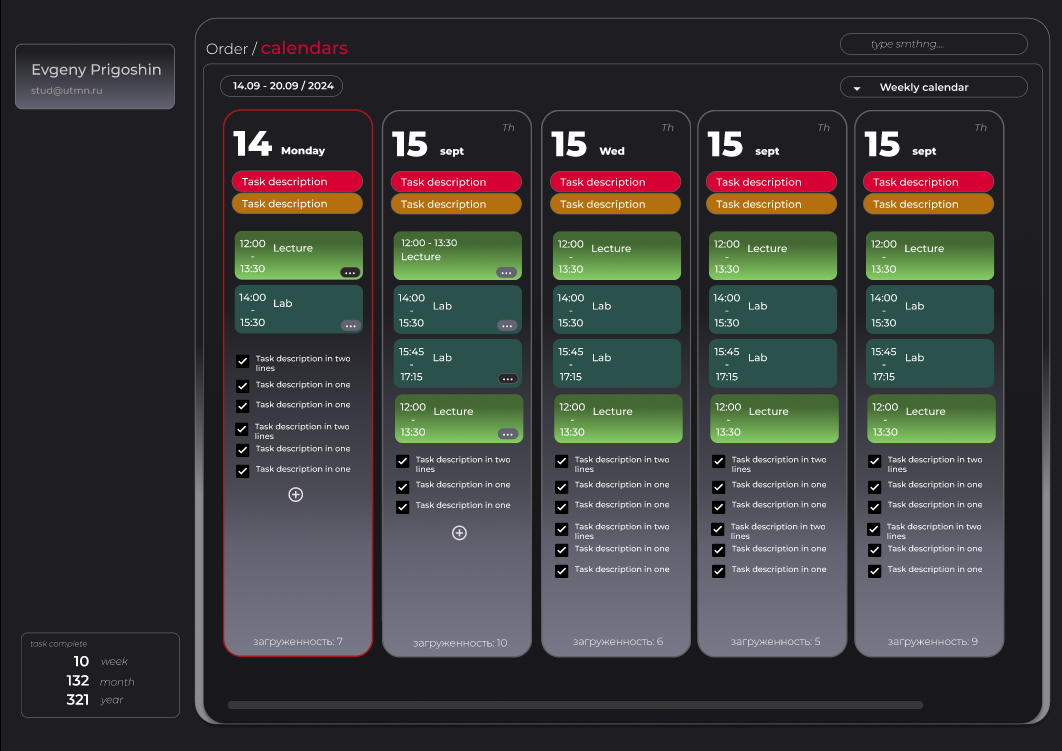
****

Рисунок 2 – макет календаря на неделю

А также формы для создания задач и событий (рис. 3, 4).

