**4. РЕАЛИЗАЦИЯ**

**4.1 Выбор инструментальных средств, системного и дополнительного программного обеспечения**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

Разраб.

Панкрат М.О.

Провер.

Соколова А.С.

..

Реценз.

Н. Контр.

Соколова А.С.

Утверд.

Казаков В.Е.

РЕАЛИЗАЦИЯ

Лит.

Листов

УО «ВГТУ» каф. ИСиТ гр. Ит-10

Приложение, разрабатывающееся в рамках дипломного проектирования, состоит из двух частей: бэкэнд и фронтэнд.

Для разработки бэкенд части информационной системы был выбран язык JavaScript с использованием платформы Node.js. Node.js - это современная, мощная и высокопроизводительная среда выполнения JavaScript, позволяющая разрабатывать серверные приложения. Node.js пользуется широкой популярностью благодаря своей эффективности, масштабируемости и гибкости.

Node.js обладает множеством преимуществ, которые делают его привлекательным выбором для разработки бэкенда. Он основан на событийно-ориентированной архитектуре, что позволяет эффективно обрабатывать большое количество запросов одновременно. Благодаря асинхронному вводу/выводу, Node.js способен обеспечить высокую производительность, что особенно важно для веб-приложений с большим трафиком.

Одной из ключевых особенностей Node.js является возможность использования JavaScript как на стороне сервера, так и на стороне клиента. Это обеспечивает единый язык программирования как для фронтенда, так и для бэкенда, что упрощает разработку и обслуживание приложений.

Node.js также предоставляет разработчикам широкий выбор библиотек и фреймворков для создания различных типов приложений. Например, Express.js - это популярный фреймворк, который облегчает создание веб-приложений и веб-сервисов. С его помощью разработчики могут легко

определять маршруты, обрабатывать запросы и управлять состоянием приложения.

Node.js поддерживает асинхронные операции и неблокирующий ввод/вывод, что делает его идеальным выбором для создания высоконагруженных приложений, работающих в реальном времени.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

Использование Node.js также обеспечивает разработчикам доступ к огромному экосистему инструментов и библиотек, что упрощает процесс разработки и расширяет возможности приложений.

Фронтэед часть разработана при помощи JavaScript-библтотеки React

React — библиотека, написанная на JavaScript, которая используется для

работы с интерфейсами. В 2011 году её начали использовать для социальной сети Facebook, а уже в 2013 году библиотеку выложили в открытый доступ и энтузиасты со всего мира начали создавать инструменты для расширения её возможностей. Разработчики используют React, чтобы создавать интерфейсы, которые способны менять контент без перезагрузки страницы. Благодаря этому сайты или нативные приложения быстро отзываются на действия пользователей. Можно добавлять товары в корзину без перезагрузки страницы или заполнять формы без переадресации.  
React предпочтительнее использовать по следующим причинам:

* технология SPA (single-page application, по-русски: “разработка одностраничных приложений”): React поможет вам создать одностраничное приложение. С помощью ReactJS вы сможете изменять (управлять/манипулировать) контент всей страницы с минимальным кодом;
* декларативный подход: React использует декларативный дизайн со всем синтаксическим сахаром, что помогает написать поддерживаемый код высокого уровня. Вам просто нужно определить цель, и React будет обрабатывать инструкции JavaScript DOM с учетом ситуаций, в которых они используются;
* компонент-управляемый пользовательский интерфейс: React основан на компонентной концепции. Компоненты являются многоразовыми строительными блоками в пользовательском интерфейсе. С помощью ReactJS вы можете создать инкапсулированный компонент, который управляет своими данными, и избежать сценария влияния на другие состояния и действия компонентов в DOM-дереве. Это только одна из характеристик компонент-управляемого интерфейса. Он также помогает повторно использовать код, разделять ответственность и избегать повторения.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

Код для фронтэнд бэкэнд части приложения написан при помощи редактора кода Visual Studio Code.

