СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc133957633)

[1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ВЕБ-ПРИЛОЖЕНИЮ 5](#_Toc133957634)

[1.1 Анализ литературных источников 5](#_Toc133957635)

[**1.1.1** Возникновение систем бронирования и резервирования 5](#_Toc133957636)

[**1.1.2** Роль систем резервирования мест в современном мире 6](#_Toc133957637)

[**1.1.3** Различие бронирования и резервирования 8](#_Toc133957638)

[1.2 Анализ существующих аналогов 8](#_Toc133957639)

[**1.2.1** Talon.by 8](#_Toc133957640)

[**1.2.2** Relax.by 10](#_Toc133957641)

[**1.2.3** 24-ая городская поликлиника спецмедосмотров 11](#_Toc133957642)

[1.3 Необходимый функционал 12](#_Toc133957643)

[1.4 Формирование требований к проектируемому программному средству 12](#_Toc133957644)

[2 АНАЛИЗ ТРЕБОВАНИЙ К ВЕБ-ПРИЛОЖЕНИЙ И РАЗРАБОТКА СПЕЦИФИКАЦИИ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 14](#_Toc133957645)

[2.1 Описание функциональности программного средства 14](#_Toc133957646)

[2.2 Спецификация функциональных требований 19](#_Toc133957647)

[3 ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ 22](#_Toc133957648)

[3.1 Разработка архитектуры веб-приложения 22](#_Toc133957649)

[3.2 Разработка модели базы данных 24](#_Toc133957650)

[3.3 Разработка алгоритмов веб-приложения 29](#_Toc133957651)

[4 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ 32](#_Toc133957652)

[4.1 Обоснование выбора языка программирования для создания серверной части приложения 32](#_Toc133957653)

[4.2 Обоснование выбора фреймворка серверной части приложения 33](#_Toc133957654)

[4.3 Обоснование выбора языка программирования и фреймворка для создания пользовательского интерфейса веб-приложения 35](#_Toc133957655)

[4.4 Обоснование выбора средств разработки 36](#_Toc133957656)

[4.5 Описание взаимодействия серверной и пользовательской части приложения 38](#_Toc133957657)

[5 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ РЕЗЕРВИРОВАНИЕМ МЕСТ НА ЯЗЫКЕ JAVA 40](#_Toc133957658)

[5.1 Характеристика разработанного по индивидуальному заказу веб-приложения 40](#_Toc133957659)

[5.2 Расчет затрат на разработку и цена веб-приложения, созданного по индивидуальному заказу 40](#_Toc133957660)

[5.3 Расчет результата от разработки и продажи веб-приложения, созданного по индивидуальному заказу 44](#_Toc133957661)

[5.4 Расчет показателей экономической эффективности разработки веб-приложения 44](#_Toc133957662)

[Список использованных источников 45](#_Toc133957663)

# ВВЕДЕНИЕ

В современном мире все больше людей используют интернет для заказа билетов на различные мероприятия, бронирования мест в кинотеатрах, ресторанах и других местах. Это означает, что разработка веб-приложений для резервирования мест является актуальной задачей, которая может значительно упростить жизнь людей.

Цель данного дипломного проекта - создание веб-приложения для резервирования мест на языке Java, которое будет позволять пользователям быстро и удобно бронировать места на различных мероприятиях. Для достижения этой цели были определены требования к функциональности и дизайну приложения, были выбраны технологии и фреймворки, которые будут использоваться в процессе разработки.

Одним из основных преимуществ создания веб-приложения для резервирования мест на языке Java является возможность использования объектно-ориентированного программирования. Это позволяет создавать модульные и масштабируемые приложения, которые могут быть легко модифицированы и расширены в будущем.

В процессе разработки веб-приложения на языке Java были использованы современные фреймворки и библиотеки, такие как Spring и Hibernate, которые значительно упростили разработку и повысили качество приложения.

В результате создано удобное и функциональное веб-приложение для резервирования мест на языке Java, которое позволяет пользователям быстро и легко выбирать места на мероприятиях и получать электронные оповещения на почту. Веб-приложение имеет простой и интуитивно понятный интерфейс, который позволяет пользователям легко ориентироваться и бронировать места в несколько кликов.

Таким образом, создание веб-приложения для резервирования мест на языке Java является важной задачей, которая может значительно облегчить жизнь людей. Разработанное в ходе данного дипломного проектирования веб-приложение является удобным и функциональным инструментом для резервирования мест на различных мероприятиях.

# АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ВЕБ-ПРИЛОЖЕНИЮ

## Анализ литературных источников

### Возникновение систем бронирования и резервирования

Бурное развитие туристской индустрии в последние десятилетия связано с двумя факторами: развитием гражданской авиации и созданием компьютерных систем бронирования. В свою очередь, увеличение числа авиалиний, самолетов, а также рост объемов авиаперевозок закономерно привели к необходимости создания и использования компьютерных систем бронирования CRS (Computer Reservation System), которые стали основным инструментом для резервирования авиабилетов. Первые системы резервирования появились на рынке в середине 60-х гг. XX в. Это были системы Apollo! фирмы United Airlines и Sabre фирмы American Airlines. Первоначально развитие компьютерных систем бронирования шло путем установки терминалов в офисах самих авиакомпаний. Это позволило существенно повысить качество и эффективность работы персонала авиакомпаний, улучшить качество обслуживания клиентов за счет сокращения времени оформления билетов, сократить штат сотрудников, обеспечивающих бронирование билетов. Техническим оснащением первых центров бронирования занималась фирма IBM, которая до сих пор поставляет оборудование для систем резервированиях [1].

Несколько позже началась установка терминалов бронирования в крупных туристских агентствах. С появлением таких систем турагенты смогли осуществлять резервирование авиабилетов в режиме реального времени в считанные секунды. Внедрение систем бронирования позволило существенно сократить время обслуживания клиентов и обеспечить резервирование в режиме on-line, снизить себестоимость предлагаемых услуг, в частности, за счет сокращения числа сотрудников, осуществляющих бронирование, увеличить количество и разнообразить предлагаемые услуги. Наряду с этим появилась возможность реализации стратегии гибкого ценообразования, оптимизации загрузки самолетов, стыковки рейсов различных авиакомпаний, оптимизации формирования маршрута перевозки туристов по цене, времени полета и осуществления других задач.

Оценив эффективность новой технологии бронирования и перспективы их использования, авиакомпании существенно расширили как спектр представленной в системах бронирования информации, так и функциональные возможности этих систем. Первоначально ориентированные на отдельно взятую авиакомпанию, эти системы превратились в программные комплексы, обслуживающие группы авиакомпаний и предоставляющие ряд дополнительных услуг по бронированию мест в гостиницах, заказу железнодорожных билетов, прокату автомобилей и т.д. Теперь в системах бронирования заложена информация не только о наличии мест, но и общая информации о рейсах, типах задействованных самолетов, подробном описании тарифов, а также информация о смежных отраслях туристского бизнеса: прокате автомобилей, размещении в гостиницах, продаже железнодорожных билетов и др.

Развитие и широкое распространение компьютерных систем резервирования CRS осуществлялись, с одной стороны, за счет увеличения числа предоставляемых услуг, а с другой стороны, за счет сокращения стоимости электронных услуг бронирования и резервирования [2]. Расширение сферы деятельности происходило по всем направлениям предоставляемых турбизнесом услуг, р том числе проживание, развлечения и т.д. Это привело к созданию глобальных распределительных систем – GDS [3]. В дополнение к возможностям электронного сервиса при бронировании меcт для транспортирования авиапассажиров эти системы позволяет резервировать места в гостиницах, брать на прокат автомобили, осуществлять обмен валюты, производить заказ билетов на развлекательные и спортивные программы и т.д. В практике западных туристских фирм эти системы естественным образом встроены в технологическую цепочку формирования и обработки туристской информации, а через механизм кредитных карт обеспечивается финансовая ответственность при бронировании туров. Через глобальные распределительные системы можно связаться с сотнями авиакомпаний, тысячами гостиниц и фирмами проката автомобилей, забронировать билеты на железнодорожные, морские и паромные перевозки, осуществить резервирование билетов на посещение всевозможных культурных мероприятий.

### Роль систем резервирования мест в современном мире

Системы бронирования и резервирования мест широко применяются в различных сферах жизни людей, включая транспортную и культурную инфраструктуру, спортивные мероприятия и прочие события. Они позволяют людям заранее планировать свои поездки и участие в различных мероприятиях, упрощая процесс покупки билетов и предоставляя возможность выбрать нужные услуги и места.

Одним из наиболее распространенных применений систем бронирования и резервирования мест является транспортная сфера. Они используются для бронирования и продажи билетов на поезда, автобусы, самолеты и другие виды транспорта. Эти системы позволяют пассажирам заранее забронировать места на нужный рейс, выбрать тип места и дополнительные услуги, такие как питание и развлечения. Без таких систем было бы гораздо сложнее планировать поездки и гарантировать наличие мест на нужном рейсе.

Системы бронирования и резервирования мест также широко используются в гостиничном бизнесе. Они позволяют гостям заранее забронировать номера на нужные даты, выбрать тип номера и дополнительные услуги, такие как завтрак, уборка номера и бассейн. Эти системы упрощают процесс бронирования и гарантируют наличие нужного номера на нужную дату, что особенно важно в высокий сезон.

Системы бронирования и резервирования мест в культурной сфере играют огромную роль в повышении доступности и комфортности посещения мероприятий. С их помощью можно легко забронировать и оплатить билеты на концерты, спектакли, выставки и другие события, что позволяет избежать очередей на кассе и гарантировать место в зале. Кроме того, системы бронирования и резервирования мест помогают организаторам мероприятий более эффективно управлять своими ресурсами и предоставлять более высокий уровень обслуживания посетителям.

Системы бронирования и резервирования мест также способствуют развитию культурной индустрии и повышению ее экономической значимости. Благодаря таким системам организаторы мероприятий имеют возможность привлекать большую аудиторию, оптимизировать залы и создавать условия для привлечения спонсоров. Более того, системы бронирования и резервирования мест позволяют посетителям планировать свое время и досуг, что в свою очередь влияет на общую культурную активность и разнообразие предложений в этой сфере.

Таким образом, системы бронирования и резервирования мест являются неотъемлемой частью современного мира. Они упрощают процесс бронирования и позволяют сэкономить время и силы, которые были бы затрачены при использовании традиционных методов бронирования. Такие системы также повышают эффективность и точность процессов управления ресурсами, а также обеспечивают лучший уровень обслуживания для клиентов.

Несмотря на ряд преимуществ, связанных с использованием систем бронирования и резервирования мест, их применение может вызывать определенные проблемы. В частности, возможна необходимость поддержания сложной инфраструктуры и системы обработки данных. Кроме того, в случае сбоев в системе или неправильного использования, могут возникнуть серьезные последствия для бизнеса и клиентов.

Тем не менее, современные технологии и методы разработки программного обеспечения позволяют создавать более надежные и безопасные системы бронирования и резервирования мест. Это в свою очередь способствует увеличению удовлетворенности клиентов и повышению эффективности бизнеса.

В целом, системы бронирования и резервирования мест остаются важным элементом современной инфраструктуры и играют ключевую роль в упрощении и повышении эффективности процессов бронирования и управления ресурсами. При этом разработчики и бизнес-пользователи должны постоянно совершенствовать их функциональность, надежность и безопасность, чтобы обеспечить наилучший уровень обслуживания для клиентов и повысить эффективность своего бизнеса.

### Различие бронирования и резервирования

Резервирование и бронирование – это два термина, которые часто используются в сфере туризма, гостиничного бизнеса и культурных мероприятий. В обоих случаях речь идет о процессе сохранения места или номера для будущего использования, но между ними есть определенные различия.

Резервирование, в отличие от бронирования, не требует оплаты заранее. Клиент может зарезервировать место или номер, не делая предоплату, и сохранить его на определенный период времени. Однако, если он не подтвердит бронирование в указанный срок, резервация будет отменена, и место освободится для других клиентов.

Бронирование, в свою очередь, предполагает оплату заранее за сохранение места или номера. После подтверждения бронирования, место становится недоступным для других клиентов, и они не могут его забронировать. Клиент получает гарантию, что место будет его, если он оплатит заранее.

Кроме того, резервирование может быть более гибким вариантом, поскольку клиент может изменить или отменить резервацию без штрафных санкций. В то время как при бронировании, отмена или изменение брони может потребовать дополнительных расходов или даже потерю предоплаты. Также бронирование может предоставить дополнительные преимущества, такие как гарантированный номер или выделенное место на мероприятии.

