**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по преддипломной практике (по профилю специальности)**

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/4 группы

Толстых София Александровна

(Фамилия, имя, отчество)

Место прохождения практики: ООО «СВИТЛАЙФ МАРКЕТ», Санкт-Петербург, наб. Обводного канала, д. 93а литер а, помещ. 5н офис 1449

(наименование и адрес организации)

Период прохождения практики

с «22» апреля 2024 г. по «18» мая 2024 г.

Руководитель практики

от учебной организации Иванова Д.В \_\_

(подпись) (расшифровка подписи)

Руководитель с места

прохождения практики Кудыма Т.А

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

М.П.

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на преддипломную практику (по профилю специальности)**

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/4 группы

Толстых София Александровна

(Фамилия, имя, отчество)

Место прохождения практики: ООО «СВИТЛАЙФ МАРКЕТ», Санкт-Петербург, наб. Обводного канала, д. 93а литер а, помещ. 5н офис 1449

(наименование и адрес организации)

Период прохождения практики

с «22» апреля 2024 г. по «18» мая 2024 г.

**Виды работ, обязательные для выполнения**:

1. Ознакомление со структурой и характером деятельности предприятия.
2. Сбор материалов для составления технического задания по теме дипломного проекта.
3. Разработка программного обеспечения на основе технического задания дипломного проекта.
4. Проведение испытаний, отладка и внедрение программного продукта на предприятии.
5. Расчет показателей экономической эффективности программного продукта.
6. Оформление отчета о прохождении производственной практики (преддипломной).

Задание выдал «22» апреля 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Иванова Д.В.

(подпись)  (Ф.И.О.)

Задание получил «22» апреля 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Толстых С.А (подпись) (Ф.И.О.)

**СОГЛАСОВАНО**

Председатель ПЦК «ИС»

\_\_\_\_\_\_\_\_\_\_/В.А. Андреев

«22» апреля 2024 г.

**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения преддипломной практики**

**(по профилю специальности)**

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/4 группы

Толстых София Александровна

(Фамилия, имя, отчество)

Место прохождения практики: ООО «СВИТЛАЙФ МАРКЕТ», Санкт-Петербург, наб. Обводного канала, д. 93а литер а, помещ. 5н офис 1449

(наименование и адрес организации)

Период прохождения практики

с «22» апреля 2024 г. по «18» мая 2024 г.

Руководитель практики

от учебной организации Иванова Д.В.

(подпись) (расшифровка подписи)

Руководитель с места

прохождения практики Кудыма Т.А

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

М.П.

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| Дата | Виды выполненных работ и заданий по программе практики | Подпись руководителя практики |
| **1** | **2** | **3** |
| 22.04.24 | Планирование и анализ требований: уточнение функциональных и нефункциональных требований, определение основных сценариев использования.  Исследование технологий: изучение документации по Nest.js, Prisma и другим технологиям. |  |
| 25.04.24 |
| 26.04.24 | Настройка окружения разработки: установка необходимых инструментов разработки (редактор кода, Node.js, Yarn), настройка Git репозитория для контроля версий.  Создание основных компонентов проекта: инициализация Nest.js проекта, настройка структуры папок и файлов. |  |
| 28.04.24 |
| 29.04.24 | Проектирование информационной модели, разработка ER-диаграммы и диаграмм последовательностей.  Обработка HTTP-запросов, разработка маршрутов, создание нескольких контроллеров. |  |
| 01.05.24 |
| 02.05.24 | Работа со статическими файлами, файловой системой, доработка базы данных.  Реализация серверной логики: создание API эндпоинтов с использованием Nest.js, интеграция с Prisma. |  |
| 04.05.24 |
| 05.05.24 | Создание регистрации и авторизации с помощью JSON web token.  Тестирование и отладка: проверка работоспособности приложения, исправление ошибок. |  |
| 07.05.24 |
| 08.05.24 | Исправление ошибок.  Подготовка к релизу: финальное тестирование и отладка, документирование кода и функционала. |  |
| 10.05.24 |
| 11.05.24 | Создание основных компонентов React приложения.  Разработка пользовательского интерфейса: создание макетов, разработка компонентов. |  |
| 13.05.24 |
| 14.05.24 | Интеграция с серверной частью, настройка взаимодействия.  Реализация функционала редактирования состояния объектов. |  |
| 16.05.24 |
| 17.05.24 | Тестирование и отладка клиентской части.  Написание и защита отчета |  |
| 18.05.24 |

СОДЕРЖАНИЕ

[Введение 6](#_Toc166355582)

[1.1 Анализ предметной области 7](#_Toc166355583)

[1.2 Постановка задачи 7](#_Toc166355584)

[1.3 Анализ требований 8](#_Toc166355585)

[1.4 Анализ рынка существующих решений 11](#_Toc166355586)

[1.5 Анализ методов решения 13](#_Toc166355587)

[1.6 Обзор средств программирования 17](#_Toc166355588)

[1.7 Вывод по общей части 21](#_Toc166355589)

[2 Специальная часть 23](#_Toc166355590)

[2.1 Описание структуры приложения 23](#_Toc166355591)

[2.2 Проектирование программы 23](#_Toc166355592)

[2.3 Разработка БД 26](#_Toc166355593)

[2.4 Разработка пользовательского интерфейса 27](#_Toc166355594)

[2.5 Разработка программы 33](#_Toc166355595)

[2.6 Тестирование программного продукта 35](#_Toc166355596)

[2.7 Вывод по специальной части 35](#_Toc166355597)

[Заключение 36](#_Toc166355598)

[Список использованных источников 37](#_Toc166355599)

[Приложение А (справочное) Результаты анкетирования 38](#_Toc166355600)

[Приложение Б (обязательное) Программный код БД 40](#_Toc166355601)

[Приложение В (обязательное) Программный код приложения 43](#_Toc166355602)

[Приложение Г (обязательное) Тест кейсы 50](#_Toc166355603)

# ВВЕДЕНИЕ

С развитием информационных технологий программные решения становятся неотъемлемой частью многих видов бизнеса. Гостиничное дело не является исключением, поскольку даже в этой сфере есть множество процессов, которые можно автоматизировать и оптимизировать. Исходя из этой потребности, возникает необходимость в создании специализированных веб-приложений, направленных на улучшение управления и мониторинга объектов отдыха.

Одним из направлений бизнеса в сфере гостиничного дела являются базы отдыха, которые предоставляют не только места для проживания, но и дополнительную придомовую территорию и объекты на ней. В свою очередь расширение предоставляемых услуг повышает количество задействованного персонала, а расширение территории увеличивает сложность отслеживания состояния объектов.

Таким образом, целью работы выступает решение проблемы по ускорению процесса обмена информации между работниками, позволяющее в более короткие сроки получать информацию о актуальном состоянии объектов.