Visual Studio Code - это «бесплатный редактор, который помогает программисту писать код, помогает в отладке и исправлении кода с помощью метода intelli-sense». В обычных условиях это облегчает пользователю написание кода простым способом. Многие говорят, что это половина IDE и редактора, но решение остается за программистами. Visual Studio Code поддерживает несколько языков программирования, имеет кроссплатформенную поддержку, огромное количество расширений для упрощения создания кода, а также встроенную поддержку системы контроля версий Git.

Visual Studio Code поддерживает несколько языков программирования. Так что раньше программистам требовалась веб-поддержка: другой редактор для разных языков, но он имеет встроенную многоязычную поддержку. Это также означает, что он легко обнаруживает, если есть какая-либо ошибка или ссылка на другой язык, он сможет легко обнаружить её.

**4.2 Описание реализации вариантов использования**

В данном разделе будет описан процесс и результаты реализации спроектированных вариантов использования программной системы (ПС).

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

В качестве наиболее интересных с точки зрения реализации вариантов использования выбраны:

* Регистрация пользователя: Проверка корректности ввода данных и отправка запроса на сервер для создания нового пользователя.
* Авторизация пользователя: Процесс аутентификации пользователя и управление сессией.

Для администратора:

* Добавление нового автомобиля: Проверка корректности введенных данных и взаимодействие с сервером для добавления автомобиля.

**Регистрация пользователя**

Процесс регистрации пользователя включает в себя несколько этапов:

1. Сбор данных пользователя: Пользователь вводит свои данные (имя, фамилию, телефон, email и пароль) в регистрационную форму.
2. Валидация данных на клиентской стороне: Проверка на заполненность всех полей формы и корректность введенных данных.
3. Отправка данных на сервер: Если все данные корректны, они отправляются на сервер с помощью POST-запроса.
4. Валидация данных на серверной стороне.
5. Сохранение данных в базе данных: Если данные прошли все проверки, создается новый пользователь в базе данных.

Реализация на клиентской стороне:

Листинг 4.1 – Функция регистрации пользователя

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

const register = () => {

    if (

      !firstNameUserReg ||

      !lastNameUserReg ||

      !phoneUserReg ||

      !emailUserReg ||

      !passwordUserReg

    ) {

      toastMessage("Пожалуйста, заполните все поля формы.");

      return;

    }

    // Проверка наличия @

    if (!emailUserReg.includes("@")) {

      toastMessage(

        "Адрес электронной почты должен содержать символ «@».",

        "error"

      );

      return;

    }

    // Проверка корректности email

    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    if (!emailRegex.test(emailUserReg)) {

      toastMessage("Некорректный email.", "error");

      return;

    }

    Axios.post("http://localhost:3307/registration", {

      firstName: firstNameUserReg,

      lastName: lastNameUserReg,

      phone: phoneUserReg,

      email: emailUserReg,

      password: passwordUserReg,

    }).then((response) => {

      const user = response.data;

      console.log(response);

      authContext.authUserChangeHandler(user);

    });

    navigate("/home");

    toastMessage("Добро пожаловать к нам! Спасибо что выбрали Нас!");

  };

Реализация на серверной стороне:

Листинг 4.2 – Функция регистрации пользователя на стороне сервера

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

 app.post("/registration", (req, res) => {

    const { firstName, lastName, phone, email, password, role, deleteStatus } = req.body;

    db.query(

        "INSERT INTO users (firstName, lastName, email, phone, password, role, deleteStatus) VALUES (?, ?, ?, ?, ?, 'user', 'Активный')",

        [firstName, lastName, email, phone, password],

        (err, result) => {

            if (err) {

                console.error("Error while registering user:", err);

                return res.status(500).send("Internal server error");

            }

            // Проверяем, была ли операция вставки успешной

            if (result.affectedRows > 0) {

                // Если да, возвращаем данные зарегистрированного пользователя

                const user = {

                    user\_id: result.insertId,

                    firstName,

                    role: 'user',

                    deleteStatus: 'Активный'