В целом, выбор между резервированием и бронированием зависит от конкретной ситуации и потребностей клиента. Если клиент не уверен в своих планах или нуждается в более гибком варианте, резервирование может быть предпочтительнее. Если же клиент точно знает, что ему нужно место, и он готов оплатить за это заранее, то бронирование может быть более удобным вариантом.

Исходя из этого, следует подчеркнуть, что данное веб-приложение разработано именно с целью предоставления услуг резервирования.

## Анализ существующих аналогов

### Talon.by

Талон.бай – это единый пациентский онлайн-сервис в Беларуси, который поставляет услуги по заказу талонов к врачу через интернет, вызов врача на дом онлайн [4].

Талон.бай начал работать в 2011 году как первая интернет-регистратура. Тогда к нашему сервису были подключены всего 5 поликлиник. Сегодня через Талон.бай вы можете записаться к врачам из 275 государственных и частных медучреждений. И их число из года в год растет. А сам Талон.бай стал больше, чем только онлайн-запись к врачам.

Сервис предоставляет большое множество услуг. Через сайт вы можете заказать талон к врачу в своей поликлинике. Сервис позволяет выбрать удобный для вас день и время визита к специалисту. Вы можете вызвать врача на дом день в день, если поликлиника активировала эту функцию на своей странице. Чтобы получить врачебную консультацию быстрее или у конкретного специалиста, вы можете записаться на платный прием. Кроме государственных, к сервису подключены и частные медцентры, где медицинскую помощь оказывают на платной основе. Также вы можете выбрать необходимые анализы и в течение месяца в любой удобный день сдать эти анализы в любом отделении Синэво. При заказе анализов онлайн у вас будет право приоритетной очереди, когда вы придете в пункт сдачи анализов. К сервису подключена сеть аптек «Белфармации» в г. Минске. В дальнейшем подключатся другие аптеки – и он будет работать в других населенных пунктах.

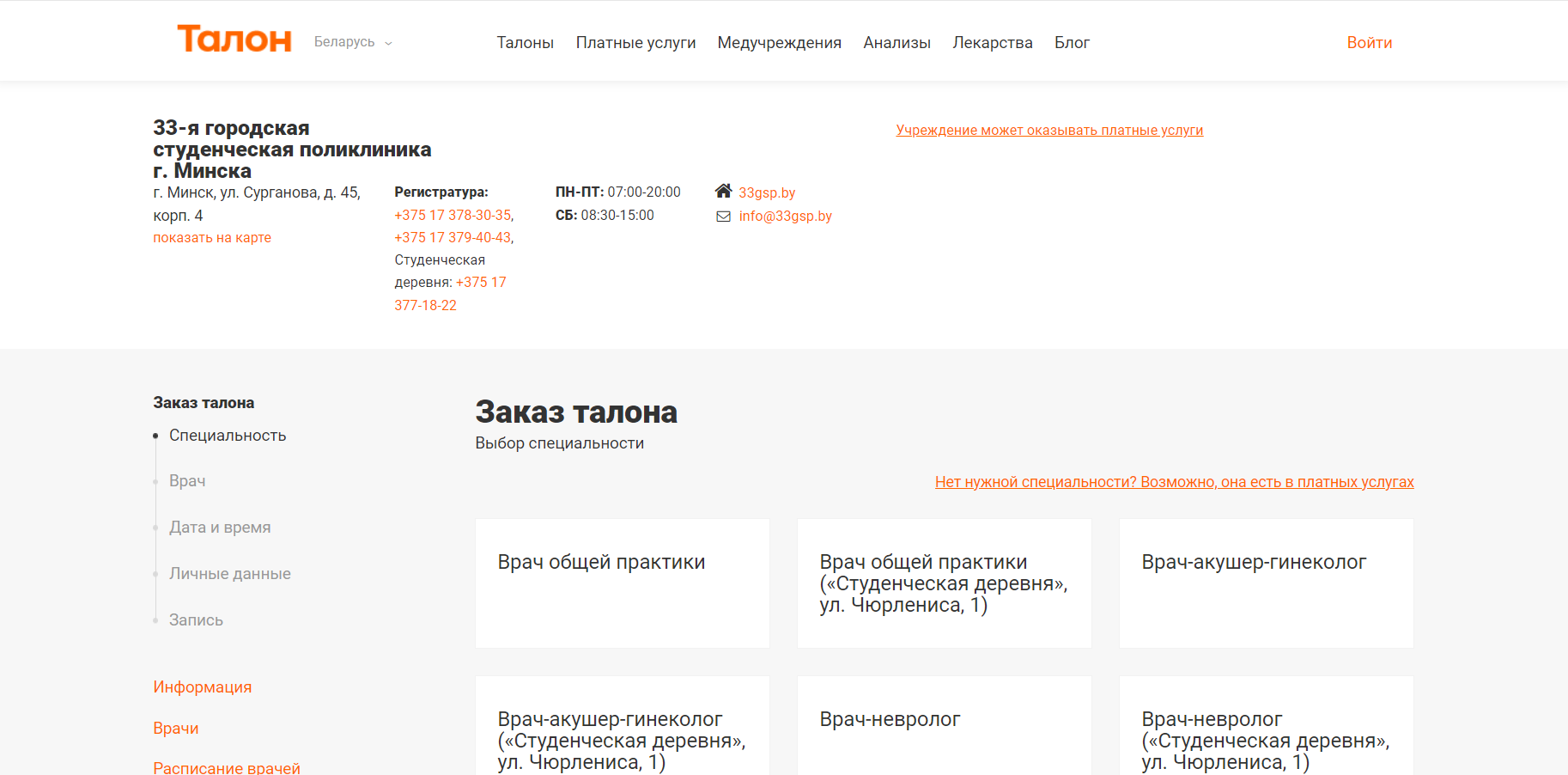


Рисунок – 1.1 Вид страницы заказа талона

На главной странице можем найти различные разделы: талоны, платные услуги, медучереждения, анализы, лекарства, блог. На сайте присутствует возможность пользования личным кабинетом, в котором можно отслеживать список заказанных талонов, вызовы врача, анализы и добавление пациентов.

Заказ талона происходит в определенном порядке:

* выбор поликлиники;
* выбор специальности врача;
* выбор врача;
* выбор времени в очереди;
* заполнение личных данных;
* оформление записи.

После оформления заказа, талон оказывается в списке взятых в личном кабинете, а также пользователь получает письмо на почтовый ящик.

Из плюсов данного сервиса хотелось бы выделить e-mail оповещения, достоверная и актуальная информация, а также дружелюбный пользовательский интерфейс.

### Relax.by

Relax.by ― информационно-развлекательный портал об отдыхе и развлечениях в Беларуси, созданный в 2006 году [5]. Проект входит в состав компании ARTOX, которая с 2012 года является резидентом Парка высоких технологий. Портал ориентирован как на пользователей – жителей Беларуси, так и на бизнес. Пользователям Relax.by помогает бронировать столики в лучших заведениях Минска, записываться онлайн на бьюти-процедуры, совершать покупки со скидками, находить подарки для любимых, узнавать городские новости и выбирать, какие события посетить в будни или выходные.

Бизнесу Relax.by предлагает комплексное обслуживание, которое обеспечивает стабильный приток клиентов и значительно укрепляет имиджевые позиции на рынке.



Рисунок 1.2 – Аудитория портала Relax.by

Миссия **–** дать пользователям максимально полную информацию обо всех заведениях города, предоставить качественный сервис по бронированию столиков и онлайн-записи. А для компаний **–** обеспечить мощное продвижение в интернете, активно способствовать развитию бизнеса и повышению узнаваемости брендов, увеличению количества постоянных клиентов.

На главной странице перечислены категории предоставляемых услуг, наиболее популярные разделы, последние новости и блог сервиса. Также присутствует личный кабинет, в котором можно найти свои забронированные услуги и избранные заведения.

Порядок одного из вариантов резервирования:

* выбор главной категории на главной странице;
* выбор интересующей подкатегории;
* добавление параметров фильтрации поиска;
* оформление резервации (ввод личных данных, даты и времени).

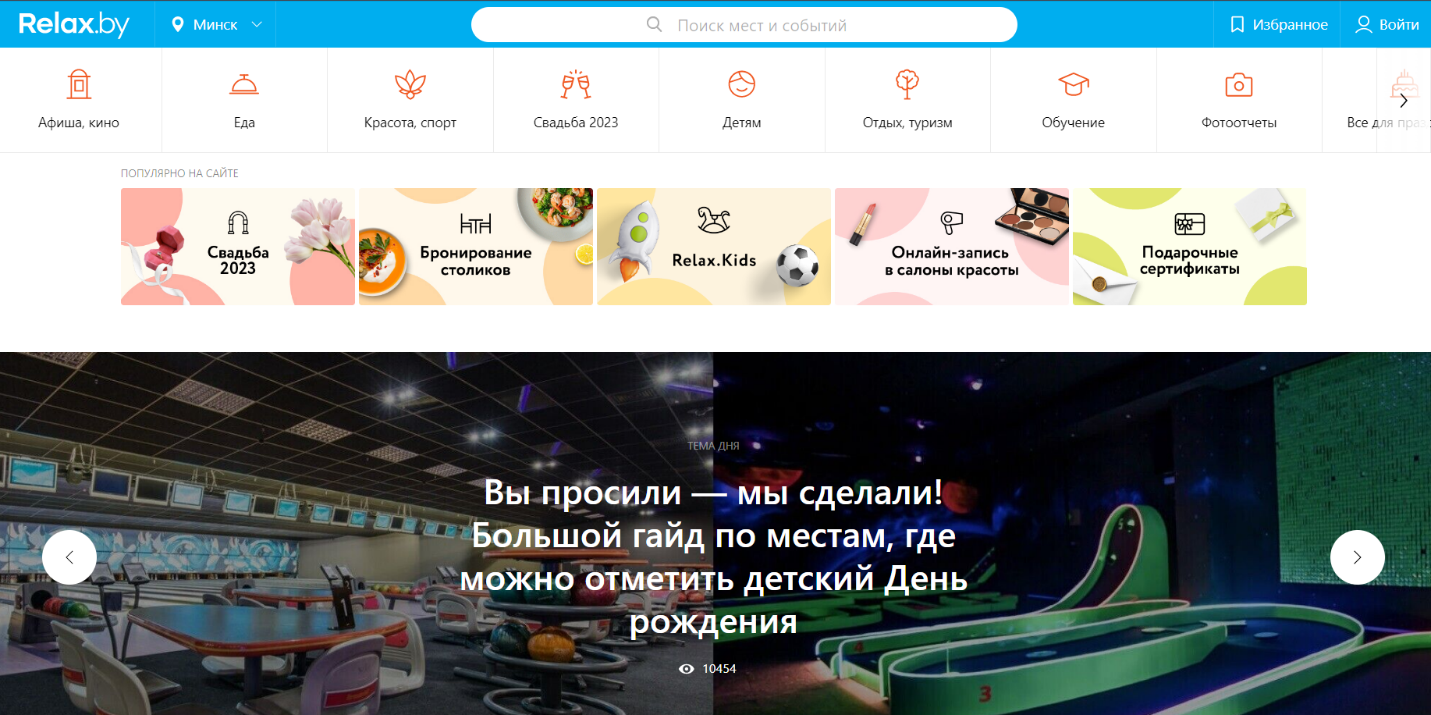


Рисунок 1.3 – Вид главной страницы сервиса Relax.by

Плюсами данного сервиса являются: удобная система фильтрации заведений, отображение карты местоположений, наличие отзывов о заведении.

### 24-ая городская поликлиника спецмедосмотров

Сайт 24й городской поликлиники является официальным ресурсом данной поликлиники. На нем предоставляется информация о порядке получения медицинской справки водителям, получению медицинской справки на работу, прохождении медосмотров на владение оружием, информация об отделении профилактических осмотров. Также текущая информация по работе поликлиники и записи [6].

Немаловажным для такого учреждения является функционал онлайн-калькулятора, позволяющий произвести подсчет расходов на все процедуры заранее.

На странице заказов описаны правила, по которым стоит подавать заявки и в каком порядке.

Порядок оформления талона:

* ознакомиться с правилами;
* ввод данных (личные, дата, время).

Единственным плюсом такой системы является предельная ясность оформления.



Рисунок 1.4 – Внешний вид страницы заказа талонов в 24-ю

Поликлинику

## Необходимый функционал

В процессе анализа спектра приложений и сервисов для предоставления услуг по резервации было выявлено, что большинство сервисов поддерживают функцию личного кабинета, обладают расширенным поиском объявлений и предложений по категориям, всевозможными фильтрациями результатов поиска при поиске интересуемых услуг. Причем личный кабинет обладает самыми различными возможностями: перечень избранных услуг, перечень рекомендаций, история бронирования и резерваций, интеграция с аккаунтами сторонних сервисов.