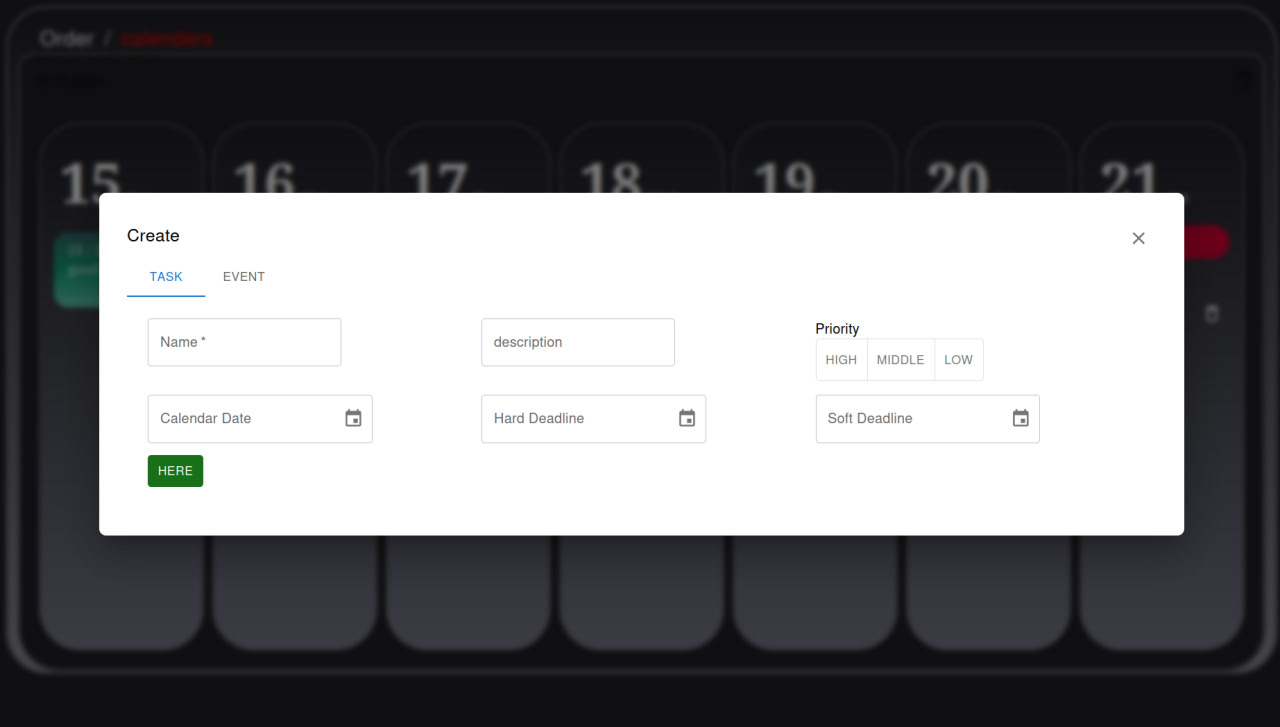


Рисунок 3 – форма для создания задачи

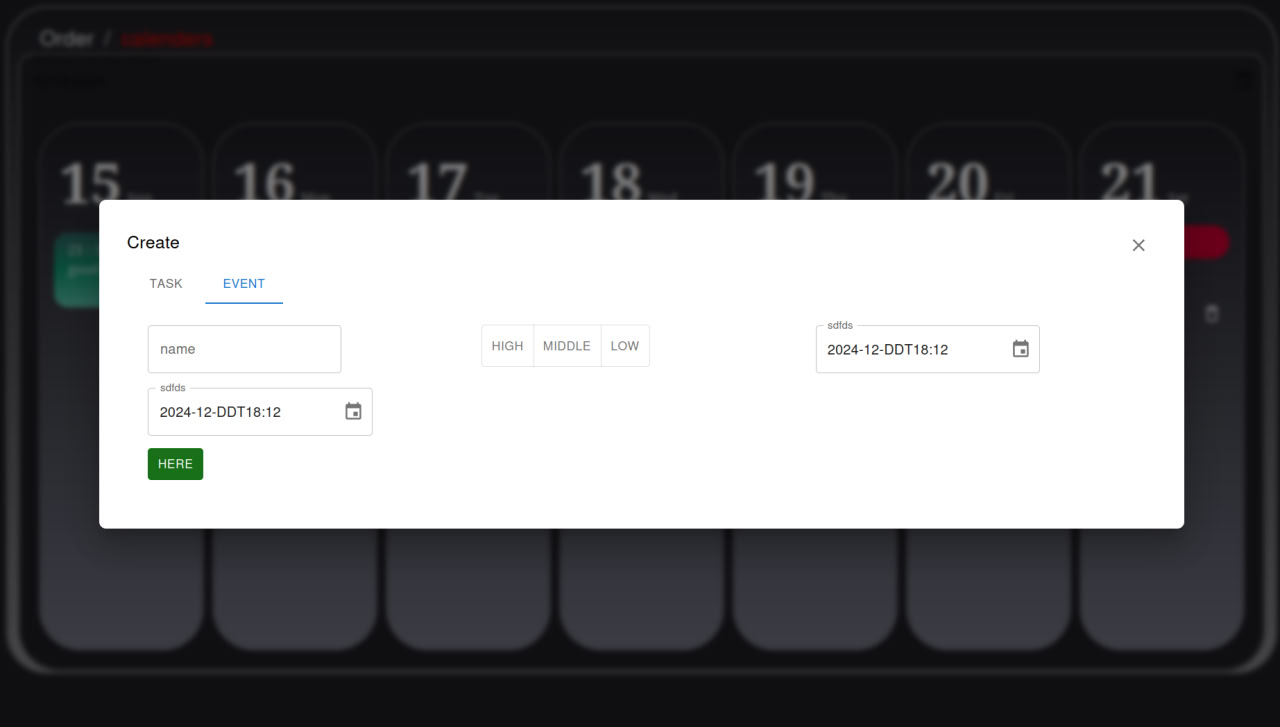


Рисунок 4 – форма для создания события

Основные функции, необходимые для реализации задуманного функционала, не требуют применения дополнительных библиотек.