**1 Общая часть**

# 1.1 Анализ предметной области

Предприятие «СВИТЛАЙФ МАРКЕТ» было зарегистрировано 29 августа 2013 года. Отрасль данного предприятия – это оптовая торговля пищевыми продуктами, напитками и табачными изделиями. Основным видом деятельности предприятия по ОКВЭД является оптовая торговля шоколадом и сахаристыми кондитерскими изделиями. Данное предприятие занимает 675 место в России по выручке за 2022 год [1]. В данном предприятии насчитывается 20 человек. Отделами предприятия являются: бухгалтерия, маркетинговый отдел, отдел кадров, отдел снабжения, финансовый отдел и юридический отдел.

В связи с расширением деятельности, в сторону оказания услуг в сфере гостиничного сервиса, необходимо создание веб-приложения для мониторинга состояния объектов на базе отдыха.

Пользователями разрабатываемого приложения будут являться сотрудники баз отдыха.

# 1.2 Постановка задачи

Основной задачей данной работы является проектирование и разработка веб-приложения для мониторинга состояния объектов на базе отдыха. Разрабатываемое приложение должно предоставлять пользователям актуальную информацию о состоянии объектов, а также давать им возможность изменять данное состояние.

Для реализации данного проекта необходимо провести анализы требований, анализ и выбор средств разработки, анализ рынка конкурентов.

# 1.3 Анализ требований

**1.3.1 Анкетирование заинтересованных лиц**

Для разработки данного веб приложения по мониторингу состояния объектов на базе отдыха, были выявлены следующие заинтересованные лица:

* Администраторы
* Горничные
* Хаусмены

На основании выявленных заинтересованных лиц, было проведено анкетирование каждого заинтересованного лица. Результаты анкетирования представлены в приложении А.

**1.3.2 Пользовательские истории**

При разработке веб приложения были сформированы пользовательские истории:

* Я как администратор хочу иметь возможность изменять статусы объектов и задач в реальном времени, чтобы оперативно управлять назначениями и следить за их выполнением.
* Я как администратор стремлюсь наблюдать за изменениями статусов объектов в приложении, чтобы анализировать загруженность объектов и эффективность работы персонала.
* Я как администратор желаю получать уведомления о сменах статусов задач и объектов, чтобы мгновенно реагировать на возникающие проблемы и потребности клиентов.
* Я как хаусмен хочу видеть обновления статусов объектов в приложении, чтобы понимать текущие приоритеты и срочность задач.
* Я как хаусмен стремлюсь к тому, чтобы иметь возможность быстро изменять статусы выполнения своих заданий, обеспечивая таким образом актуальное состояние информации для коллег и администрации.
* Я как хаусмен желаю получать уведомления о изменении статусов объектов, на которые распространяется моя ответственность, для оперативного реагирования на требования по обслуживанию и ремонту.
* Я как горничная хочу иметь возможность изменять статусы номеров после их уборки, чтобы администрация и гости были в курсе готовности номеров к заселению.
* Я как горничная стремлюсь получать информацию об изменениях статусов номеров, на которые распространяется моя ответственность, чтобы эффективно планировать свой рабочий день.
* Я как горничная желаю видеть список номеров с актуальными статусами на уборку, чтобы оптимизировать порядок своей работы и убедиться в её качестве перед сдачей номеров.

**1.3.3 Функциональные и нефункциональные требования**

На основании пользовательских историй и проведенного анкетирования, были сформированы функциональные и нефункциональные требования:

* приложение должно позволять пользователям изменять статусы объектов (например, номеров, зон обслуживания) в реальном времени (например, "готово к заселению", "требуется уборка", "требуется ремонт");
* приложение должно предоставлять возможность регистрации и авторизации различных типов пользователей (администраторы, хаусмены, горничные) с разграничением уровней доступа к функциям приложения;
* приложение должно автоматически отправлять уведомления пользователям о смене статусов объектов, важных задачах или сообщениях от администрации;
* приложение должно иметь наглядный интерфейс для быстрого просмотра текущих статусов всех объектов и задач в системе;
* приложение должно иметь возможность отправки сообщений в рабочие чаты через функционал приложения;
* приложение должно быть адаптировано для использования на мобильных устройствах, чтобы обеспечить доступ к функциям в любое время и в любом месте.

Не функциональные требования:

* система должна иметь интуитивно понятный и легко используемый интерфейс;
* данные пользователей, которые хранятся в базе данных, должны быть зашифрованы;
* время загрузки формы, не должно превышать 10 секунд.

**1.3.4 Построение диаграммы вариантов использования**

В процессе проектирования веб-приложения были учтены требования к функциональности программы, что привело к созданию диаграммы вариантов использования. Эта диаграмма отображает, какие функции доступны различным категориям пользователей [2]. Иллюстрация данной диаграммы представлена на рисунке 1.



Рисунок 1 – Диаграмма вариантов использования

# 1.4 Анализ рынка существующих решений

**1.4.1 TravelLine**

TravelLine – российская компания разрабатывающая единую функциональную платформу c 2008 года для автоматизации процессов гостиничного бизнеса. В сферу работы платформы входит практически все что связано с онлайн-бронированием и управлением номерным фондом в отелях.

Компания TravelLine работает с более чем 12 000 клиентами по всей России и предоставляет свои услуги для бизнеса, связанного с отелями, апартаментами, базами отдыха, санаториями.

Данное приложение отлично зарекомендовало себя на рынке, оно имеет удобный интерфейс и хорошую техническую поддержку.

Приложение TravelLine также имеет возможность отслеживания и изменения состояния жилищных объектов, но не имеет возможности управления дополнительными объектами, а также само приложение предоставляет свой функционал только для администраторов гостиничного сервиса.

Таким образом программа TravelLine решает поставленную задачу только частично, так как главной функцией этой программы является работа с бронированием, а все остальные ее функции являются дополнительными и мало настраиваемыми. Приложение TravelLine представлено на рисунке 2.

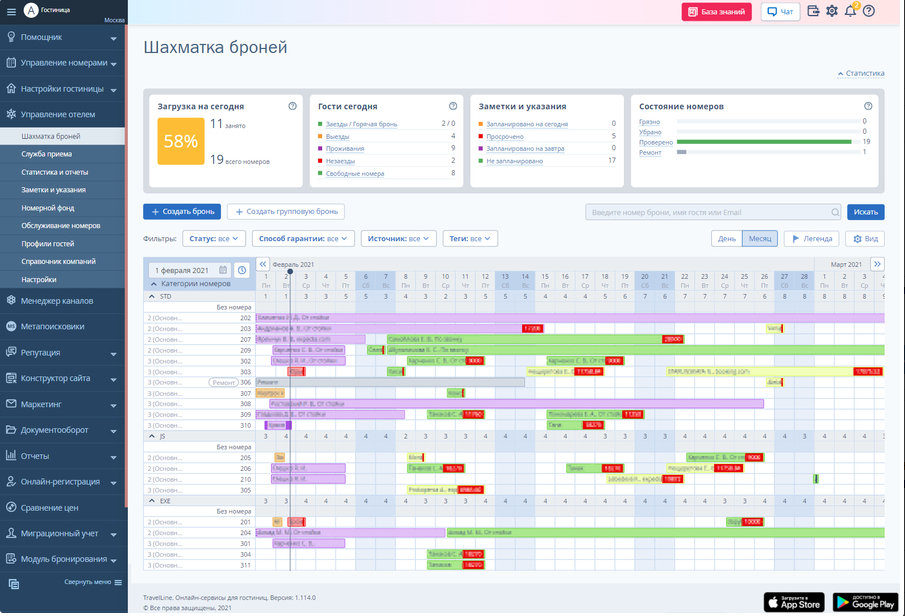


Рисунок 2 – Приложение TravelLine

**1.4.2 Bnovo**

Bnovo – ведущий российский разработчик IT-решений для гостиничного бизнеса. Компания работает на рынке с 2012 года и создает современные решения для эффективного управления и увеличения дохода объектов размещения любого типа: крупных, средних и малых отелей, апартаментов, баз отдыха, пансионатов, санаториев и гостевых домов.