Общими для всех являются следующие элементы: личный кабинет и возможность изменения личных данных, расширенный поиск, основанный на категориях – эти функции были реализованы в веб-приложении.

## Формирование требований к проектируемому программному средству

Система, которая должна быть разработана, должна предоставлять возможность по размещению мест для резервирования, а также резервации этих мест в реальном времени.

Доступ к разработанному веб-приложению должно быть реализовано путем создания веб-сервиса, что позволит избежать проблем, связанных с совместимостью с различными операционными системами, а также предоставить единообразный интерфейс для доступа.

Программное средство должно обеспечивать выполнение следующего перечня функций:

* размещение резервируемых услуг (мест, билетов, предметов, объектов и так далее);
* резервирование услуг (мест, билетов, предметов, объектов и так далее);
* личный кабинет пользователя;
* категоризация размещенных услуг (мест, билетов, предметов, объектов и так далее);
* учет порядочности пользователя.

Разрабатываемая система должна соответствовать следующим требованиям, перечень которых был сформирован по итогам анализа литературных источников и существующих аналогов:

* разграничение прав доступа к различным компонентам системы посредством введения механизма регистрации и авторизации;
* удобный формат API-запросов к веб-сервису;
* обеспечение сохранности данных пользователей;
* предоставление графического пользовательского интерфейса к веб-сервису.

Система должна обеспечивать валидацию входных данных:

* личные данные;
* данные резервации;
* обратная связь поставщика услуг для резервации.

# АНАЛИЗ ТРЕБОВАНИЙ К ВЕБ-ПРИЛОЖЕНИЙ И РАЗРАБОТКА СПЕЦИФИКАЦИИ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

## Описание функциональности программного средства

Для представления функциональной модели была выбрана диаграмма вариантов использования UML [7], которая отражает отношения между актерами и прецедентами и позволяет описать систему на концептуальном уровне. Прецедент соответствует отдельному сервису системы, определяет один из вариантов ее использования и описывает типичный способ взаимодействия пользователя с системой. UML предназначен для определения, визуализации, проектирования и документирования программных систем.

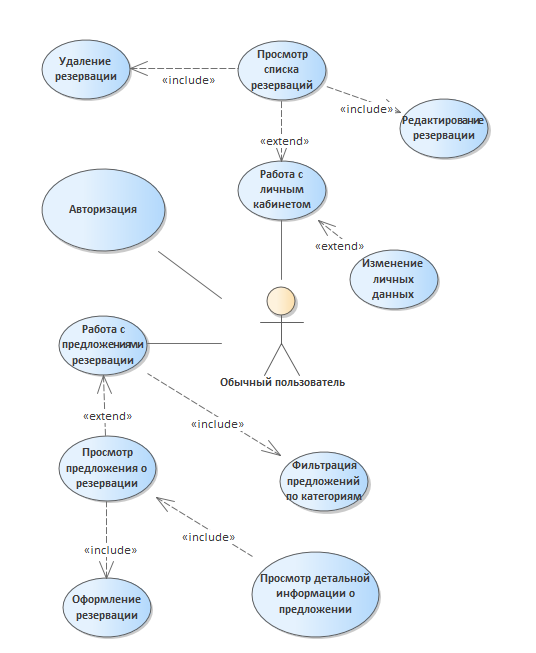


Рисунок 2.1 – Диаграмма вариантов использования для обычного пользователя

На диаграмме можно выделить два основных составляющих элемента – актер и прецедент. Актер – стилизованный человек, обозначающий набор ролей пользователя, взаимодействующего с некоторой сущностью. Прецедент – эллипс с надписью, обозначающий выполняемые системой действия, приводящие к наблюдаемым актером результатам.

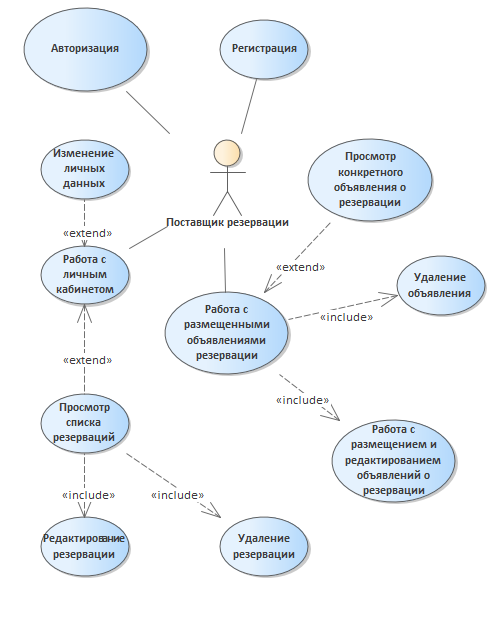


Рисунок 2.3 – Часть диаграммы вариантов использования для

поставщика резервации

На основании представленной диаграммы вариантов использования можно сделать вывод, что в системе будет существовать пять основных актеров:

* обычный пользователь;
* гость;
* поставщик резервации;
* ответственное лицо резервации;
* администратор системы.

Далее подробно описываются прецеденты каждого актера.

Обычному пользователю предоставляются возможности:

* работа с личным кабинетом:
  1. просмотр списка резерваций;
  2. изменение личных данных;
* работа с предложениями резервации:
  1. поиск предложений;
  2. просмотр детальной информации о предложении.

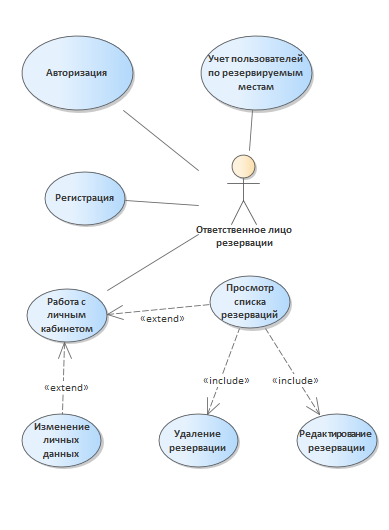


Рисунок 2.3 – Диаграмма вариантов использования для

ответственного лица резервации

Гостю предоставлен следующий функционал:

* регистрация;
* работа с предложениями резервации:

1. поиск предложений;
2. просмотр детальной информации о предложении.

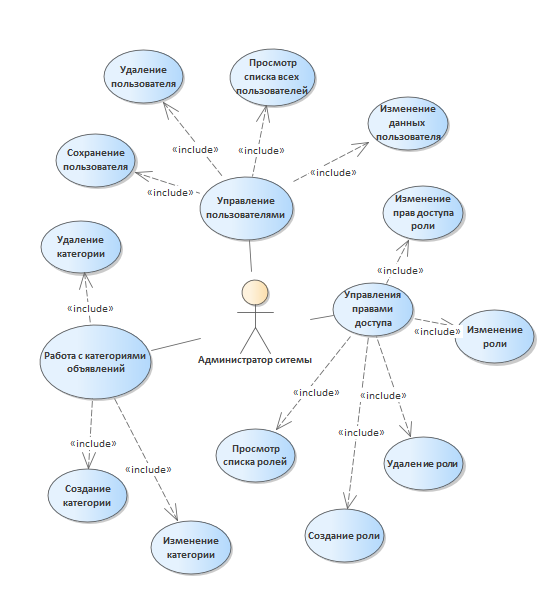


Рисунок 2.4 – Часть диаграммы вариантов использования для

администратора системы

Поставщик резервации имеет доступ к следующему функционалу систему:

* регистрация;
* работа с личным кабинетом:
* просмотр списка резерваций;
* изменение личных данных;
* работа с объявлениями о резервации:

1. размещение и редактирование объявлений;
2. просмотр объявлений.

Действия по размещению и редактированию объявлений же включают в себя:

* работа с местоположением резервации;
* работа с событиями резервации;
* работа с категориями объявлений о резервации;
* работа с резервацией на основе очереди;
* работа с резервацией на основе перечня объектов (услуг):
* работа с типами объектов (услуг);
* работа с резервацией на основе местонахождения;
* организация структурных единиц местонахождения;

Просмотр объявлений же включает в себя:

* просмотр статистических данных;
* изменение статуса резервации.

Ответственное лицо резервации имеет доступ к следующим функциям системы:

* регистрация;
* работа с личным кабинетом:
  1. просмотр списка резерваций;
  2. изменение личных данных;
* учет пользователей по резервируемым местам.

Администратор (для данного программного средства данным пользова­телем является администратор сервера) имеет доступ к следующим возмож­ностям:

* работа с личным кабинетом:
  1. просмотр списка резерваций;
  2. изменение личных данных;
* работа с предложениями резервации:
  1. поиск предложений;
  2. просмотр детальной информации о предложении;
* работа с объявлениями о резервации:
  1. размещение и редактирование объявлений;
  2. просмотр объявлений;
* управление пользователями системы;
* управление категориями объявлений;
* управление ролями пользователей системы.

Действия по размещению и редактированию объявлений состоит из:

* работа с событиями резервации;
* работа с местоположением резервации;
* работа с категориями объявлений о резервации;
* работа с резервацией на основе очереди;
* работа с резервацией на основе перечня объектов (услуг):
  1. работа с типами объектов (услуг);
* работа с резервацией на основе местонахождения;
  1. организация структурных единиц местонахождения;

Просмотр объявлений же включает в себя:

* просмотр статистических данных;
* изменение статуса резервации.

## Спецификация функциональных требований

Спецификация требований к программному обеспечению – это описание поведения системы, которую необходимо разработать. Спецификация функциональных требований включает пользовательские сценарии, которые носят название варианты использования. Они описывают все варианты взаимодействия между пользователями и информационной системой. Кроме пользовательских сценариев спецификация также содержит требования, которые могут определять ограничения на реализацию, например, такие как стандарты качества, требования производительности или какие-либо проектные ограничения.

Веб-приложение состоит из двух логически раздельных частей backend и frontend. Backend представлен в виде серверной части приложения, написанном на языке Java c помощью Spring Framework. Frontend же представляет собой приложение на ReactJs. Оба приложения могут располагаться как на одном устройстве, так и раздельно.

Дополнительной функцией разрабатываемого программного обеспечения является предоставление статистических данных о резервации.

Данная спецификация включает в себя следующие функциональные требования к разрабатываемому приложения:

* регистрация пользователя должна быть доступна любому неавторизованному пользователю веб-приложения, то есть пользователю с ролью «Гость»;
* регистрация пользователя в веб-приложении должна производиться путем заполнения регистрационной формы;
* регистрационная форма должна содержать следующие данные со стороны пользователя веб-приложения:
  1. адрес электронной почты, который является уникальным;
  2. пароль;
  3. имя;
  4. фамилия;
* регистрация может производиться путем создания нового пользователя администратором системы;
* авторизация пользователя должна быть доступна зарегистрированному пользователю веб-приложения;
* авторизация пользователя в веб-приложении должна производиться после ввода логина и пароля со стороны пользователя и проверки данных авторизации с данными, указанными при регистрации пользователя в веб-приложении;
* редактирование профиля пользователя должно быть доступно зарегистрированному пользователю веб-приложения;
* редактирование профиля пользователя должно включать в себя:
  1. возможность изменения данных авторизации (почты и пароля);
  2. возможность изменения личных данных;
* редактирование всех профилей пользователей доступно администраторам системы;
* размещение объявлений о резервации должно начинаться непосредственно с создания объявления и выбора его типа;
* выбор типа объявления резервации будет осуществляться из следующего списка:

1. резервация, основанная на очереди;
2. резервация, основанная на местоположении;
3. резервация, основанная на перечне объектов или услуг;

* добавление дополнительной информации о резервации доступно только после создания объявления о резервации;
* добавление события к резервации доступно только после создания объявления о резервации;
* работа с категориями объявлений доступна только после создания объявления о резервации и подразумевает использование заранее созданных администраторами категорий;
* работа с типами объектов или услуг резервации доступна только для типа резервации, основанном на перечне объектов или услуг и доступна только после создания объявления о резервации;
* изменение данных резервации, связанных с типами объектов или услуг возможно только в том случае, когда объявление имеет статус «не активно»;
* работа по организации структурных единиц местонахождения доступна только для типа резервации, основанном на местоположении и возможна только после создания объявления о резервации;
* работа с местоположением услуг резервации доступна только после создания объявления о резервации;
* изменение статуса объявления о резервации доступно владельцу объявления и администраторам;
* статистические данные объявления о резервации доступны владельцу объявления и администраторам;
* работа по созданию и управлению категориями размещаемых объявлений о резервации доступна только администраторам;
* управление пользователями, в частности деактивация их аккаунтов, доступна только администраторам;
* функциональность по учету пользователей по резервируемым местам доступна только ответственным лицам резервации, непосредственно связанных с поставщиком резервации;
* просмотр предложений о резервации и резервирование мест доступно зарегистрированным пользователям системы, а также администраторам;
* изменение данных о конкретной резервации доступно владельцу объявления о резервации, администратору, пользователю, оформившему резервацию, а также лицу, ответственному за резервацию;
* возможность создания ролей и управление настройками доступов пользователей доступна только администраторам;

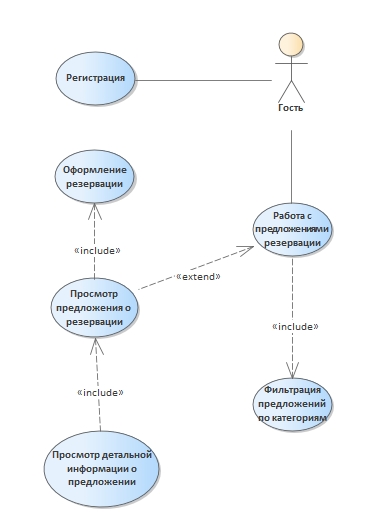


Рисунок 2.4 – Диаграмма вариантов использования для гостя

В произведенной спецификации функциональных требований четко представлено разделение на основные и дополнительные функциональные возможности проектируемого веб-приложения, а также более детальное описание некоторых функций.

# ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

## Разработка архитектуры веб-приложения

Клиент-серверная архитектура является одной из наиболее популярных архитектур для веб-приложений. Эта архитектура заключается в разделении приложения на две части: клиентскую и серверную. Клиентская часть представляет собой интерфейс, через который пользователь взаимодействует с приложением, а серверная часть содержит бизнес-логику приложения и работает с базой данных.

Одним из основных преимуществ клиент-серверной архитектуры является возможность разделения нагрузки между клиентской и серверной частями. Клиентская часть обеспечивает пользовательский интерфейс и отображение данных, а серверная часть занимается бизнес-логикой и хранением данных. Это позволяет снизить нагрузку на сервер, что приводит к более быстрой и стабильной работе приложения.

Кроме того, клиент-серверная архитектура позволяет улучшить безопасность приложения. Бизнес-логика и данные хранятся на сервере, что делает их более защищенными от внешних угроз. Клиентская часть не имеет прямого доступа к данным, что позволяет снизить вероятность несанкционированного доступа к ним. В результате для создания веб-приложения была выбрана именно эта архитектура.

Также необходимо провести анализ необходимой аппаратной конфигурации, на которой будут работать части конечного программного средства, и описать их взаимодействие между собой. Для описания узлов и их связей может использоваться диаграмма развертывания, которая представлена на рисунке 3.1.

На основе упомянутой выше диаграммы были сформированы следующие требования:

* веб-приложение может быть целиком размещено на один физический (либо виртуальный) сервер, так и на различные выделенные сервера;
* программное обеспечение «PostgreSQL Server» устанавливается на отдельный выделенный сервер либо на кластер серверов;
* узлы могут располагаться на отдельных серверах, однако целесообразно всю инфраструктурную часть приложения поместить в локальную сеть;
* для снижения сетевых нагрузок сервер базы данных вынесен в отдельный узел;
* клиент может осуществлять работу с системой по протоколам HTTP и HTTPS. При этом обязательным условием является использования протокола HTTPS при необходимости аутентификации пользователя в системе.
* Далее приведены минимальные рекомендуемые требования к аппаратной платформе. Для работы сервера приложения необходимы:
* процессор уровня Intel Core 5 с поддержкой 64-битной архитектуры и с тактовой частотой 2,2 ГГц (либо более производительный);
* объем ОЗУ – не менее 8 Гбайт;
* сетевой адаптер (допускается встроенный) с пропускной способностью не менее 100 Мбит/сек;
* объем свободного пространства на жестком диске – 255 Гбайт.

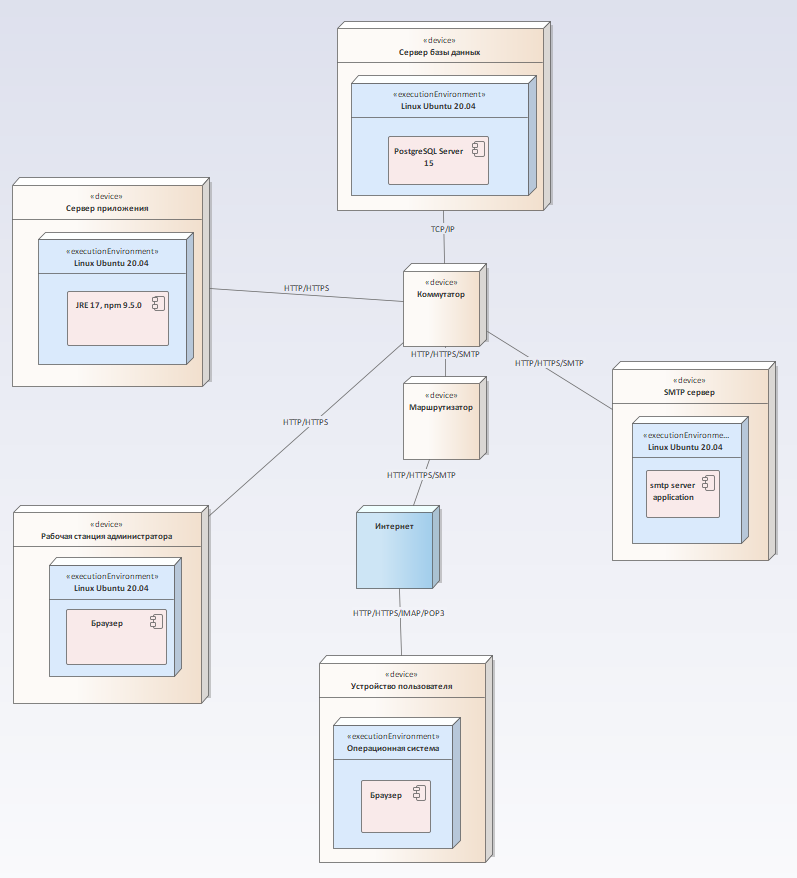


Рисунок 3.1 – Диаграмма развертывания

Для работы сервера базы данных требуются:

* процессор (один или несколько) уровня Intel Core 5 с поддержкой 64-битной архитектуры и с тактовой частотой не менее 2 ГГц (либо более производительный);
* объем ОЗУ – не менее 16 Гбайт;
* два сетевых адаптера (допускаются встроенные), обеспечивающих доступ со скоростью 1 Гбит/сек;
* объем свободного пространства на жестком диске не менее 30 Гбайт для установки СУБД «PostgreSQL» и не менее 200 Гб для размещения файлов данных.

Представленное выше описание дает полное представление об технических характеристиках узлов, на которых будет эксплуатироваться данное программное средство.

## Разработка модели базы данных

Поскольку разрабатываемая система в значительной степени нацелена на работу со значительным объемом данных, а также работу на центральном сервере, то база данных является важным компонентом системы, который позволит предоставлять доступ к программному средству из любой точки с доступом в интернет.

Сущность «roles» содержит информацию о ролях пользователей системы и содержит следующие поля:

* уникальный идентификатор роли;
* название роли.

Сущность «users» содержит информацию о личных данных пользователей системы и содержит следующие поля:

* дата рождения;
* уникальный идентификатор;
* адрес почтового ящика;
* имя;
* фамилия;
* пол;
* пароль;
* логин;
* идентификатор роли.

Сущность «reservations» содержит информацию о резервации пользователя:

* уникальный идентификатор;
* время совершения резервации;
* дата совершения резервации;
* статус резервации;
* идентификатор пользователя;

Сущность «additional\_offer\_info» содержит дополнительную информацию предложения резервации:

* уникальный идентификатор;
* ссылка на местоположение;
* ссылка на событие;
* ссылка на изображение;
* идентификатор предложения резервации.



Рисунок 3.2 – Первая часть схемы базы данных

Сущность «city» содержит информацию о городе:

* уникальный идентификатор;
* название города;
* идентификатор страны.

Сущность «country» содержит информацию о стране:

* уникальный идентификатор;
* название страны.

Сущность «establishment» содержит информацию о местоположении резервации:

* уникальный идентификатор;
* флаг присутствия страны в местоположении;
* идентификатор страны;
* флаг присутствия города в местоположении;
* идентификатор города;
* флаг присутствия улицы в местоположении;
* название улицы;
* флаг присутствия здания в местоположении;
* название здания;
* флаг присутствия апартаментов в местоположении;
* название апартаментов;
* идентификатор пользователя.

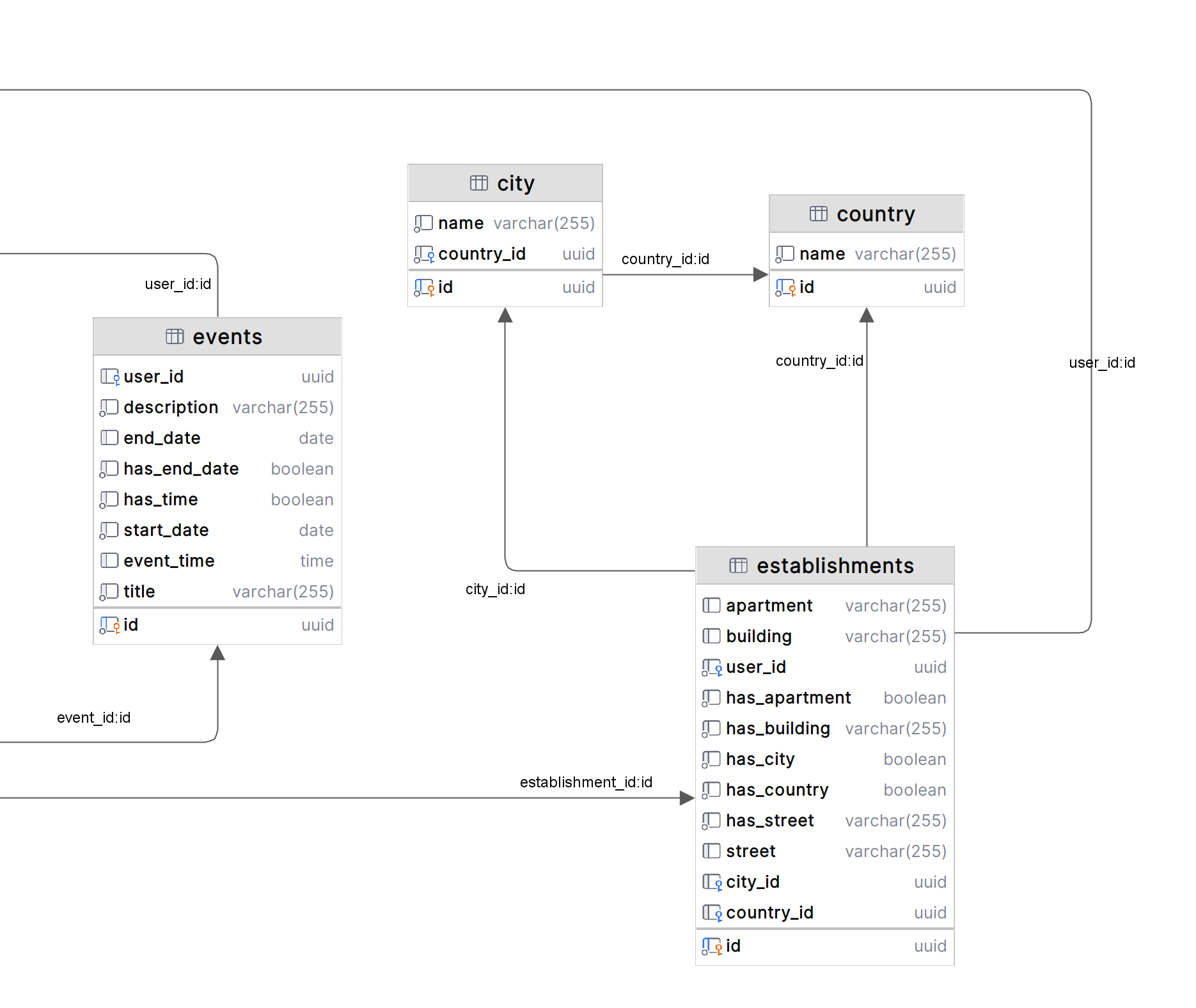


Рисунок 3.3 – Вторая часть схемы базы данных

Сущность «events» содержит информацию о событии:

* уникальный идентификатор;
* идентификатор пользователя;
* описание события;
* дата окончания;
* флаг присутствия даты окончания;
* флаг присутствия времени события;
* дата начала;
* время начала;
* название события.

Сущность «offer\_categories» содержит информацию о категории предложения резервации:

* уникальный идентификатор;
* название категории.