                };

                res.send(user);

            } else {

                // В случае неудачи возвращаем сообщение об ошибке

                res.status(500).send("Error while registering user");

            }

        }

    );

});

**Авторизация пользователя**

Процесс авторизации пользователя включает в себя следующие этапы:

* Сбор данных пользователя: Пользователь вводит телефон и пароль в форму входа.
* Отправка данных на сервер: Данные отправляются на сервер для проверки.
* Проверка данных на сервере: Сервер проверяет наличие пользователя с указанным телефоном и корректность пароля.
* Управление сессией: При успешной авторизации создается сессия пользователя.

Реализация на клиентской стороне:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

Листинг 4.3 – Функция авторизации пользователя

const login = (event) => {

    event.preventDefault();

    Axios.post("http://localhost:3307/login", {

      phone: phoneUser,

      password: passwordUser,

    }).then((response) => {

      if (response.data.message) {

        toastMessage("Некорректные данные", "error");

        return;

      } else {

        const user = response.data;

        if (user.deleteStatus === "Удален") {

          toastMessage("Ошибка доступа.", "error");

        } else {

          toastMessage("Вы успешно вошли в аккаунт!");

          authContext.authUserChangeHandler(user);

        }

      }

    });

  };

Реализация на серверной стороне:

Листинг 4.4 – Функция регистрации пользователя

app.post("/login", (req, res) => {

  const phone = req.body.phone;

  const password = req.body.password;

  db.query(

      "SELECT \* FROM users WHERE phone = ? AND password = ? AND deleteStatus IS NOT NULL",

      [phone, password],

      (err, result) => {

          if (err) {

              res.send({err: err});

          }

          if(result.length > 0){

              const user = {

                  user\_id: result[0].user\_id,

                  firstName: result[0].firstName,

                  role: result[0].role,

                  deleteStatus: result[0].deleteStatus

              };

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

              res.send(user);

          } else {

              res.send({message: "Wrong phone/password "}); // Вернуть сообщение об ошибке, если пользователь не найден или удален

          }

      }

  );

});

Рассмотрим действия для администратора:

**Добавление нового автомобиля**

Процесс добавления нового автомобиля включает следующие этапы:

* Сбор данных автомобиля: Администратор вводит данные автомобиля в форму.
* Валидация данных на клиентской стороне: Проверка на заполненность всех полей.
* Отправка данных на сервер: Если данные корректны, они отправляются на сервер с помощью POST-запроса.
* Сохранение данных в базе данных: Сервер сохраняет данные автомобиля в базе данных.

Реализация на клиентской стороне:

Листинг 4.5 – Функция добавления нового авто администратором

const addCar = async () => {

    try {

      await Axios.post("http://localhost:3307/addcar", newCar);

      toastMessage("Автомобиль успешно добавлен!");

      setNewCar({

        brand: "",

        carName: "",

        model: "",

        price: "",

        power: "",

        gps: "",

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

        automatic: "",

        description: "",

        status: "",

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

        imgUrl: "",

      });

      fetchAllCars();

      setAddingCar(false);

    } catch (error) {

      console.error("Ошибка при добавлении автомобиля:", error);

    }

  };

Реализация на серверной стороне:

Листинг 4.6 – добавления нового авто администратором на сервере

  app.post("/addCar", (req, res) => {

    const { brand, carName, model, price, power, gps, automatic, description, imgUrl } = req.body;

    db.query(

      "INSERT INTO cars (brand, carName, model, price, power, gps, automatic, description, imgUrl, status) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, 'Временно не работает')",

      [brand, carName, model, price, power, gps, automatic, description, imgUrl],

      (err, result) => {

          if (err) {

              console.error("Error when adding a car:", err);

              return res.status(500).send("Error when adding a car:");

          }

          console.log("The car was added successfully");

          return res.status(200).send("The car was added successfully");

      }

    );

  });

В результате реализации описанных вариантов использования и служебных операций обеспечивается корректное взаимодействие различных частей ПС, что позволяет пользователям удобно и безопасно регистрироваться, авторизоваться, а для администратора добавлять новые автомобили автомобилей.