Необходимо разделять пришедшую с бэкенда информацию по дням недели, а также выделять необходимую информацию (мягкие дедлайны и жёсткие), как представлено на рис. 5.

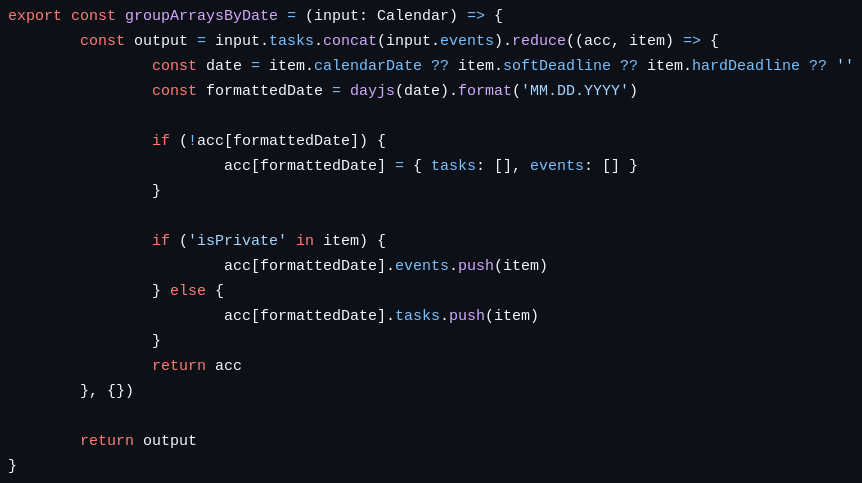


Рисунок 5 – обработка информации с бэкенда

Применяя возможности Javascript, было решено организовывать данные на фронтенде для отрисовки следующим образом: на выходе получается объект, ключи которого являются датой в календаре, а значение – объект, содержащий два ключа tasks и events, значениями которых являются массивы. Благодаря данной организации поступившей информации с ней облегчается взаимодействие.

Следующий метод: выделение задач с определенными дедлайнами; он необходим только для правильного отражения информации в календаре. Обе функции находятся на уровне widgets, так как требуются только для элементов управления (рис. 6).

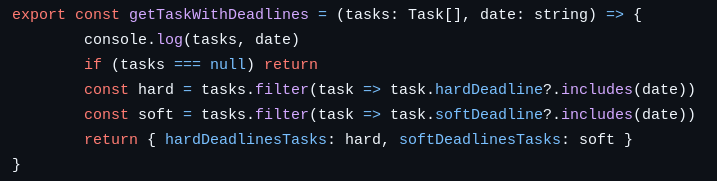


Рисунок 6 – метод выделения задач с определенными дедлайнами

Была реализована функция проверки валидности токенов на фронтенде (рис. 7). Эта функция нужна для защиты путей в приложении, чтобы не зарегистрированные пользователи или пользователи с истекшим токеном не могли получить доступ ко всем возможностям и всем страницам.



Рисунок 7 – функция проверки валидности токенов

Был реализован генератор форм, так как большинство форм используют один и тот же функционал, но различаются только полями и функциями отправки значений на сервер.

Генератор форм на вход принимает файл конфигурации, который выглядит подобным образом: это массив объектов, где каждый объект имеет обязательные ключи name (под каким именем необходимо зарегистрировать компонент, чтобы потом получить введенные пользователем данные) и ключ component, значением которого является React компонент, который будет отрисован и зарегистрирован на форме. Было предусмотрено, чтобы для каждого компонента можно было указать необходимые параметры (обязательность заполнения – required, максимальная дата – maxDate, многострочность – multiline, и т.д.).

Чтобы отобразить форму, необходимо написать файл конфигурации и предоставить функцию отправки. Для примера: AuthConfig (рис. 8). Это компоненты формы, необходимые для регистрации и входа.



Рисунок 8 - AuthConfig

Это значительно облегчает поддержку приложения в будущем и написание функционала сейчас, так как исключает излишнее дублирование кода. Файлы конфигурации представлены на уровне shared (данный уровень предполагает, что код может быть использован на всех уровнях ниже себя и в нескольких местах), функции отправки могут располагаться на уровнях widgets, pages, т.к. они реализуют логику, необходимую только на данных слоях и сегментах.