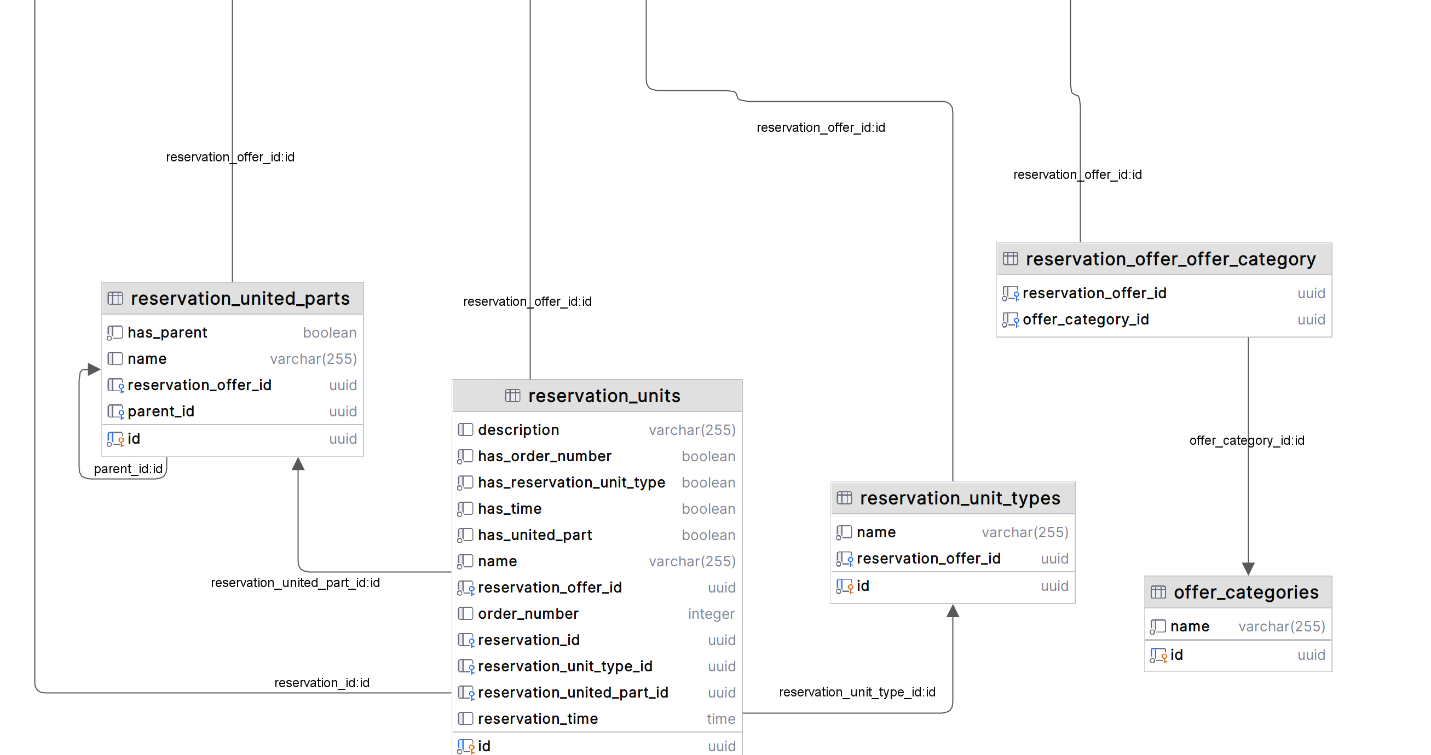


Рисунок 3.4 – Третья часть схемы базы данных

Сущность «reservation\_offer\_offer\_category» представляет собой связь многие ко многим между таблицами категорий объявлений и самими объявлениями:

* идентификатор объявления о резервации;
* идентификатор категории.

Сущность «reservation\_offers» содержит информацию о предложениях резервации:

* уникальный идентификатор;
* идентификатор пользователя;
* идентификатор местоположения;
* идентификатор события;
* флаг присутствия дополнительной информации;
* флаг присутствия местоположения;
* флаг присутствия события;
* флаг присутствия времени;
* название объявления о резервации;
* статус;
* тип;
* время резервации;
* дата резервации;
* тип резервации очереди.

Сущность «reservation\_unit\_types» содержит информацию о типах услуг или объектов, указанных в объявлении:

* уникальный идентификатор;
* наименование типа объекта или услуги;
* идентификатор объявления о резервации.

Сущность «reservation\_united\_parts» содержит информацию об объединенных единицах размещения мест:

* уникальный идентификатор;
* флаг присутствия родительской единицы размещения;
* наименование;
* идентификатор объявления;
* идентификатор родительской единицы.

Сущность «reservation\_units» содержит информацию о местах резервирования:

* уникальный идентификатор;
* описание;
* флаг присутствия порядкового номера;
* флаг присутствия типа объекта или услуги;
* флаг присутствия времени;
* флаг присутствия объединенной единицы;
* наименование;
* идентификатор объявления о резервации;
* порядковый номер;
* идентификатор резервации;
* идентификатор места резервации;
* идентификатор объединенной единицы резервации;
* время резервации.

## Разработка алгоритмов веб-приложения

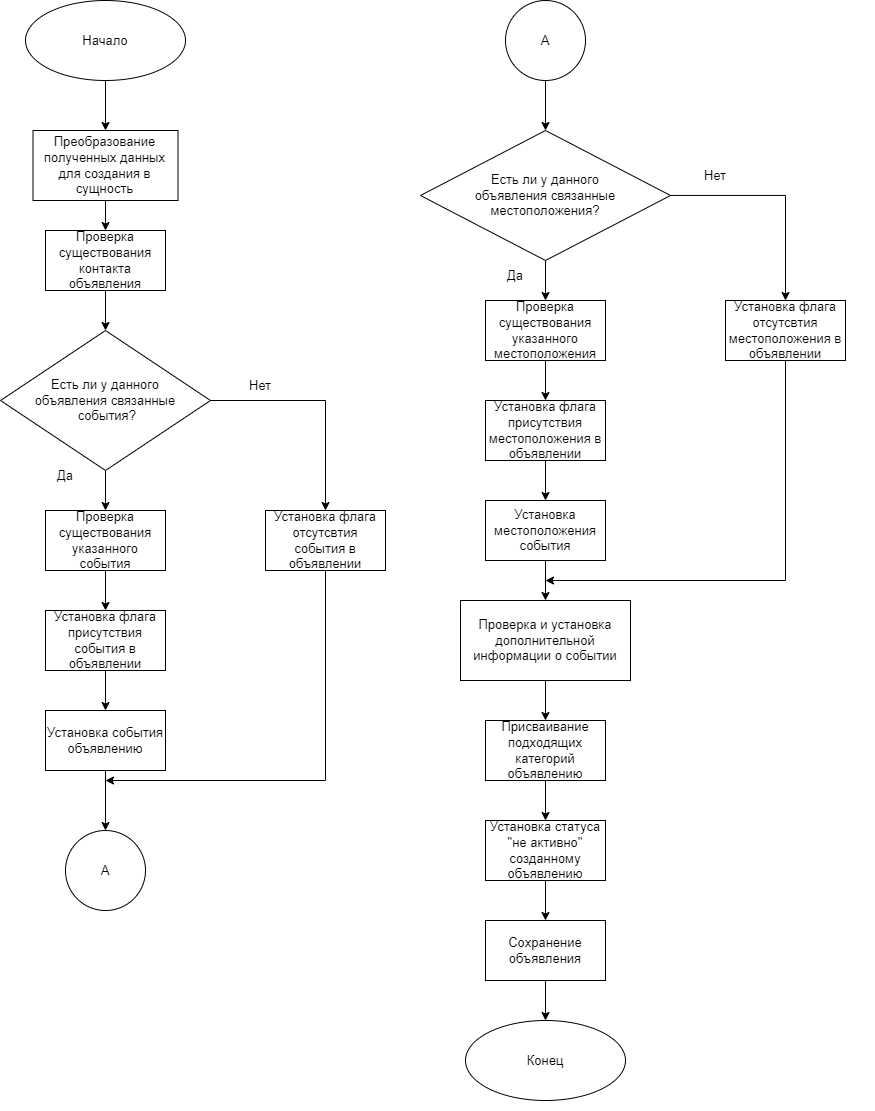


Рисунок 3.3 – Алгоритм создания объявления о резервации

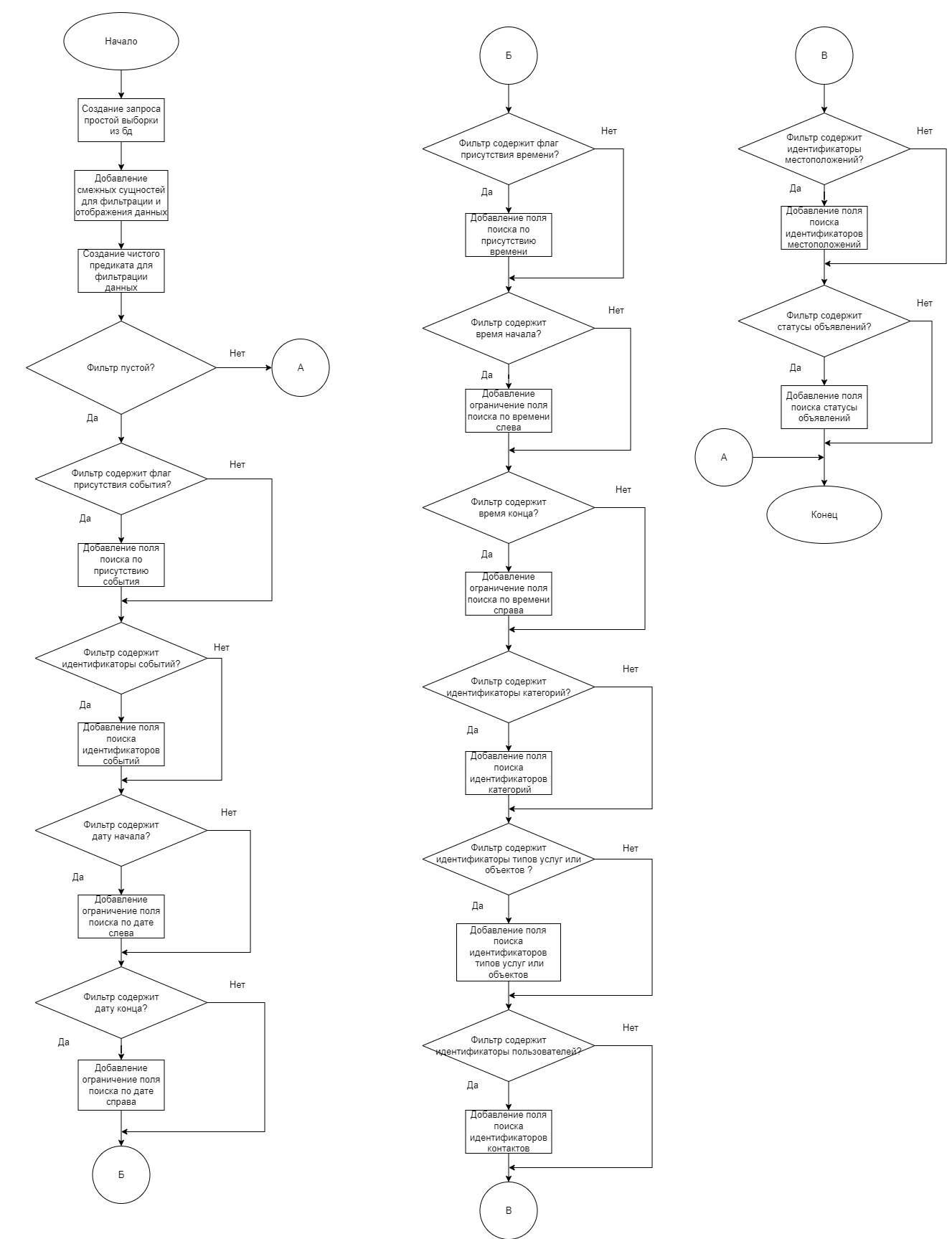


Рисунок 3.4 – Алгоритм создания объекта запроса по данным фильтра



Рисунок 3.5 – Алгоритм создания резервации

# РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ

## Обоснование выбора языка программирования для создания серверной части приложения

Java – один из наиболее распространенных языков программирования в мире, и это не случайно. В частности, он широко используется для разработки веб-приложений благодаря своей универсальности, переносимости и отличной производительности. В данном разделе мы рассмотрим основные преимущества Java в разработке веб-приложений.