Помимо привычного браузерного управления, система Bnovo имеет адаптивную мобильную версию, а также мобильное приложение для Android и IOS.

Данное приложение также предоставляет возможность онлайн бронирования, но не решает поставленную задачу.

Приложение Bnovo представлено на рисунке 3.

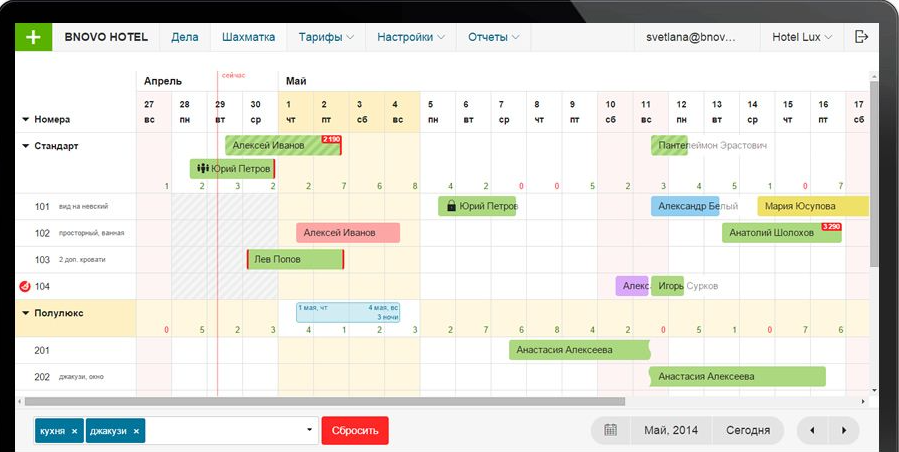


Рисунок 3 – Приложение Bnovo

**1.4.3** **Вывод анализа рынка существующих решений**

После анализа различных существующих решений, можно сделать вывод, что поставленную цель либо выполняют частично, либо вовсе не выполняют. Большинство решений перекладывает ответственность за отслеживанием состояния объектов базы отдыха на администратора. В связи с этим разработка приложения является целесообразной.

# 1.5 Анализ методов решения

**1.5.1 Ручной метод решения**

Для решения задачи мониторинга состояний объектов на базе отдыха ручным методом можно использовать оповещения в рабочих чатах. В этом случае администраторы объектов могут регулярно обновлять информацию о состоянии объектов и услугах, отправляя сообщения в специальные чаты, доступные горничным и хаусменам.

Преимущества:

1. Простота в использовании. Для обновления информации администраторам достаточно отправить сообщение в чат, что делает процесс управления состоянием объектов отдыха быстрым и удобным.
2. Мгновенное уведомление. Горничные и хаусмены получают оповещения в реальном времени о любых изменениях состояния объектов, что позволяет им оперативно реагировать на требующиеся действия.

Недостатки:

1. Ограниченная структурированность. Использование чатов может привести к разрозненности информации и затруднить ее структурирование, особенно при большом объеме объектов.
2. Ограниченные возможности управления. Чаты не предоставляют возможности удобного управления информацией, такой как фильтрация, сортировка или поиск, что может затруднить работу с данными.
3. Недостаточная безопасность. Использование чатов может создать уязвимости в защите данных, особенно если информация о состоянии объектов содержит конфиденциальные данные.

**1.5.2 Автоматический метод решения**

Для автоматического переноса данных о бронировании и купленных услугах из приложения по бронированию в специализированное приложение о состоянии объектов на базе отдыха, можно разработать интеграцию между этими двумя приложениями. Эта интеграция позволит передавать данные о бронированиях и услугах автоматически, без необходимости ручного вмешательства.

Процесс автоматического переноса данных может выглядеть следующим образом:

1. После того, как пользователь совершает бронирование или покупку услуги в приложении по бронированию, данные о бронировании и услугах передаются в специальный интерфейс или API.
2. С помощью этого интерфейса данные автоматически передаются в приложение о состоянии объектов на базе отдыха.
3. В приложении о состоянии объектов данные о бронированиях и купленных услугах обрабатываются и автоматически вносятся в соответствующие разделы или базы данных.

Преимущества автоматического метода:

1. Эффективность. Автоматический перенос данных снижает вероятность ошибок и ускоряет процесс обновления информации о бронированиях и услугах.
2. Точность. Передача данных между приложениями происходит автоматически, что уменьшает риск возникновения неточностей или пропусков.
3. Временные сроки. Автоматический метод позволяет мгновенно обновлять данные в приложении о состоянии объектов, что обеспечивает оперативную информацию о состоянии и доступности услуг.
4. Удобство для пользователей. Пользователям не нужно вручную вносить информацию о бронированиях и купленных услугах в различные системы, что упрощает процесс и снижает вероятность ошибок.

Недостатки автоматического метода:

1. Технические сложности: Разработка и настройка интеграции между приложениями требует определенного уровня технической экспертизы.
2. Необходимость поддержки: Автоматический метод требует постоянной поддержки и обновлений, чтобы обеспечить стабильную и надежную работу системы.
3. Риски безопасности: Передача данных между приложениями может повлечь за собой риски безопасности, поэтому важно обеспечить защиту данных во время передачи.

**1.5.3 Автоматизированный метод решения**

Для решения задачи мониторинга состояний объектов на базе отдыха с использованием автоматизированного метода можно разработать веб-приложение, в котором администраторы объектов могут ручным образом вносить изменения, а горничные и хаусмены имеют возможность просматривать и обновлять эти статусы в реальном времени.

Процесс работы с таким приложением может быть следующим.

1. Администраторы объектов, имеющие доступ к панели управления приложением, могут вносить информацию о текущем состоянии объектов отдыха, такую как уровень уборки, наличие необходимости в растопке и т.д. Эта информация отображается в приложении и доступна для просмотра всем пользователям.
2. Горничные и хаусмены, имеющие доступ к приложению через свои мобильные устройства или компьютеры, могут просматривать текущие статусы объектов отдыха и обновлять их в соответствии с актуальной ситуацией. Например, горничная может отметить, что номер убран, а хаусмен – что выполнена растопка.
3. После обновления статусов администраторы могут видеть изменения в реальном времени и принимать необходимые меры, если требуется дополнительное вмешательство.