Для отправки запросов на сервер на слое shared есть сегмент API. API подразделяется ещё на несколько уровней, файлы, содержащие конкретные запросы по каждому конкретному эндпоинту на бэкенде. Для примера, представлен запрос на регистрацию пользователя и получение токена для него на рис. 9.

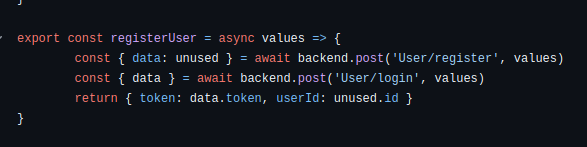


Рисунок 9 – запрос на регистрацию пользователя и получение токена

## 7.2 Разработка бэкенда web-приложения

При разработке базы данных использовался подход code-first, предполагающий создание классов сущностей в приложении ASP.Net Core, а затем перенос структуры таблиц и связей в СУБД PostgreSQL с использованием механизма миграций, предоставляемого библиотекой Entity Framework Core.

Со стороны бэкенда была реализована логика работы с основными сущностями (CRUD-операции), а также основные эндпоинты для обмена данными с фронтендом в формате запросов. Пример представлен на рис. 10. Для автоматизации обновления полей сущностей применялся маппинг. Зачастую нет необходимости обновлять все поля сущности, а при использовании маппинга есть возможность изменять поля точечно: если поле в полученном объекте пусто (имеет значение null), то значение объекта остается без изменений.

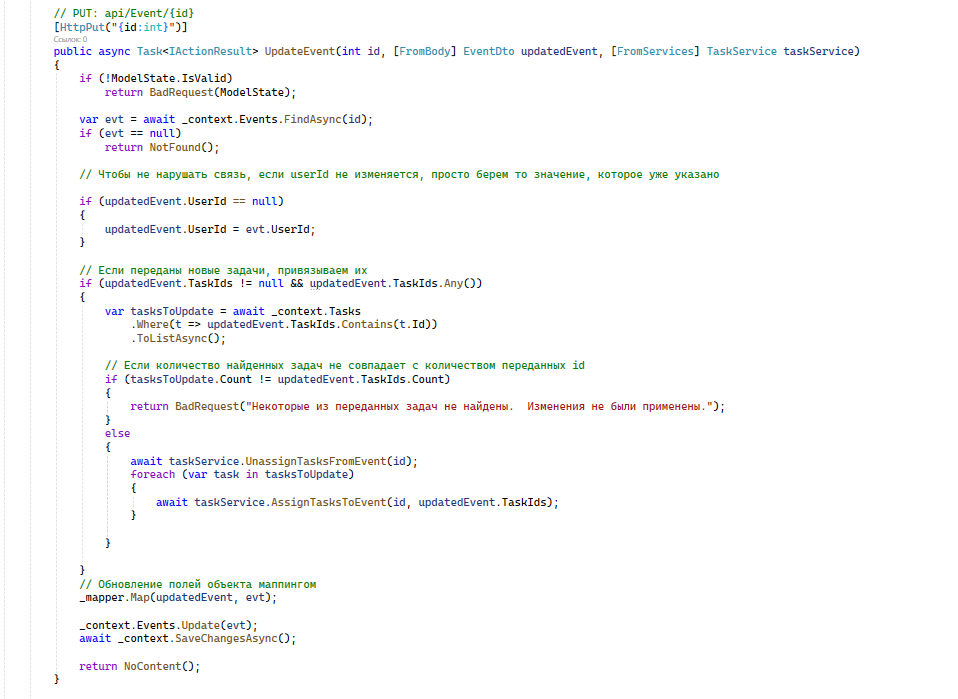


Рисунок 10 – метод обновления полей сущности события с использованием маппинга

Некоторые отдельные функции были вынесены в сервисный слой приложения. Например, было необходимо реализовать привязку задач к событиям и проектам: задача может быть привязана к одному событию/проекту, а событие/проект может иметь несколько задач. Функции для привязки и отвязки задач были вынесена в TaskService, пример кода представлен на рис. 11.