Основным преимуществом Java является ее платформенная независимость. Это означает, что приложения, написанные на Java, могут работать на любой операционной системе без изменений. Это достигается за счет того, что Java-приложения выполняются на виртуальной машине Java (JVM), которая переводит байт-код Java в машинный код для конкретной операционной системы. Это также означает, что разработчики могут писать приложения на одной платформе и выпускать их на другой без необходимости переписывать код [8].

Еще одно преимущество Java – это высокая производительность и быстродействие. Java является компилируемым языком программирования, что означает, что код, написанный на Java, преобразуется в машинный код, который может выполняться непосредственно процессором. Благодаря этому Java обладает быстрой скоростью выполнения кода, что делает его идеальным для создания сложных и масштабируемых веб-приложений. Кроме того, Java обладает встроенными средствами оптимизации кода, которые позволяют улучшить производительность приложений.

Высокая степень надежности – очередное преимущество Java. Java предоставляет различные механизмы для обработки ошибок и исключительных ситуаций, что делает приложения на Java более устойчивыми к сбоям и сбоям в работе. Это также облегчает отладку и тестирование приложений, поскольку ошибки можно отслеживать и исправлять в более ранних стадиях разработки.

Кроме того, Java предоставляет различные инструменты и библиотеки для упрощения разработки веб-приложений. Например, библиотека Spring Framework предоставляет широкий спектр инструментов для управления зависимостями, конфигурирования приложений и управления транзакциями, что делает разработку более простой и эффективной. Также существует множество других библиотек и фреймворков, таких как Hibernate, Struts и JSF, которые упрощают разработку веб-приложений на Java.

Наконец, Java также предоставляет широкие возможности для создания масштабируемых и производительных веб-приложений. Благодаря своей архитектуре, Java-приложения могут масштабироваться горизонтально и вертикально, что позволяет поддерживать высокий уровень нагрузки и обеспечивать высокую производительность. Кроме того, Java предоставляет возможности для оптимизации производительности и управления памятью, что делает приложения на Java быстрее и эффективнее.

## Обоснование выбора фреймворка серверной части приложения

Spring Framework – это один из самых популярных фреймворков для разработки веб-приложений на Java. Он предоставляет множество инструментов и компонентов, которые упрощают создание сложных приложений и повышают производительность разработчиков. Ниже перечислены основные преимущества Spring Framework в разработке веб-приложений.

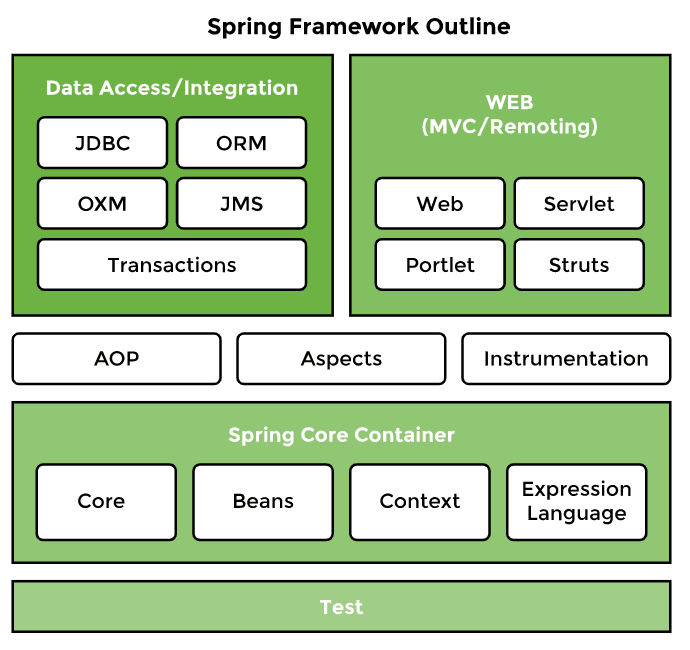


Рисунок 4.1 – Краткое описание компонентов Spring Framework

Инверсия контроля и внедрение зависимостей. Одной из главных особенностей Spring Framework является использование принципов инверсии контроля (IoC) и внедрения зависимостей (DI). Это позволяет разработчикам создавать приложения с легкими и гибкими компонентами, которые могут быть легко заменены или изменены во время выполнения приложения.

Аспектно-ориентированное программирование. Spring Framework также предоставляет поддержку аспектно-ориентированного программирования (AOP), которая позволяет разработчикам создавать модульные и легко поддерживаемые приложения. AOP позволяет выносить повторяющийся код из различных частей приложения и инкапсулировать его в аспекты.

Упрощенный доступ к данным. Spring Framework обеспечивает удобный доступ к данным с помощью инструментов, таких как Spring JDBC и Spring ORM. Они позволяют разработчикам работать с различными источниками данных, включая реляционные базы данных и NoSQL хранилища данных, используя единый и простой интерфейс.

Удобство тестирования. Spring Framework обеспечивает хорошую поддержку тестирования благодаря своей архитектуре. Разработчики могут легко создавать модульные тесты для каждого компонента приложения и выполнять автоматическое тестирование с использованием инструментов, таких как JUnit и Mockito.

Поддержка безопасности. Spring Framework предоставляет множество инструментов для обеспечения безопасности приложения. Он включает в себя механизмы аутентификации и авторизации, поддержку SSL и HTTPS, защиту от атак CSRF и XSS, а также множество других инструментов для обеспечения безопасности приложения.

Перечень компонентов Spring Framework довольно обширен [9]. Он включает в себя следующие компоненты:

Spring Core – ядро фреймворка, которое предоставляет основные функции для управления объектами и управления жизненным циклом компонентов приложения. Сюда входят такие инструменты, как контейнеры внедрения зависимостей, бины Spring и фабрики бинов.

Spring AOP – это компонент, который предоставляет поддержку аспектно-ориентированного программирования. Он позволяет выносить повторяющийся код в аспекты, что упрощает создание модульных приложений.

Spring MVC – это компонент, который предоставляет инструменты для создания веб-приложений на основе паттерна Model-View-Controller (MVC). Он включает в себя контроллеры, представления и модели, которые позволяют разработчикам создавать легко поддерживаемые и гибкие веб-приложения.

Spring Data – это компонент, который предоставляет поддержку работы с различными источниками данных, включая реляционные базы данных и NoSQL хранилища. Он включает в себя инструменты, такие как Spring JDBC, Spring ORM, Spring Data JPA и Spring Data MongoDB.

Spring Security – это компонент, который предоставляет инструменты для обеспечения безопасности приложения. Он включает в себя механизмы аутентификации и авторизации, поддержку SSL и HTTPS, защиту от атак CSRF и XSS, а также множество других инструментов для обеспечения безопасности приложения.

Spring WebFlux – это компонент, который предоставляет инструменты для создания реактивных веб-приложений. Он позволяет разработчикам создавать приложения, которые могут обрабатывать большое количество запросов с низкой задержкой.

Spring Test – это компонент, который предоставляет инструменты для тестирования приложений. Он включает в себя инструменты для создания модульных тестов, интеграционных тестов и тестового окружения.

Это далеко не полный список компонентов Spring Framework, но основные компоненты перечислены выше. Каждый из них предоставляет разработчикам множество инструментов для упрощения создания приложений на Java.

## Обоснование выбора языка программирования и фреймворка для создания пользовательского интерфейса веб-приложения

ReactJS – это библиотека JavaScript с открытым исходным кодом, которая широко используется для разработки пользовательских интерфейсов веб-приложений. ReactJS имеет множество преимуществ, которые делают его одним из наиболее популярных инструментов для разработки современных веб-приложений [10].

Одним из главных преимуществ ReactJS является его компонентный подход к разработке веб-приложений. Компоненты являются фундаментальным строительным блоком в ReactJS, и позволяют разработчикам создавать чрезвычайно масштабируемые и переиспользуемые пользовательские интерфейсы. Вся страница в ReactJS представляет собой дерево компонентов, где каждый компонент может содержать другие компоненты. Такой подход обеспечивает высокую модульность кода и облегчает сопровождение проектов.

Еще одним преимуществом ReactJS является его способность работать в сочетании с другими инструментами и технологиями. ReactJS не привязывается к какому-то конкретному стеку технологий, поэтому он может быть легко интегрирован с различными библиотеками и фреймворками. Например, ReactJS может быть использован совместно с Redux для управления состоянием приложения, с React Native для создания мобильных приложений, а также с серверными фреймворками, такими как Node.js, для разработки полноценных приложений, работающих как на сервере, так и на клиенте.

Кроме того, ReactJS имеет высокую скорость работы благодаря использованию виртуального DOM (Document Object Model). Виртуальный DOM является абстрактным представлением реального DOM, которое ReactJS использует для сравнения и оптимизации изменений в пользовательском интерфейсе. В результате, ReactJS минимизирует количество дорогостоящих операций, таких как перерисовка всей страницы, и ускоряет работу приложения.

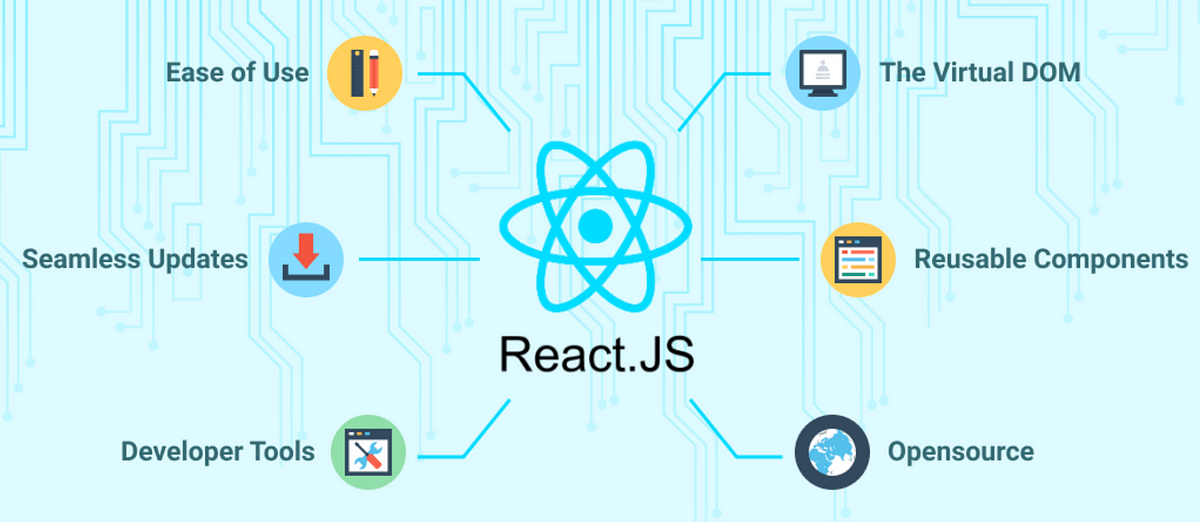


Рисунок 4.2 – Основные преимущества ReactJs

ReactJs также имеет множество удобных инструментов для отладки, включая расширение Chrome DevTools, которое позволяет быстро и легко находить и устранять ошибки. Это может существенно сократить время на отладку приложения и повысить его качество.

В целом, ReactJs предоставляет мощный инструментарий для разработки высококачественных, масштабируемых веб-приложений с повторно используемыми компонентами, что позволяет ускорить процесс разработки, улучшить качество кода и обеспечить простую отладку.