Преимущества автоматизированного метода:

1. Гибкость. Администраторы могут легко вносить изменения в состояние объектов отдыха, а персонал может мгновенно реагировать на изменения и обновлять статусы.
2. Эффективность. Обновление статусов в реальном времени позволяет поддерживать высокий уровень обслуживания и оперативно реагировать на требования клиентов.
3. Прозрачность. Все изменения отображаются в приложении, что обеспечивает прозрачность процесса для всех участников.

Недостатки автоматизированного метода:

1. Возможность ошибок. Ввиду ручного внесения данных администраторами, существует риск возникновения ошибок или неправильной интерпретации информации.
2. Необходимость обучения персонала. Персоналу может потребоваться обучение по использованию приложения и правильному обновлению статусов.
3. Ограниченные возможности отслеживания. В зависимости от функциональности приложения, может быть ограничен доступ к истории изменений статусов и отслеживание трендов или паттернов.

**1.5.4 Выбор метода решения**

Исходя из описания и требований к проекту по мониторингу состояния объектов на базе отдыха, автоматизированный метод решения является оптимальным выбором. Данный подход позволит обеспечить непрерывный и актуальный мониторинг состояния объектов, а также эффективное управление информацией о них.

Таким образом, применение автоматизированного метода в проекте по мониторингу состояния объектов на базе отдыха обеспечит высокую эффективность, точность и удобство использования системы, что сделает её незаменимым инструментом для управления объектами отдыха.

# 1.6 Обзор средств программирования

**1.6.1 Обзор и выбор среды программирования**

При сравнении сред разработки Visual Studio Code (VS Code) и Visual Studio, предпочтение отдается Visual Studio Code. VS Code представляет собой бесплатную и открытую среду разработки, разработанную компанией Microsoft, которая предназначена для создания различных типов приложений, включая веб-приложения, с использованием различных языков программирования.

Преимущества VS Code включают быструю загрузку и отзывчивость, поддержку различных операционных систем (Windows, macOS, Linux), наличие обширного сообщества пользователей и множества расширений для индивидуальной настройки среды разработки. Интеграция с Git упрощает контроль версий, а удобный интерфейс и широкий набор инструментов обеспечивают эффективную работу с кодом [3].

Несмотря на множество преимуществ, у VS Code есть и некоторые недостатки. Например, она может быть менее функциональной по сравнению с Visual Studio, особенно при работе над крупными проектами. Также некоторые типы проектов и языки программирования могут потребовать дополнительной настройки среды, а на крупных проектах с большим объемом файлов может возникнуть снижение производительности.

С другой стороны, Visual Studio предлагает мощные инструменты разработки, интеграцию с экосистемой Microsoft и встроенные инструменты отладки и профилирования. Однако она требует больше ресурсов компьютера и доступна только для операционной системы Windows, что может ограничить выбор разработчиков, предпочитающих другие платформы [4].

Исходя из вышеупомянутых факторов, выбор делается в пользу Visual Studio Code, учитывая её универсальность, легкость в использовании и возможности настройки.

**1.6.2 Обзор и выбор языка программирования**

**1.6.2.1 Python**

Python — это высокоуровневый интерпретируемый язык программирования, который обладает простым синтаксисом и широким спектром применения, включая веб-разработку.

Плюсы:

1. Простота и читаемость кода.
2. Большое количество готовых библиотек для работы с данными.
3. Мощные фреймворки для веб-разработки, такие как Django и Flask.

Минусы:

1. Менее подходит для создания высокопроизводительных приложений.
2. Меньшее распространение в сравнении с JavaScript.

**1.6.2.2 JavaScript**

JavaScript — это динамический язык программирования, широко используемый для создания интерактивных веб-приложений.

Плюсы:

1. Широкое распространение и поддержка.
2. Возможность использования на клиентской и серверной сторонах.
3. Богатый выбор библиотек и фреймворков.

Минусы:

1. Отсутствие строгой типизации.
2. Некоторые асинхронные операции могут привести к сложностям в управлении потоком выполнения.

**1.6.2.3 TypeScript**

TypeScript — это язык программирования, являющийся надстройкой над JavaScript, добавляющий статическую типизацию и другие возможности, упрощающие разработку крупных приложений.

Плюсы:

1. Статическая типизация обеспечивает более безопасное и надежное программирование.
2. Интеграция с современными фреймворками и библиотеками.
3. Улучшенная поддержка IDE и обширное сообщество.

Минусы:

1. Небольшой порог входа из-за необходимости изучения статической типизации.
2. Дополнительное время на разработку из-за необходимости написания типов.

**1.6.2.4 Выбор языка программирования**

При выборе языка программирования для разработки веб-приложения по мониторингу состояний объектов на базе отдыха был проведён анализ различных языков программирования, включая JavaScript, Python и TypeScript.

JavaScript был рассмотрен в первую очередь из-за его широкого использования и поддержки в веб-разработке. Однако, его динамическая типизация может привести к некоторым сложностям в управлении кодом на больших проектах.

Python, с другой стороны, предлагает простой и читаемый синтаксис, что облегчает разработку. Однако, в контексте веб-разработки, он может быть менее предпочтителен из-за своей меньшей распространённости и ограничений в производительности.

В результате анализа был выбран TypeScript. TypeScript представляет собой надстройку над JavaScript, добавляющую статическую типизацию и другие возможности, такие как улучшенная поддержка IDE и управление большими проектами. Эти возможности делают TypeScript оптимальным выбором для создания надёжного и масштабируемого веб-приложения для мониторинга состояний объектов на базе отдыха [5].

**1.6.3 Выбор технологий**

При выборе технологий для разработки веб-приложения по мониторингу состояний объектов на базе отдыха был выбран популярный набор технологий, состоящий из Nest.js для разработки серверной части приложения и React для создания клиентской части.

Nest.js был выбран из-за своей модульной структуры, интеграции с TypeScript и возможности использования паттерна MVC (Model-View-Controller). Этот фреймворк обеспечивает эффективную разработку масштабируемых и поддерживаемых серверных приложений [6].

React был выбран для разработки клиентской части приложения из-за его гибкости, быстродействия и активного сообщества разработчиков. React позволяет создавать интерактивные пользовательские интерфейсы с использованием компонентного подхода, что делает разработку более эффективной и удобной.

Была выбрана база данных PostgreSQL. PostgreSQL была выбрана из-за своей надёжности, масштабируемости и широкого функционального набора. Она предоставляет мощные средства для работы с данными, обеспечивает поддержку транзакций, а также имеет расширенные возможности для выполнения запросов и администрирования базы данных. Использование PostgreSQL в сочетании с Nest.js позволит создать надёжную и эффективную серверную часть приложения, способную обрабатывать большие объемы данных и обеспечивать высокую производительность [7].

Таким образом, технологии Nest.js и React были выбраны в качестве оптимального решения для создания современного и функционального веб-приложения по мониторингу состояний объектов на базе отдыха.