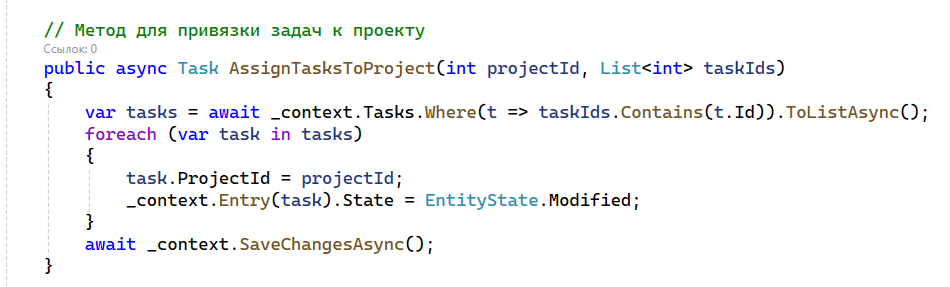


Рисунок 11 – метод для автоматического изменения поля projectId задачи в случае, если был обновлен список привязанных к проекту задач

Была реализована регистрация и авторизация пользователей в приложении с использованием JWT-токенов (рис. 12).



Рисунок 12 – методы регистрации и авторизации пользователей

Был частично реализован функционал получения информации об учебных встречах из modeus (рис. 13), однако не был внедрен в прототип на данном этапе разработки.



Рисунок 13 – метод получения расписания из modeus по токену

Получаемые с использованием токена пользователя данные преобразуются к структуре сущности события и автоматически вносятся в базу данных с привязкой к пользователю приложения по его id (рис. 14).

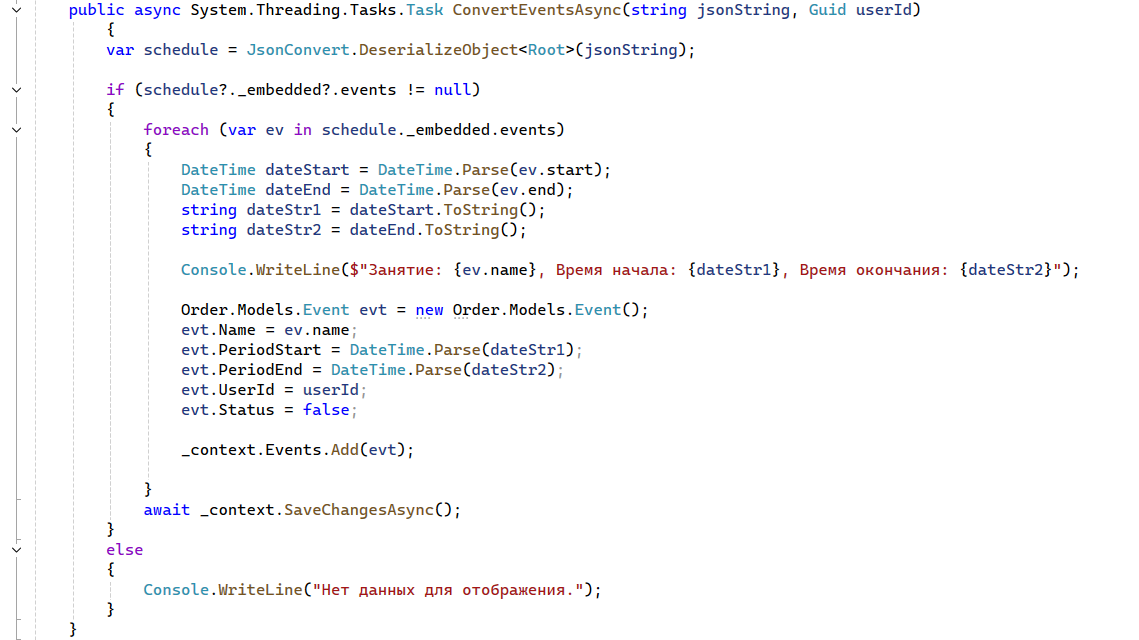


Рисунок 14 – метод для преобразования данных к формату сущности события, внесение информации в базу данных

Был реализован механизм создания публичной ссылки (рис. 15). Для этого указывается начало и конец периода, из которых будут браться события и задачи пользователя.