## Обоснование выбора средств разработки

IntelliJ IDEA – это одна из наиболее популярных интегрированных сред разработки (IDE), которая используется для создания приложений на Java. Она предоставляет разработчикам множество инструментов для упрощения процесса разработки, а также для повышения производительности и качества кода. Рассмотрим некоторые из преимуществ IntelliJ IDEA для разработки приложений на Java:

* + удобный и интуитивно понятный интерфейс: IntelliJ IDEA имеет очень удобный и интуитивно понятный интерфейс, который облегчает работу разработчиков. Он позволяет легко настраивать окружение и переключаться между различными проектами;
  + кодогенерация: IntelliJ IDEA предоставляет широкие возможности по генерации кода, что значительно ускоряет процесс разработки приложений. Он может генерировать код для классов, методов, интерфейсов и многого другого, что позволяет разработчикам сосредоточиться на создании более сложных частей приложения;

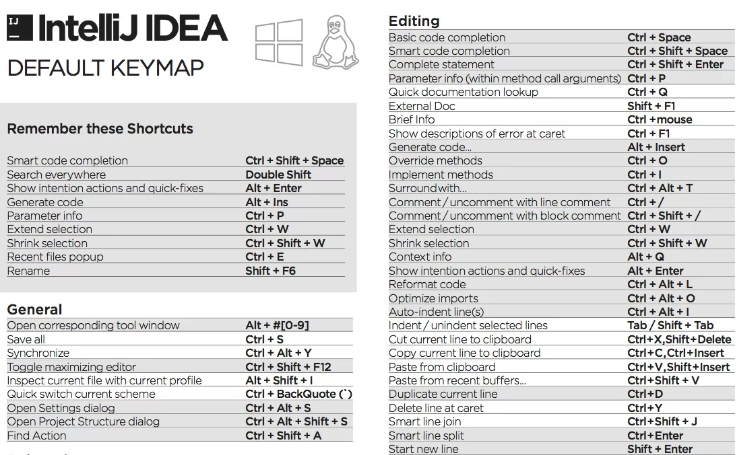


Рисунок 4.3 – Часть списка сочетаний клавиш для быстрого доступа к функционалу Intelij IDEA

* + поддержка различных фреймворков: IntelliJ IDEA обеспечивает поддержку различных фреймворков для разработки приложений на Java. Например, он поддерживает Spring Framework, Hibernate, Struts и многие другие. Это позволяет разработчикам использовать инструменты, которые им наиболее удобны, и создавать приложения более быстро и эффективно;
  + рефакторинг: IntelliJ IDEA предоставляет мощные инструменты для рефакторинга кода, что позволяет разработчикам улучшать качество своего кода и упрощать его структуру. Он может автоматически изменять имена классов, методов и переменных, извлекать методы, создавать интерфейсы и многое другое;
  + интеграция с системами контроля версий: IntelliJ IDEA обеспечивает интеграцию с различными системами контроля версий, такими как Git, SVN и Mercurial. Это позволяет разработчикам легко управлять своими проектами и работать в команде, даже если они находятся в разных местах;
  + отладка IntelliJ IDEA предоставляет мощные инструменты для отладки кода, которые помогают разработчикам быстро находить и устранять ошибки в своем коде. Он может автоматически останавливать выполнение программы на определенных точках, отображать значения переменных в режиме реального времени и многое другое**;**
  + поддержка тестирования IntelliJ IDEA обеспечивает поддержку различных инструментов для тестирования приложений, включая JUnit, TestNG и многие другие. Это позволяет разработчикам легко тестировать свой код и обнаруживать ошибки до того, как они попадут в готовый продукт;
  + поддержка многих языков программирования IntelliJ IDEA поддерживает многие языки программирования, включая Java, Kotlin, Groovy, Scala и многие другие. Это позволяет разработчикам создавать приложения на разных языках программирования и использовать инструменты, которые им наиболее удобны**.**

## Описание взаимодействия серверной и пользовательской части приложения

Для общения между бэкендом и фронтендом используется протокол HTTP. Веб-приложение на Spring Framework представляет собой REST API, которое принимает HTTP запросы от фронтенда и возвращает данные в формате JSON. Фронтенд, в свою очередь, отправляет HTTP запросы на сервер, чтобы получить данные или отправить данные на бэкенд.

HTTP (HyperText Transfer Protocol) – это протокол передачи данных, используемый в Интернете. Он предназначен для передачи гипертекстовых документов, таких как HTML. Одним из главных преимуществ HTTP является его простота, которая позволяет быстро и легко создавать и использовать веб-приложения. Одним из ключевых элементов HTTP являются его методы, которые определяют, какие действия выполняются с ресурсами на сервере.

GET – это метод HTTP-запроса, который используется для получения ресурсов с сервера. Он часто используется для получения HTML-страниц, изображений, видео и других файлов с сервера. Запрос GET может быть выполнен в браузере, в консоли или с помощью программы-клиента. GET-запрос может содержать параметры запроса, которые передаются в строке запроса URL [11].

POST – это метод HTTP-запроса, который используется для отправки данных на сервер для обработки. Он может использоваться для отправки форм, комментариев, изображений и других типов данных на сервер. POST-запросы могут содержать тело запроса, которое содержит данные, отправляемые на сервер.

PUT – это метод HTTP-запроса, который используется для обновления существующих ресурсов на сервере. PUT-запрос может содержать тело запроса, которое содержит обновленные данные ресурса.

DELETE – это метод HTTP-запроса, который используется для удаления ресурсов на сервере. Он может быть использован для удаления файлов, комментариев, записей и других ресурсов на сервере.

PATCH – это метод HTTP-запроса, который используется для частичного обновления ресурсов на сервере. Он может использоваться для обновления только некоторых полей ресурса вместо того, чтобы отправлять полный набор данных ресурса.

OPTIONS – это метод HTTP-запроса, который используется для определения опций и возможностей, доступных для ресурса на сервере. Он может быть использован для определения доступных методов запросов, допустимых заголовков запроса и других параметров для взаимодействия с ресурсом.

HEAD – это метод HTTP-запроса, который аналогичен методу GET, но не содержит тела ответа. Он может быть использован для получения метаданных ресурса, таких как дата изменения, размер файла и другие характеристики, без необходимости получения полного содержимого ресурса.

Для отправки HTTP запросов на сервер из фронтенда используется объект XMLHttpRequest (XHR) или fetch API. Объект XHR используется для отправки асинхронных запросов на сервер, а fetch API является более современным API для работы с HTTP запросами.

Для отправки запросов на сервер, фронтенд должен знать адрес эндпоинта на сервере. Это может быть либо IP-адрес, либо доменное имя сервера. После того, как адрес эндпоинта был получен, фронтенд может отправить запрос на сервер, указав метод HTTP запроса, путь к эндпоинту и, возможно, тело запроса.

Как правило, на сервере Spring Framework для обработки HTTP запросов используются контроллеры, которые обрабатывают запросы и возвращают данные. Контроллеры могут принимать параметры из URL-адреса запроса, параметры запроса или тело запроса в формате JSON.

При получении HTTP запроса на сервере, Spring Framework выполняет обработку запроса и создает объект модели данных, который затем возвращается в формате JSON. Фронтенд может использовать эти данные для отображения информации на странице или для выполнения каких-то действий, например, добавления, удаления или обновления данных.

# ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ РЕЗЕРВИРОВАНИЕМ МЕСТ НА ЯЗЫКЕ JAVA

## Характеристика разработанного по индивидуальному заказу веб-приложения

Веб-приложение предназначено для предоставления услуг по резервированию мест. Оно разрабатывается организацией-разработчиком по индивидуальному заказу организации-заказчика.

Веб-приложение поддерживает следующие функции:

* размещение резервируемых мест;
* резервирование доступных мест;
* учет порядочности пользователя;
* рейтинг заведений резервируемых мест.

Веб-приложение построено на клиент-серверной архитектуре. Разработка была осуществлена на языке программирования Java с использованием фреймворка Spring в интегрированной среде разработки Intelij IDEA.

Потенциальными пользователями данного приложения являются владельцы ресторанов, кинотеатров, концертных залов, а также их посетители.

## Расчет затрат на разработку и цена веб-приложения, созданного по индивидуальному заказу

Расчет затрат на основную заработную плату команды разработчиков осуществляется исходя из состава и численности команды, размера месячной заработной платы каждого участника команды, а также трудоемкости работ, выполняемых при разработке веб-приложения отдельными исполнителями по формуле 7.1:

, (7.1)

где n– количество исполнителей;

Кпр – коэффициент премий (1,35);

Зч*i* – часовая заработная плата *i*-го исполнителя, руб.;

ti– трудоемкость работ, выполняемых *i*-м исполнителем, ч.

Согласно постановлению министерства труда и соцзащиты, на 2023-й год для пятидневной рабочей недели с выходными днями в субботу и воскресенье установлена расчетная норма рабочего времени 2011 часов. Исходя из этого количество рабочих часов в месяце принято равным 169,58 часам. Данные по заработной плате команды разработчиков предоставлены организацией на 23 марта 2023 г. Результаты расчетов приведены в таблице 4.1.

Таблица 4.1 ‒ Расчет затрат на основную заработную плату команды

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование должности разработчика | Вид выполняемой  Работы | Месячная заработная плата, руб. | Часовая заработная плата, руб. | Трудоёмкость  работ, ч | Итого, р. |
| Бизнес-аналитик | Анализ требований, руководство проектом | 3700 | 21,82 | 113 | 2465,7 |
| Системный архитектор | Руководство процессом разработки, разработка программного модуля | 3300 | 19,46 | 339 | 6596,9 |
| Инженер-программист | Разработка программного модуля | 1600 | 9,43 | 339 | 3196,7 |
| Тестировщик | Тестирование программного модуля | 1300 | 7,66 | 169 | 1294,5 |
| Итого | | | | | 13553,8 |
| Премия (35%) | | | | | 4743,8 |
| Всего затраты на основную заработную плату сотрудников | | | | | 18297,6 |

Дополнительная заработная плата (Зд) определяется по формуле:

, (7.2)

где Зо – основная заработная плата, руб.;

Нд – норматив дополнительной заработной платы (13%).

Воспользовавшись формулой 7.2, получим затраты на дополнительную заработную плату:

Отчисления на социальные нужды (в фонд социальной защиты населения и на обязательное страхование) в соответствии с действующим законодательством по формуле:

, (7.3)

где Нсоц – норматив отчислений от фонда оплаты труда (34,6%).

Прочие расходы определяются по следующей формуле

, (7.4)

где Нпр – норматив прочих расходов (33%).

Тогда, если подставить полученные ранее значения в формулу 7.4, получим:

Общая сумма затрат на разработку веб-приложения рассчитывается по формуле:

. (7.5)

Воспользовавшись формулой 7.5, получим общую сумму затрат на разработку веб-приложения:

руб.

Плановая прибыль, включаемая в цену веб-приложения, определяется по следующей формуле:

, (7.6)

где Зр – затраты на разработку веб-приложения, руб.;

Рп.с – рентабельность затрат на разработку веб-приложения (32%).

Тогда, используя формулу 7.6, можно рассчитать прибыль организации-разработчика до налогообложения:

Определить отпускную цену веб-приложения для заказчика можно по формуле:

Ц = Зр + Пп.с + НДС, (7.7)

где Зр – затраты на разработку веб-приложения, руб.;

Пп.с – плановая прибыль, руб.;

НДС – сумма налога на добавленную стоимость, руб.

Сумма налога на добавленную стоимость при таком подходе к ценообразованию рассчитывается по формуле 7.8:

, (7.8)

где Зр – затраты на разработку веб-приложения, руб.;

Пп.с – прибыль, руб.;

Ндс – ставка налога на добавленную стоимость согласно действующему законодательству (20%).

Подставив полученные значения в формулу 7.7, можно рассчитать отпускную цену веб-приложения с НДС для заказчика:

Ц = 33868,5 + 10837,9 + 8941,3 = 53647,7 руб.

Результаты расчетов представлены в таблице 7.2.

Таблица 4.2 ‒ Формирование цены веб-приложения на основе затрат

|  |  |
| --- | --- |
| Наименование статьи затрат | Значение, р. |
| 1. Основная заработная плата разработчиков | 18297,6 |
| 2. Дополнительная заработная плата разработчиков | 2378,8 |
| 3. Отчисления на социальные нужды | 7154 |
| 4. Прочие расходы | 6038,2 |
| 5. Общая сумма затрат на разработку | 33868,5 |
| 6. Плановая прибыль, включаемая в цену  веб-приложения | 10837,9 | |
| 7. Отпускная цена веб-приложения без НДС | 44706,4 | |
| 8. НДС | 8941,3 | |
| 9. Отпускная цена веб-приложения с НДС | 53647,7 | |

## Расчет результата от разработки и продажи веб-приложения, созданного по индивидуальному заказу

Для организации-разработчика экономическим эффектом является прирост чистой прибыли, полученной от разработки и реализации приложения заказчику. Так как веб-приложение будет реализовываться организацией-разработчиком по отпускной цене, сформированной на основе затрат на разработку, то экономический эффект, полученный организацией-разработчиком, в виде прироста чистой прибыли от его разработки, определяется по формуле:

, (7.9)

где Пп.с – прибыль, включаемая в цену веб-приложения, руб.

Тогда чистая прибыль организации-разработчика будет равна:

## Расчет показателей экономической эффективности разработки веб-приложения

Для организации-разработчика веб-приложения оценка экономической эффективности разработки осуществляется с помощью расчёта простой нормы прибыли по формуле:

(7.10)

где – прирост чистой прибыли, полученной от разработки веб-

приложения организацией-разработчиком;

– затраты на разработку веб-приложения.

Исходя из формулы (7.10) рентабельность инвестиций равна:

По результатам расчётов, затраты на разработку веб-приложения составили 33868,5 руб., прирост чистой прибыли от разработки веб-приложения составляет 8670,3 руб., а величина рентабельности равна 26%.

# Тестирование веб-приложения и анализ полученных результатов

## Методы тестирования

Существует множество различных методов и подходов к тестированию программного обеспечения, каждый из которых имеет свои преимущества и недостатки [12].