# 1.7 Вывод по общей части

Анализ существующих решений играет ключевую роль в процессе выбора подходящих технологий разработки для проекта. При изучении различных альтернатив были выделены как плюсы, так и минусы каждого решения. Это позволяет осуществить взвешенное сравнение и принять обоснованное решение относительно того, какие технологии лучше всего соответствуют требованиям проекта.

На основе этого анализа были выбраны подходящие технологии разработки, включая язык программирования, фреймворки и библиотеки. Этот выбор основан на сопоставлении возможностей и особенностей каждой технологии с требованиями проекта, а также учете их надежности, распространенности и поддержки сообществом разработчиков.

Таким образом, результаты анализа существующих решений позволили принять обоснованное решение о выборе технологий разработки, которые наилучшим образом соответствуют поставленным задачам и требованиям проекта.

# 2 Специальная часть

# 2.1 Описание структуры приложения

Веб-приложение организовано вокруг трех основных компонентов:

* клиентская часть (Client), это интерфейс, с которым взаимодействуют пользователи. Здесь располагается графическое представление сервиса, которое обеспечивает пользовательский опыт и интерактивность;
* серверная часть (Server), этот компонент отвечает за обработку запросов, поступающих от клиентской части приложения. Он управляет логикой приложения, включая валидацию данных, выполнение бизнес-логики и взаимодействие с базой данных;
* база данных (DataBase), здесь хранится информация, необходимая для функционирования приложения.

Такая структура позволяет эффективно организовать взаимодействие между компонентами и обеспечить функциональность и производительность веб-приложения. Структура приложения представлена на рисунке 4.

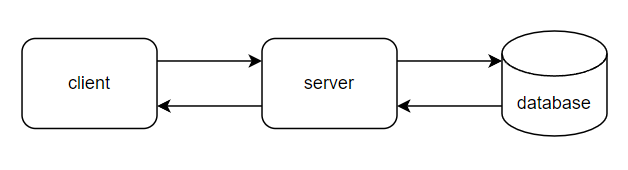


Рисунок 4 – Структура приложения

# 2.2 Проектирование программы

**2.2.1 Построение диаграммы последовательностей**

В процессе разработки веб-приложения для мониторинга состояния объектов на базе отдыха, была создана диаграмма последовательностей. Данная диаграмма отображает взаимодействие различных компонентов приложения в рамках различных сценариев использования [8]. Диаграмма последовательностей представлена на рисунке 5.



Рисунок 5 – Диаграмма последовательностей

**2.2.2 Построение диаграммы активности**

В процессе разработки веб-приложения для мониторинга состояния объектов на базе отдыха, была разработана диаграмма активностей. Этот инструмент визуализации позволяет подробно изучить последовательность действий и взаимодействие различных компонентов приложения в различных сценариях использования [9]. Диаграмма активностей представлена на рисунке 6.



Рисунок 6 – Диаграмма активности

**2.2.3 Проектирование базы данных**

В ходе разработки веб приложения, была спроектирована база данных. Для проектирования базы данных была разработана ER-диаграмма. Данная диаграмма представлена на рисунке 7.



Рисунок 7 – ER-Диаграмма

# 2.3 Разработка БД

Для разработки базы данных был выбран Prisma - современный ORM (Object-Relational Mapping) инструмент управления базами данных.

С помощью Prisma были созданы модели базы данных, а с помощью команд поддерживаемых Prisma, данные модели были перенесены в базу данных [10]. Код для описания и создания базы данных с использованием Prisma представлен в приложении Б.

# 2.4 Разработка пользовательского интерфейса

При разработке пользовательского интерфейса была разработана карта навигации. Данная карта представлена на рисунке 8.



Рисунок 8 – Карта навигации

Разработка пользовательского интерфейса велась с использованием компонентного подхода, который предусмотрен библиотекой React. Также при разработке использовались и расширялись библиотеки компонентов.

Настольная и мобильная версии страницы авторизации представлены на рисунке 9 и 10.

**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание**

Рисунок 9 – Настольная версия страницы авторизации

**Изображение выглядит как текст, снимок экрана, Прямоугольник, Шрифт

Автоматически созданное описание**

Рисунок 10 – Мобильная версия страницы авторизации

Страница администратора предоставляет возможность просмотра статусов каждого дома, а также возможность просмотра пользователей приложения и их ролей. Страница просмотра статусов домов представлена на рисунке 11 и 12. Страница просмотра пользователей приложения представлена на рисунке 13 и 14.

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 11 – Настольная версия страницы просмотра статусов

Изображение выглядит как текст, снимок экрана, Веб-сайт, веб-страница

Автоматически созданное описание

Рисунок 12 – Мобильная версия страницы просмотра статусов

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 13 – Настольная версия страницы просмотра пользователей и их ролей

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 14 – Мобильная версия страницы просмотра пользователей и их ролей

Администратор может перейти на любой дом, для редактирования статусов в нем. Процесс редактирования статусов дома представлен на рисунке 15 и 16.

Изображение выглядит как текст, число, линия, Шрифт

Автоматически созданное описание

Рисунок 15 – Процесс редактирования статусов

Изображение выглядит как текст, визитная карточка, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 16 – Информация о успешном изменение статусов

Администратор также имеет возможность регистрации пользователей в системе. Модальное окно регистрации пользователей представлен на рисунке 17.

Изображение выглядит как текст, снимок экрана, Прямоугольник, линия

Автоматически созданное описание

Рисунок 17 – Окно регистрации пользователей в системе

На странице горничной пользователь имеет возможность изменять статусы с «Требует уборки» на «Требует проверки чистоты» и обратно. Страница горничной представлена на рисунке 18.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 18 – Страница горничной

На странице хаусмена, пользователь имеет возможность изменять статусы участка, а также информацию о растопках бани и купели. Страница хаусмена предсталена на рисунке 19.

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание**

Рисунок 19 – Страница хаусмена

# 2.5 Разработка программы

При разработке веб-приложения, была разработана клиентская и серверная часть приложения, в ходе работы были использованы различные алгоритмы. С помощью фреймворка Nest были созданы контроллеры для обработки пользовательских запросов и сервисы для работы с базой данных. Таким образом для более значимых сущностей базы данных существуют контроллеры и сервисы для работы с ними.

Таким образом существуют следующие методы:

* async login(dto: LoginDto): метод для авторизации
* async getNewTokens(accessToken: string): метод для получения токена
* async register(dto: AuthDto): метод для регистрации
* async getHousesFromExcel(file): метод для получения данных из Excel файла
* async updateHousesFromExcel(houses): метод для обновления состояния дома из файла excel
* async updateCountPeople(count: number, houseId: number): метод для обновления количества людей проживающих в доме
* async updateHouseStatus(houseId,statusId, placeId, updateTime=false,timeStart=null, timeEnd=null): метод для обновления статуса
* async getHouseInformation(): метод для получнения информации о домах
* async getHouseInformationById(houseId: number): метод для получения информации о доме по id

Для разработки клиентской части веб приложения была использована библиотека React. Архитектура клиентской части веб приложения представлена на рисунке 20.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 20 – Архитектура клиентской части веб приложения

* Папка «components» содержит в себе экраны и переиспользуемые компоненты системы.
* Папка «Page» содержит в себе целые страницы приложения
* Папка «provider» содержит в себе компоненты, которые оборачивают приложение.
* Папка «routes» задает маршрутизацию в приложении
* Папка «services» предоставляет методы по взаимодействию с api.
* Папка «store» предоставляет хранилище и функции по управление и взаимодействию с хранилищем.