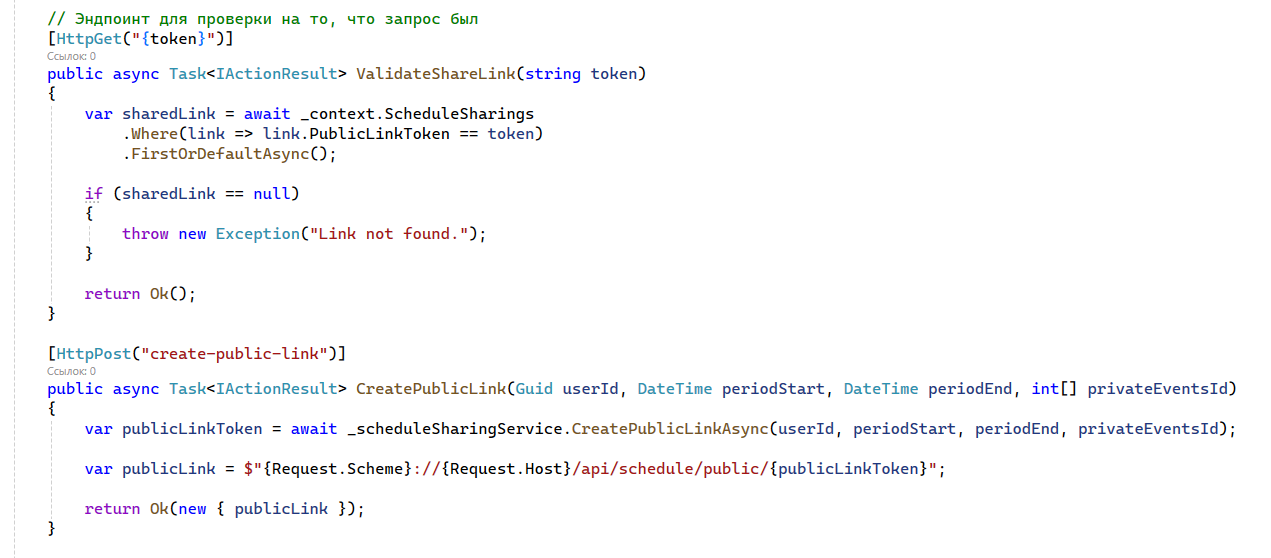


Рисунок 15 – методы получения публичной ссылки

Далее формируется и сохраняется в базу уникальный токен, ему в соответствие ставится id пользователя и диапазон дат. По этому токену в последствии получаются данные пользователя (рис. 16).

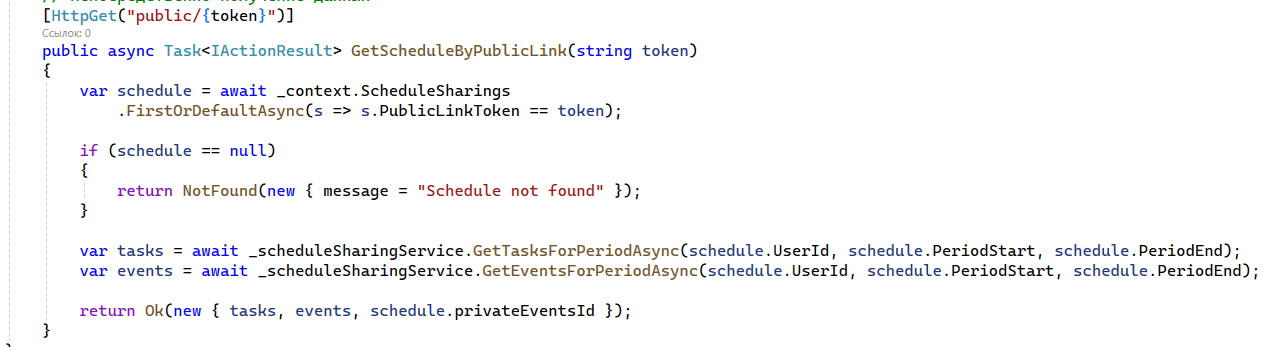


Рисунок 16 – метод получения данных по токену

# Глава 8. Тестирование проекта

Цель тестирования: подтвердить корректность работы всех функций приложения, оценить его производительность и удобство использования, а также выявить возможные ошибки или недочеты.

Виды тестирования:

* Интеграционное тестирование. Цель: убедиться, что взаимодействие между компонентами системы происходит без ошибок.
* Функциональное тестирование. Цель: проверить соответствие функционала приложения требованиям.

Основные кейсы:

* Создание, редактирование и удаление задач.
* Настройка напоминаний и проверка их корректной работы.
* Проверка, что при загрузке расписания приложение корректно его отображает и встраивает задачи.
* Пользовательское тестирование (Usability testing). Цель: оценить удобство интерфейса реальными пользователями.