Модульное тестирование – это тестирование, которое проводится на уровне отдельных модулей программы, чтобы убедиться в их правильной работе в изоляции от других модулей. Этот подход позволяет выявлять ошибки на ранней стадии разработки и ускоряет процесс отладки.

Интеграционное тестирование – этот подход используется для проверки работы различных модулей программы вместе. Он направлен на выявление ошибок во взаимодействии между модулями и общей работоспособности системы.

Системное тестирование – этот вид тестирования проводится на уровне всей системы и включает в себя проверку всех функций и возможностей программы. Он позволяет убедиться в соответствии системы требованиям и оценить ее работоспособность в различных сценариях использования.

Приемочное тестирование – это тестирование, которое проводится после завершения разработки и позволяет заказчику проверить, соответствует ли продукт их требованиям. Это также может включать тестирование производительности и нагрузочное тестирование.

Регрессионное тестирование – этот вид тестирования проводится после каждого изменения в коде программы и направлен на выявление ошибок, которые могут возникнуть из-за изменений в других частях системы.

Тестирование безопасности – вид тестирования, который проверяет устойчивость системы к взлому и другим видам кибератак.

Тестирование производительности – вид тестирования, который проводится для оценки скорости и масштабируемости программы. Он позволяет определить, как система работает при различных нагрузках и выявить узкие места в производительности.

Нагрузочное тестирования – проверка производительности продукта, которая позволяет определить, как он будет работать при больших нагрузках.

Функциональное тестирование:

* тестирование черного ящика – проверка функциональности продукта без знания его внутренней структуры и алгоритмов;
* тестирование белого ящика – проверка функциональности продукта с знанием его внутренней структуры и алгоритмов;
* тестирование пользовательского интерфейса – проверка удобства использования и соответствия дизайна требованиям пользователей;
* тестирование совместимости – проверка совместимости продукта с другими программами и оборудованием.

Это только несколько примеров из множества различных методов тестирования, которые используются в индустрии. В зависимости от типа проекта и требований заказчика, могут использоваться различные комбинации этих методов, а также другие методы и подходы к тестированию.

## Сценарии тестирования программного средства

Таблица 5.1 – Сценарии тестирования программного средства

|  |  |  |
| --- | --- | --- |
| Описание теста | Ожидаемый результат | Результат |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу регистрации.  4 Ввести корректные данные пользователя.  5 Отправить данные. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница регистрации.  4 Все поля заполнились.  5 Регистрация пройдена успешно. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу регистрации.  4 Ввести некорректные данные пользователя.  5 Отправить данные. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница регистрации.  4 Все поля заполнились.  5 Отображение ошибки регистрации пользователя с неверно введенными данными. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу авторизации.  4 Ввести корректные пароль и логин пользователя.  5 Отправить данные. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница авторизации.  4 Все поля заполнились.  5 Авторизация пройдена успешно. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу авторизации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница авторизации.  4 Все поля заполнились. | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 4 Ввести некорректные логин и пароль.  5 Отправить данные. | 5 Отображение ошибки авторизации пользователя. |  |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу авторизации.  4 Ввести корректные данные пользователя.  5 Отправить данные. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница авторизации.  4 Все поля заполнились.  5 Регистрация пройдена успешно. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу регистрации.  4 Ввести данные существующего в системе пользователя.  5 Отправить данные. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница регистрации.  4 Все поля заполнились.  5 Отображение ошибки регистрации пользователя о том, что такой пользователь уже существует. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу авторизации.  4 Авторизоваться. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница авторизации.  4 Все поля заполнились.  5 Авторизация пройдена успешно. Открылась страница личного кабинета с информацией о пользователе. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу авторизации.  4 Авторизоваться. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница авторизации.  4 Авторизация пройдена успешно. | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 5 Отредактировать личные данные некорректной информацией.  6 Подтвердить изменения. | 5 Отображение ошибки о некорректно введенных данных. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу авторизации.  4 Авторизоваться.  5 Перейти на страницу с предложениями о резервации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница авторизации.  4 Все поля заполнились.  5 Авторизация пройдена успешно.  6 Отображение страницы со списком доступных резерваций в порядке убывания по дате размещения. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу авторизации.  4 Авторизоваться.  5 Перейти на страницу с предложениями о резервации.  6 Выбор данных для фильтрации каталога контента.  7 К списку отображаемого контента применить фильтрацию. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница авторизации.  4 Все поля заполнились.  5 Авторизация пройдена успешно.  6 Отображение страницы со списком доступных резерваций в порядке убывания по дате размещения.  7 Отображение отфильтрованных данных о доступных резервациях. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/).  3 Перейти на страницу авторизации.  4 Авторизоваться. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница авторизации.  4 Все поля заполнились. | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 5 Перейти на страницу с предложениями о резервации.  6 Выбор данных для фильтрации каталога контента.  7 К списку отображаемого контента применить фильтрацию.  8 Выбрать конкретное предложение о резервации для детального просмотра. | 5 Авторизация пройдена успешно.  6 Отображение страницы со списком доступных резерваций в порядке убывания по дате размещения.  7 Отображение отфильтрованных данных о доступных резервациях.  8 Отображение страницы детальной информации предложения о резервации. |  |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с предложениями о резервации.  4 Выбрать конкретное предложение о резервации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком доступных резерваций в порядке убывания по дате размещения.  5 Отображение детальной информации о предложении о резервации. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений о резервациях в порядке убывания по дате размещения. | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Выбор конкретного объявления. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений в порядке убывания по дате размещения.  5 Отображение детальной информации о размещенном объявлении. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Выбор конкретного объявления.  5 Удаление объявления. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений в порядке убывания по дате размещения.  5 Отображение детальной информации о размещенном объявлении.  6 Удаление объявления о резервации.  7. Отображение страницы со списком размещенных объявлений в порядке убывания по дате размещения. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 4 Нажать кнопку создания нового объявления о резервации.  5 Ввести корректные данные о резервации.  6 Подтвердить отправку введенных в форму данных. | объявлений в порядке убывания по дате размещения.  5 Отображение окна создания объявления о резервации.  6 Все необходимые поля заполнены.  7 Отображение страницы созданного объявления о резервации. |  |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Нажать кнопку создания нового объявления о резервации.  5 Ввести некорректные данные о резервации.  6 Подтвердить отправку введенных в форму данных. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных  объявлений в порядке убывания по дате размещения.  5 Отображение окна создания объявления о резервации.  6 Все поля заполнены некорректными данными.  7 Отображение окна создания объявления о резервации с ошибкой о неверном вводе данных. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Нажать кнопку создания нового объявления о резервации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений в порядке убывания по дате размещения. | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Нажать кнопку создания нового объявления о резервации.  5 Ввести корректные данные о резервации.  6 Подтвердить отправку введенных в форму данных.  7 Нажать на кнопку добавления события.  8 Ввод корректных данных о событии.  9 Нажатие кнопки подтверждения данных о добавлении события. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений в порядке убывания по дате размещения.  5 Отображение окна создания объявления о резервации.  6 Все необходимые поля заполнены.  7 Отображение страницы созданного объявления о резервации.  8 Отображения окна добавления события.  9 Все поля необходимые заполнены.  10 Отображение страницы редактированного объявления о резервации. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Нажать кнопку создания нового объявления о резервации.  5 Ввести некорректные данные о резервации.  6 Подтвердить отправку введенных в форму данных. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений в порядке убывания по дате размещения.  5 Отображение окна создания объявления о резервации. | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 7 Нажать на кнопку добавления события.  8 Ввод корректных данных о событии.  9 Нажатие кнопки подтверждения данных о добавлении события. | 6 Все необходимые поля заполнены некорректными данными.  7 Отображение сообщения о неверно введенных данных. |  |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Нажать кнопку создания нового объявления о резервации.  5 Ввести корректные данные о резервации.  6 Подтвердить отправку введенных в форму данных.  7 Нажать на кнопку добавления местоположения.  8 Ввод корректных данных о местоположении.  9 Нажатие кнопки подтверждения данных о добавлении местоположении. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений в порядке убывания по дате размещения.  5 Отображение окна создания объявления о резервации.  6 Все необходимые поля заполнены.  7 Отображение страницы созданного объявления о резервации.  8 Отображения окна добавления местоположения.  9 Все поля необходимые заполнены.  10 Отображение страницы редактированного объявления о резервации. | Тест пройден успешно |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 4 Нажать кнопку создания нового объявления о резервации.  5 Ввести корректные данные о резервации.  6 Подтвердить отправку введенных в форму данных.  7 Нажать на кнопку добавления местоположения.  8 Ввод некорректных данных о местоположении.  9 Нажатие кнопки подтверждения данных о добавлении местоположении. | объявлений в порядке убывания по дате размещения.  5 Отображение окна создания объявления о резервации.  6 Все необходимые поля заполнены.  7 Отображение страницы созданного объявления о резервации.  8 Отображения окна добавления местоположения.  9 Все поля необходимые заполнены некорректными данными.  10 Отображение ошибки о вводе некорректных данных о местоположении. |  |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Нажать кнопку создания нового объявления о резервации.  5 Ввести корректные данные о резервации.  6 Подтвердить отправку введенных в форму данных.  7 Нажать на кнопку добавления дополнительной информации о резервации.  8 Ввод корректных данных о резервации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений в порядке убывания по дате размещения.  5 Отображение окна создания объявления о резервации.  6 Все необходимые поля заполнены.  7 Отображение страницы созданного объявления о резервации. | Тест пройден успешно |

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 9 Нажатие кнопки подтверждения данных о добавлении дополнительной информации. | 8 Отображения окна добавления дополнительной информации.  9 Все поля необходимые заполнены.  10 Отображение страницы редактированного объявления о резервации |  |
| 1 Открыть веб-браузер.  2 Перейти по адресу [http://localhost:3000/](http://localhost:300/) как авторизованный пользователь.  3 Перейти на страницу с размещенными предложениями о резервации.  4 Нажать кнопку создания нового объявления о резервации.  5 Ввести корректные данные о резервации.  6 Подтвердить отправку введенных в форму данных.  7 Нажать на кнопку добавления дополнительной информации о резервации.  8 Ввод некорректных данных о резервации.  9 Нажатие кнопки подтверждения данных о добавлении дополнительной информации. | 1 Открылся веб-браузер.  2 Открылась главная страница.  3 Открылась страница личного кабинета.  4 Отображение страницы со списком размещенных объявлений.  5 Отображение окна создания объявления о резервации.  6 Все необходимые поля заполнены.  7 Отображение страницы созданного объявления о резервации.  8 Отображения окна добавления дополнительной информации.  9 Все поля необходимые заполнены некорректными данными.  10 Отображение ошибки о вводе некорректных данных | Тест пройден успешно |

Все сценарии тестирования были воспроизведены посредством ручного тестирования. В ходе их проведения, программное средство отработало корректно и выдало ожидаемые результаты. Таким образом, этап тестирования программного средства можно считать успешным.

# Список использованных источников

1. Системы бронирования и резервирования [Электронный ресурс]. – Режим доступа: <https://studfile.net/preview/2779807/page:6/>. – Дата доступа: 06.04.2023.
2. Компьютерные системы резервирования [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Computer_reservation_system>. – Дата доступа: 06.04.2023.
3. Компьютерные системы распределенных запросов [Электронный ресурс]. – Режим доступа: <https://en.wikipedia.org/wiki/Global_distribution_system>. – Дата доступа: 06.04.2023.
4. Талон.бай [Электронный ресурс]. – Режим доступа: <https://talon.by/>. – Дата доступа: 10.04.2023.
5. Relax.by [Электронный ресурс]. – Режим доступа: <https://www.relax.by/cat/ent/karaoke/>. – Дата доступа: 10.04.2023.
6. 24-я городская поликлиника спецмедосмотров [Электронный ресурс]. – Режим доступа: <https://24gp.by/услуги/заказ-талона-онлайн>. – Дата доступа: 10.04.2023.
7. Понятие UML-диаграмм [Электронный ресурс]. – Режим доступа: https://www.omg.org/spec/UML/. – Дата доступа: 15.04.2023.
8. Особенности Java [Электронный ресурс]. – Режим доступа: https://dev.java/. – Дата доступа: 20.04.2023.
9. Spring Framework [Электронный ресурс]. – Режим доступа: https://spring.io/. – Дата доступа: 20.04.2023.
10. ReactJs [Электронный ресурс]. – Режим доступа: https://react.dev/. – Дата доступа: 20.04.2023.
11. Hyper Text Transfer Protocol [Электронный ресурс]. – Режим доступа: https://datatracker.ietf.org/doc/html/rfc2616. – Дата доступа: 20.04.2023.
12. International Software Testing Qualifications Board [Электронный ресурс]. – Режим доступа: https://www.istqb.org/. – Дата доступа: 25.04.2023.