Программный код клиентской и серверной части веб приложения представлен в приложении В.

# 2.6 Тестирование программного продукта

При разработке веб приложения по мониторингу состояния объектов на базе отдыха, были созданы тест кейсы, представленные в приложении Г.

# 2.7 Вывод по специальной части

В результате специальной части проекта была выполнена проектировка системы, включающая разработку диаграммы активности, диаграммы последовательностей, схемы базы данных и карты навигации. Для реализации системы было разработано серверное приложение с использованием фреймворка Nest, а также клиентское приложение на базе React. В результате была достигнута цель по разработке веб-приложения для мониторинга состояния объектов на базе отдыха.

# ЗАКЛЮЧЕНИЕ

В результате проделанной работы была достигнута цель по ускорению процесса обмена информации между работниками. Цель была достигнута посредством проектирования и разработки веб приложения для отслеживания состояния объектов на базе отдыха.

В ходе работы все поставленные задачи были выполнены.

Разработанное решение позволяет автоматизировать и оптимизировать процесс передачи информации о состоянии объектов на базе отдыха, что благоприятно сказывается на скорости работы персонала, за счет оперативного поступления и изменения информации.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. РусПрофайл [Электронный ресурс] / ООО "РусПрофайл". – Электрон. дан. – Режим доступа : https://www.rusprofile.ru/id/6945587. – Загл. с экрана.
2. Использование диаграммы вариантов использования UML при проектировании программного обеспечения [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/566218/. – Загл. с экрана.
3. Bruce Johnson Visual Studio Code. - 1 изд. - Нью-Йорк: Wiley, 2019. - 192 с.
4. Самоучитель Microsoft Visual Studio 2019. — СПб.: БХВ-Петербург, 2019. — 464 с.: ил.
5. Черный Б. Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений. - 1 изд. - СПБ.: Питер, 2021. - 352 с.
6. Гайд по Nest.js: что это такое и как написать свой первый код [Электронный ресурс]. – Режим доступа: https://ru.hexlet.io/blog/posts/gid-po-nest-js. – Загл. с экрана.
7. John C. Practical PostgreSQL. - 3 изд. - Сан-Франциско: O'Reilly Media, 2021. - 651 с.
8. Документация для построения диаграммы последовательности [Электронный ресурс]. – Режим доступа: <https://intuit.ru/studies/courses/32/32/lecture/1014>, свободный. – Загл. с экрана.
9. Создание диаграммы активности. Учебное пособие [Электронный ресурс]. – Режим доступа: <https://anylogic.help/ru/advanced/actionchart/action-chart-tutorial.html>. – Загл. с экрана.
10. Prisma ORM. Руководство. [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/timeweb/articles/654567/. – Загл. с экрана.

# Приложение А

(справочное)

**Результаты анкетирования**

Таблица А.1 ˗ Результаты анкетирования администратора

|  |  |
| --- | --- |
| Вопросы | Ответы |
| ФИО | Иванов Иван Иванович |
| Какую роль вы играете в обеспечении работы объекта базы отдыха? | Я отвечаю за координацию работы всех служб, своевременное бронирование и общение с клиентами. |
| Какие задачи вы решаете в процессе управления объектами? | Контроль за качеством предоставляемых услуг, решение оперативных вопросов, планирование загрузки и распределение задач между сотрудниками. |
| Какие параметры состояния объектов базы отдыха вам критичны? | Чистота и порядок в номерах, исправность всех систем и удобства, доступность и функциональность общественных зон. |
| Какую информацию вы бы хотели видеть в мониторинге состояния объектов? | Состояние номерного фонда, наличие неисправностей, планируемые проверки и ремонты, отзывы клиентов. |

Таблица А.2 ˗ Результаты анкетирования хаусмена

|  |  |
| --- | --- |
| Вопросы | Ответы |
| ФИО | Петров Петр Петрович. |
| Каковы ваши основные обязанности в поддержании объектов базы отдыха? | Обеспечение технического состояния зданий и прилегающих территорий, мелкий ремонт и обслуживание. |
| Какие трудности возникают в процессе мониторинга состояния объектов? | Недостаточная оперативность в получении информации о неисправностях и замечаниях от гостей и персонала. |

Продолжение таблицы А.2

|  |  |
| --- | --- |
| Вопросы | Ответы |
| Какие функции в приложении помогли бы вам в проведении эффективного мониторинга? | Мгновенные уведомления о проблемах, возможность отмечать выполнение задач, планировщик регулярных проверок. |
| Как вы видите оптимальный формат отчетности по состоянию объектов базы отдыха? | Ежедневные и еженедельные отчеты с фотофиксацией выполненных работ и текущим состоянием объектов. |

Таблица А.3 ˗ Результаты анкетирования горничной

|  |  |
| --- | --- |
| Вопросы | Ответы |
| ФИО | Сидорова Анна Михайловна |
| Какие задачи вы выполняете в рамках обслуживания объектов базы отдыха? | Уборка номеров, замена постельного белья и полотенец, пополнение расходных материалов. |
| Каковы основные трудности, с которыми вы сталкиваетесь в коммуникации с коллегами и администраторами? | Иногда информация о специальных запросах гостей доходит с задержкой, что затрудняет своевременное обслуживание. |
| Какие функции или возможности вы бы хотели видеть в приложении для упрощения коммуникации? | Чат для быстрого обмена сообщениями с администрацией и коллегами, функцию списка задач с возможностью отметки о выполнении. |
| Какие виды отчетов были бы вам полезны для повышения эффективности работы? | Отчеты о времени на уборку номера, список проверенных и готовых к заселению номеров, чек-лист уборки с возможностью отметок. |

# Приложение Б

(обязательное)

**Программный код БД**

generator client {

provider = "prisma-client-js"

}

datasource db {

provider = "postgresql"

url = env("DATABASE\_URL")

}

model Role {

roleId Int @id @map("role\_id")

name String

users User[]

status Status[]

}

model User {

userId Int @id @default(autoincrement()) @map("user\_id")

password String @map("password")

firstName String @map("first\_name")

lastName String @map("last\_name")

phone String @unique @map("phone")

createdAt DateTime @default(now()) @map("created\_at")

updatedAt DateTime @updatedAt @map("updated\_at")

roleId Int @map("role\_id")

role Role @relation(fields: [roleId], references: [roleId], onDelete: Cascade, onUpdate: Cascade)

loginLogs LoginLog[]

}

model HouseType {

houseTypeId Int @id @map("house\_id")

name String

houses House[]

}

model House {

houseId Int @id @map("house\_id")

houseTypeId Int @map("house\_type\_id")

peopleCount Int @map("people\_count")

houseStatus HouseStatus[]

houseType HouseType @relation(fields: [houseTypeId], references: [houseTypeId], onDelete: Cascade, onUpdate: Cascade)