Процедура тестирования

1. Первый этап: тестовые данные.

Предполагается проверка работы функционала на созданном наборе данных, включающем:

* Тестовые учетные записи пользователей.
* Наборы задач, событий и проектов с различными характеристиками.
* Разные контексты и приоритеты задач.

1. Второй этап: привлечение пользователей.

Участники: студенты, которые представляют целевую аудиторию.

* Выполнение участниками набора заранее определенных задач.
* Сбор обратной связи через анкеты или интервью.

Параметры замера результатов

* Удобство интерфейса:
  + Время, затраченное на выполнение типовых задач.
  + Сложности, с которыми сталкиваются пользователи.
  + Частота обращения к обучающим материалам.
* Качество работы приложения:
  + Количество ошибок в работе приложения.
  + Корректность расчетов временных интервалов для задач.
  + Скорость отклика интерфейса.

# Заключение

В ходе работы над проектом был разработан прототип веб-приложения планирования задач, ориентированного на студентов.

Была разработана и реализована структура базы данных в соответствии с требованиями к приложению.

На стороне бэкенда была реализована логика работы с основными сущностями (CRUD-операции), а также основные эндпоинты для обмена данными с фронтендом в формате запросов. Был реализован функционал создания публичной ссылки, а также получения информации о событиях и задачах для отображения по ней.

На стороне фронтенда был разработан интерфейс недельного календаря, а также формы для создания задач и событий, в соответствии со структурой базы данных и принципами цветового кодирования.

Был частично реализован функционал получения информации об учебных встречах из modeus, однако не был внедрен в прототип на данном этапе разработки.

По итогу работы готовность системы оценивается в 60% в соответствии с установленными задачами. Необходимо доработать интеграцию с modeus, добавить функционал уведомлений, а также интерфейс для Inbox, создания публичной ссылки.

# Список литературы

1. Кульгейко С. А. Анализ уровня прокрастинации у студентов [Электронный ресурс] // Репозиторий БНТУ. — 2024. — URL: https://rep.bntu.by/handle/data/126870 (дата обращения: 24.11.2024).
2. Диагностика взаимосвязи учебной мотивации и академической прокрастинации студентов технического университета. [Электронный ресурс] // Международный научно-популярный журнал “Мастерство online”. – <https://ripo.by/index.php?id=7824> (дата обращения: 24.11.2024).
3. Психологические аспекты обучения: мотивация, управление стрессом, развитие навыков критического мышления и самоорганизации [Электронный ресурс] // CyberLeninka. — URL: https://cyberleninka.ru/article/n/psihologicheskie-aspekty-obucheniya-motivatsiya-upravlenie-stressom-razvitie-navykov-kriticheskogo-myshleniya-i-samoorganizatsii/viewer (дата обращения: 23.11.2024).
4. Getting Things Done: The Science behind Stress-Free Productivity. Francis H. Clément V. — 2008. — 41. — №6. — С. 585-605. — URL: https://www.sciencedirect.com (дата обращения: 24.11.2024).
5. Emerald Insight: GTD и современные условия. Malin, V., Murphy, J. and Siltaoja, M. — 2013. — Dialogues in Critical Management Studies. — №2. — URL: https://www.emerald.com (дата обращения: 24.11.2024).
6. FacileThings: 10 преимуществ GTD. Sáez F. 10 Benefits of Using GTD // FacileThings [Электронный ресурс]. — URL: https://facilethings.com (дата обращения: 23.11.2024).
7. Pérez, C., Rodríguez, D. Designing APIs with Swagger and OpenAPI. — Springer, 2018. — 250 с.
8. Smith, A., Johnson, T. The role of color in memory and attention // Visual Cognition. — 2012. — Vol. 20, No. 5. — P. 543–561. — DOI: https://link.springer.com/. — URL: https://link.springer.com/article/10.3758/s13414-023-02714-4 (дата обращения: 24.11.2024).
9. Anson C., Josée R., Patrick C. Attention capture by salient color cues // Vision Research. — 1997. — Vol. 37, No. 3. — P. 214–232. — DOI: https://link.springer.com/. — URL: https://link.springer.com/article/10.3758/s13414-020-02054-7 (дата обращения: 24.11.2024).