}

model Status {

statusId Int @id @map("status\_id")

name String

roleId Int @map("role\_id")

houseStatus HouseStatus[]

placeId Int? @map("place\_id")

place Place? @relation(fields: [placeId], references: [placeId], onDelete: SetNull, onUpdate: Cascade)

role Role @relation(fields: [roleId], references: [roleId], onDelete: Cascade, onUpdate: Cascade)

}

model HouseStatus {

houseId Int @map("house\_id")

statusId Int @map("status\_id")

timeStart DateTime?

timeEnd DateTime?

house House @relation(fields: [houseId], references: [houseId], onDelete: Cascade, onUpdate: Cascade)

status Status @relation(fields: [statusId], references: [statusId], onDelete: Cascade, onUpdate: Cascade)

placeId Int @map("place\_id")

place Place @relation(fields: [placeId], references: [placeId], onDelete: Cascade, onUpdate: Cascade)

@@id([houseId, statusId])

}

model Place {

placeId Int @id @map("place\_id")

name String

houseStatuses HouseStatus[]

Status Status[]

}

model LoginLog {

id Int @id @default(autoincrement())

userId Int

username String

action String // "login" или "logout"

timestamp DateTime @default(now())

user User @relation(fields: [userId], references: [userId], onDelete: Cascade, onUpdate: Cascade)

}

# Приложение В

(обязательное)

**Программный код приложения**

import { ExcelService } from './../excel/excel.service';

import {

BadRequestException,

Body,

Controller,

Get,

HttpCode,

Param,

Post,

Put,

UploadedFile,

UseGuards,

UseInterceptors,

UsePipes,

ValidationPipe,

} from '@nestjs/common';

import { HouseService } from './house.service';

import { Auth } from 'src/auth/decorators/auth.decorators';

import { Roles } from 'src/decorators/roles.decorators';

import { RoleType } from 'src/role/role.enum';

import { RolesGuard } from 'src/decorators/roles.guard';

import { HouseCreateDto } from './dto/house.create.dto';

import { FileInterceptor } from '@nestjs/platform-express';

import { HouseStatusDto } from './dto/houseStatus.update.dto';

@Controller('house')

export class HouseController {

constructor(

private readonly houseService: HouseService,

private excelService: ExcelService,

) {}

@Auth()

@Roles(RoleType.Admin)

@UseGuards(RolesGuard)

@Get('all')

async getAdminHouseInformation() {

return this.houseService.getAdminHouseInformation();

}

@Auth()

@Roles(RoleType.Admin, RoleType.Housemaid)

@UseGuards(RolesGuard)

@Get('housemaidStatus')

async getHousemaidInformation() {

return this.houseService.getHousemaidInformation();

}

@Auth()

@Roles(RoleType.Admin, RoleType.Houseman)

@UseGuards(RolesGuard)

@Get('housemanStatus')

async getHousemanInformation() {

return this.houseService.getHousemanInformation();

}

//TODO вот это надо сделать в целом

@UsePipes(new ValidationPipe())

@Auth()

@HttpCode(200)

@Roles(RoleType.Admin)

@UseGuards(RolesGuard)

@Post('create')

async createHouse(@Body() dto: HouseCreateDto) {}

@Post('updateFromExcel')

@UseInterceptors(FileInterceptor('file'))

async updateHouseFromExcel(@UploadedFile() file) {

if (!file.originalname.match(/\.(xlsx)$/)) {

throw new BadRequestException(

'Неверный формат файла. Пожалуйста, загрузите файл Excel (.xlsx).',

);

}

const houses = await this.excelService.getHousesFromExcel(file);

await this.houseService.updateHouseFromExcel(houses);

}

//!Разделение на разные эндпоинты для огрничения возможности ролей или проверка роли через декоратор

@Put('updateStatus')

async updateStatus(@Body() dto: HouseStatusDto) {

await this.houseService.updateHouseStatus(

+dto.houseId,

+dto.statusId,

+dto.placeId,

dto.updateTime,

dto.timeStart,

dto.timeEnd,

);

}

@Auth()

@Get(':houseId')

async getHouseInformationById(@Param('houseId') houseId: string) {

return this.houseService.getHouseInformationById(+houseId);

}

}

import { HouseStatus } from '../../consts/HouseStatus.const';

import { instance } from '../api/api.interceptors';

import { HttpMethods, createRequestConfig } from '../service.config';

import { HouseEndPoint, HouseEndPointsMap } from './house.config';

import {

IAdminHouseStatus,

IHouseInformation,

IHousemaidHouseInformation,

} from './house.interface';

export const HouseService = {

async getHousemaidStatus() {

const response = await instance<IHousemaidHouseInformation[]>(

createRequestConfig(

HttpMethods.GET,

HouseEndPointsMap[HouseEndPoint.HOUSEMAID\_STATUS],

),

);

return response.data;

},

async uploadExcel(file: any) {

const formData = new FormData();

formData.append('file', file);

const response = await instance<void>({

method: HttpMethods.POST,

url: HouseEndPointsMap[HouseEndPoint.UPDATE\_FROM\_EXCEL],

data: formData,

headers: {

'Content-Type': 'multipart/form-data',

},

});

return response.data;

},

async getHousemanStatus() {

const response = await instance<IHousemaidHouseInformation[]>(

createRequestConfig(

HttpMethods.GET,

HouseEndPointsMap[HouseEndPoint.HOUSEMAN\_STATUS],

),

);

return response.data;

},

async getHouseStatusById(houseId: number) {

const response = await instance<IHouseInformation>(

createRequestConfig(

HttpMethods.GET,

HouseEndPointsMap[HouseEndPoint.HOUSE\_INFORMATION\_BY\_ID],

null,

String(houseId),

),

);

return response.data;

},

async updateStatus(

houseId: number,

placeId: number,

statusId: number,

timeStart: string | null = null,

timeEnd: string | null = null,

updateTime: boolean = false,

) {

const response = await instance<void>(

createRequestConfig(

HttpMethods.PUT,

HouseEndPointsMap[HouseEndPoint.UPDATE\_HOUSE],

{ houseId, placeId, statusId, timeStart, timeEnd, updateTime },

),

);

return response.data;

},

async getAdminStatus() {

const response = await instance<IAdminHouseStatus[]>(

createRequestConfig(

HttpMethods.GET,

HouseEndPointsMap[HouseEndPoint.ALL],

),

);

return response.data;

},

};

import { Tab, Tabs, useMediaQuery, useTheme } from '@mui/material';

import { Link as RouterLink, useLocation } from 'react-router-dom';

import React, { useMemo } from 'react';

import AdminUserRegisterScreen from '../AdminUserRegiserScreen/AdminUserRegisterScreen';

import HouseAdminList from '../../UI/HouseAdminList/HouseAdminList';

const AdminHomeScreen = () => {

const location = useLocation();

const [value, setValue] = React.useState(0);

const theme = useTheme();

const isMobile = useMediaQuery(theme.breakpoints.down('sm'));

const tabs = useMemo(

() => [

{ label: 'Дома', path: '/' },

{ label: 'Пользователи', path: '/registration' },

],

[],

);

React.useEffect(() => {

// Находим индекс таба, который соответствует текущему маршруту

const tabIndex = tabs.findIndex(tab => tab.path === location.pathname);

// Если индекс найден, обновляем состояние value

if (tabIndex !== -1) {

setValue(tabIndex);

}

}, [location, tabs]);

const handleChange = (event: any, newValue: any) => {

setValue(newValue);

};

return (

<>

<Tabs

value={value}

onChange={handleChange}

variant={isMobile ? 'scrollable' : 'standard'}

scrollButtons='auto'

textColor='primary'

indicatorColor='primary'

sx={{

bgcolor: '#ffffff', // цвет фона таба

'& .MuiTab-root': {

color: '#236092', // цвет текста таба

'&.Mui-selected': {

fontWeight: '600',

color: '#236092', // цвет текста активного таба

},

},

'& .PrivateTabIndicator-root': {

backgroundColor: '#236092', // цвет индикатора

},

}}

>

{tabs.map((tab, index) => (

<Tab

key={index}

label={tab.label}

component={RouterLink}

to={tab.path}

/>

))}

</Tabs>

{/\* Отображаемый компонент зависит от текущего маршрута \*/}

{location.pathname === '/registration' && <AdminUserRegisterScreen />}

{location.pathname === '/' && <HouseAdminList />}

{/\* Здесь должен быть компонент для маршрута Settings \*/}

</>

);

};

export default AdminHomeScreen;

# Приложение Г

(обязательное)

**Тест кейсы**

Таблица Г.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | заголовок | Действия | Исходные данные | Ожидание: | Реальность | Итог |
| 1 | Проверка авторизации в роли администратора | 1.Нажать на вкладку авторизации  2.Ввести тестовые данные надать кнопку войти | Логин: +7999999999999 Пароль: Sofa2005 | Успешный вход отображается страница администратора | Успешный вход отображается страница администратора | Passed |
| 2 | Проверка авторизации в роли горничной | 1.Нажать на вкладку авторизации  2.Ввести тестовые данные надать кнопку войти |  | Успешный вход отображается страница горничной | Успешный вход отображается страница горничной | Passed |
| 3 | Проверка авторизации в роли хаусмена | 1.Нажать на вкладку авторизации  2.Ввести тестовые данные надать кнопку войти |  | Успешный вход отображается страница хаусмена | Успешный вход отображается страница хаусмена | passed |
| 4 | Проверка возможности регистрации новых пользователей | Авторизация (тест 1)  1)Нажать на вкладку добавление пользователей  2. Заполнить данные ФИО номер и роль, и пароль |  | В списке сотрудников появляется новая запись, соответствующая только что заполненной | В списке сотрудников появляется новая запись, соответствующая только что заполненной | passed |
| 5 | Изменение статуса на "Убрано" | Авторизация (тест 1)1. Выбрать место из списка. 2. Нажать на кнопку изменения статуса места. 3. Выбрать статус "Убрано" из списка доступных статусов. 4. Подтвердить |  | Статус места успешно изменён на "Убрано". Изменение отображается в интерфейсе приложения. | Статус места успешно изменён на "Убрано". Изменение отображается в интерфейсе приложения. | passed |

Продолжение таблицы Г.1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 6 | Изменение статуса на "Требуется уборка" | Авторизация (тест 1)  1. Выбрать место из списка. 2. Нажать на кнопку изменения статуса места. 3. Выбрать статус "Требуется уборка"  4.Подтвердить изменение статуса. |  | Статус места успешно изменён на "Требуется уборка". Изменение отображается в интерфейсе приложения. | Статус места успешно изменён на "Требуется уборка". Изменение отображается в интерфейсе приложения. | passed |
| 7 | Проверка уведомлений об изменении статуса | Авторизация (тест 1)  1. Выбрать место из списка. 2. Нажать на кнопку изменения статуса места. 3. Выбрать любой статус из списка доступных статусов. 4. Подтвердить изменение статуса. |  | Пользователь получает уведомление об изменении статуса места | Пользователь получает уведомление об изменении статуса места | passed |
| 8 | Проверка возможности входа в приложение с неправильным номером телефона | 1.перейти на страницу авторизации  2.ввести тестовые данные  3.нажать кнопку войти | Логин: +7999999999999 Пароль: Sofa2005 | Пользователь остаётся на странице логина Данные с формы логина пропадает Сверху справа появляется подсказка о неправильном вводе номера телефона | Пользователь остаётся на странице логина Данные с формы логина пропадает Сверху справа появляется подсказка о неправильном вводе номера телефона | passed |
| 9 | Проверка возможности входа в приложение с неправильным паролем | 1.перейти на страницу аввторизации  2.ввести тестовые данные  3.нажать кнопку войти | Логин: +79650561757 Пароль: 11111 | Пользователь остаётся на странице логина Данные с формы логина пропадает Сверху справа появляется подсказка о неправильном вводе пароля | Пользователь остаётся на странице логина Данные с формы логина пропадает Сверху справа появляется подсказка о неправильном вводе пароля | passed |
| 10 | Проверка изменения статусов с аккаунта горничной | 1. Авторизироваться с акаунта горничной (тест 2) 2. Изменить статус дома на убрано 3. Авторизироваться с аккаунта администратора (тест 1) | Авторизация с аккаунта горничной: +79638527845  Nadya1  Авторизация с аккаунта админа: +79650561757 Sofa2005 | При изменение с аккаунта горничной статуса места у администратора на главной странице автоматически отображается новый статус | При изменение с аккаунта горничной статуса места у администратора на главной странице автоматически отображается новый статус | passed |

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО ПРЕДДИПЛОМНОЙ ПРАКТИКЕ (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/4 группы

Толстых София Александровна

(Фамилия, имя, отчество)

Место прохождения практики: ООО «СВИТЛАЙФ МАРКЕТ», Санкт-Петербург, наб. Обводного канала, д. 93а литер а, помещ. 5н офис 1449

(наименование и адрес организации)

Период прохождения практики

с «22» апреля 2024 г. по «18» мая 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| **Тема 1.** Ознакомление со структурой и характером деятельности предприятия. Организационные вопросы оформления на предприятии, установочная лекция, инструктаж по охране труда и технике безопасности, распределение по рабочим местам | 30 |  |
| **Тема 2**. Сбор материалов для составления технического задания по теме дипломного проекта | 26 |  |
| **Тема 3.** Разработка программного обеспечения на основе технического задания дипломного проекта | 30 |  |
| **Тема 4.** Проведение испытаний, отладка и внедрение программного продукта на предприятии | 16 |  |
| **Тема 5.** Расчет показателей экономической эффективности программного продукта | 22 |  |
| **Тема 6.** Оформление отчета о прохождении производственной практики (преддипломной) | 20 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, **освоены /** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Иванова Д.В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «18» мая 2024 г.