**4.3 Модульное тестирование**

Модульное тестирование – заключается в изолированной проверке каждого отдельного элемента путем запуска тестов в искусственной среде. Поэлементное тестирование – первейшая возможность реализовать исходный код. Оценивая каждый элемент изолированно и подтверждая корректность его работы, точно установить проблему значительно проще чем, если бы элемент был частью системы [7].

Примеры тестирования алгоритмов реализации вариантов использования:

1. Тестирование модуля оформление заказа на аренду автомобиля

Ожидаемый результат: пользователь выбирает нужный ему автомобиль, после чего вводит необходимые данные (дата аренды, время в которое нужно предоставить автомобиль, комментарии, если таковые нужны, а также способ оплаты), после чего пользователь получит уведомление об успешном принятии заказа.

Шаги для воспроизведения:  
1) Перейти на страницу с автомобилями.

2) Выбрав нужный автомобиль нажать на кнопку “Арендовать”.

3) Ввести необходимые данные.

4) Нажать кнопку “Арендовать”, тем самым забронировать автомобиль.

Результат: заказ будет создан, а пользователь получит уведомлении об успешном оформлении заказа.

Изм.

Лист

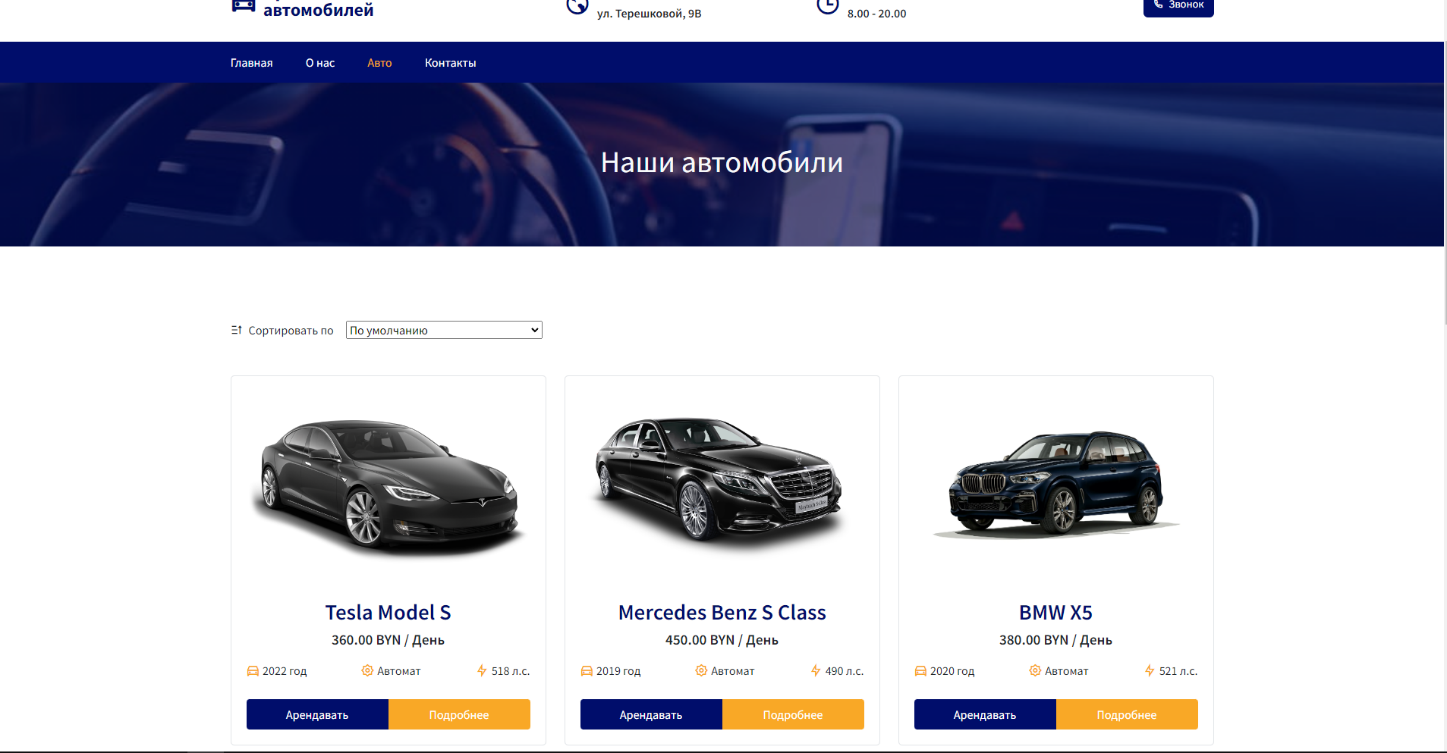
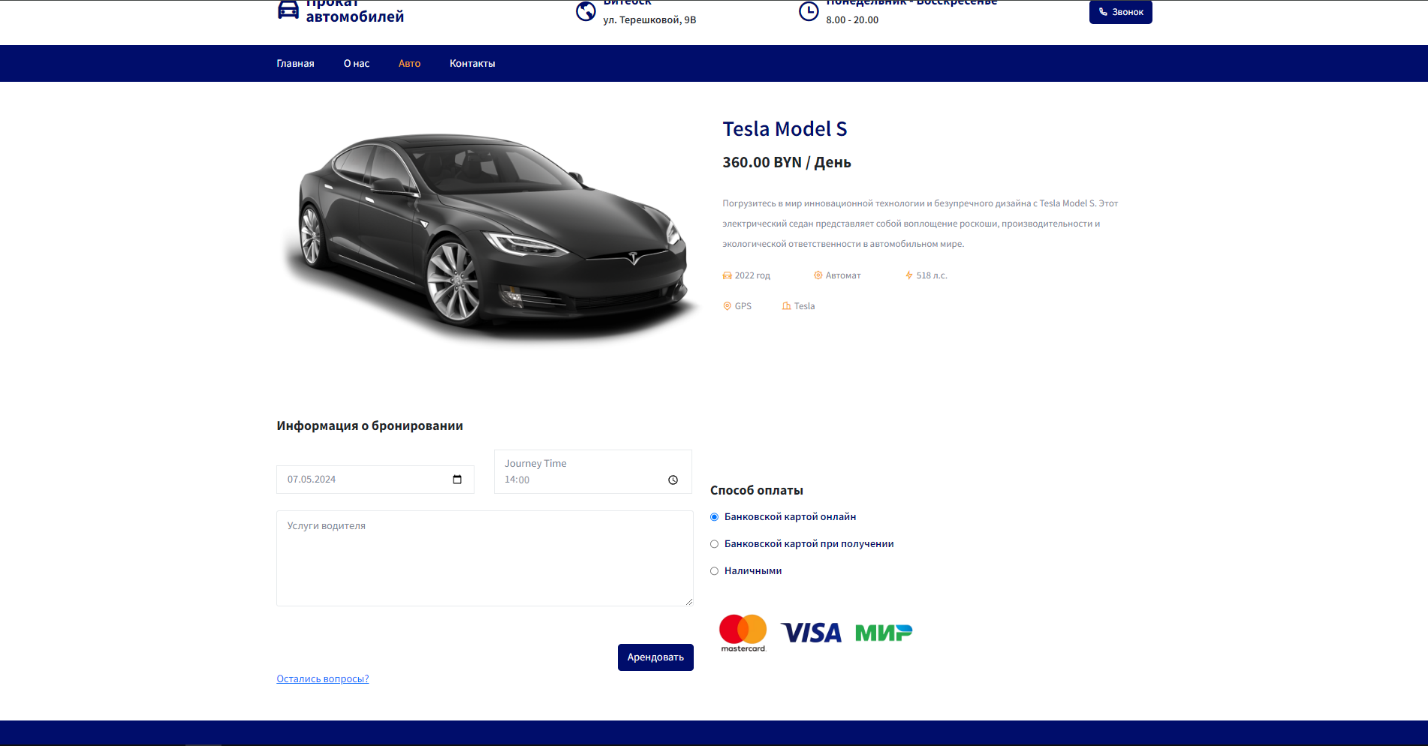
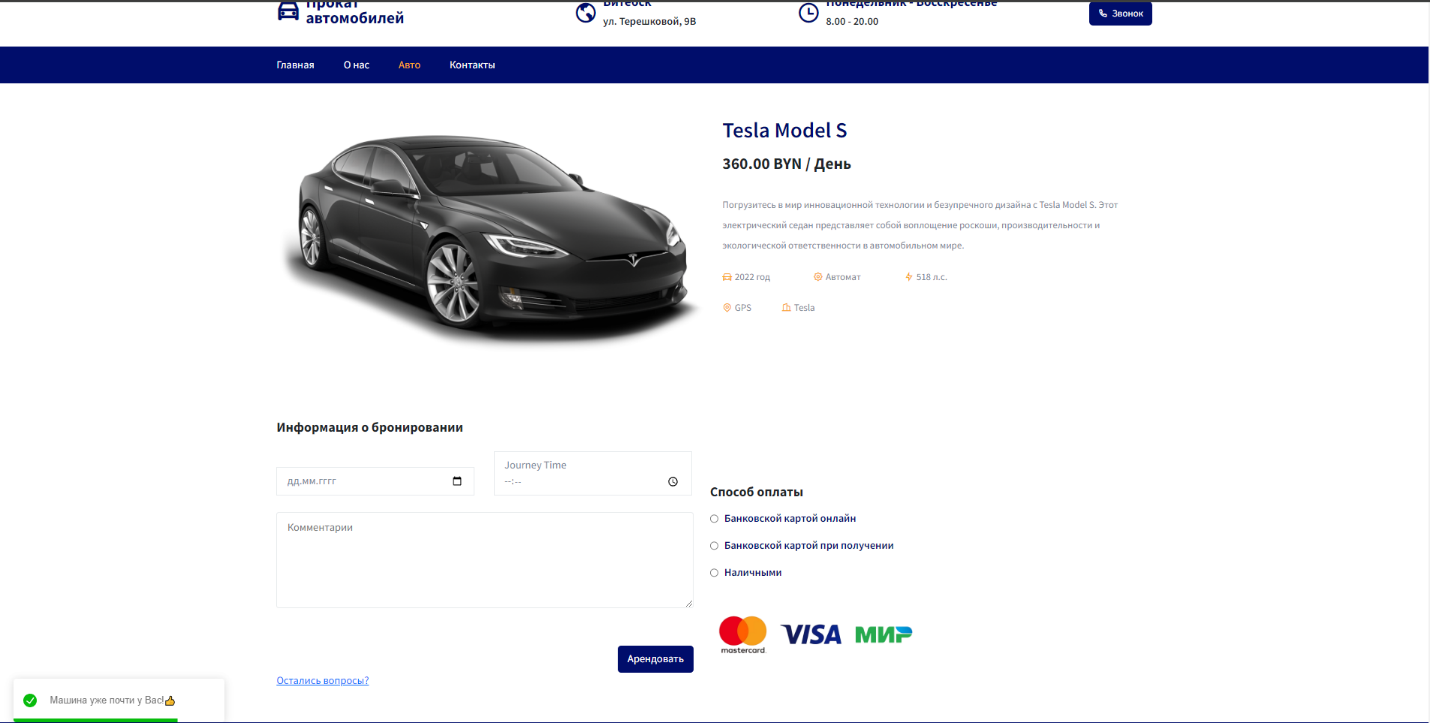
№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ



Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

Рисунок 4. … – Тестирование оформление заказа на автомобиль

1. Тестирование модуля оформление заказа на аренду автомобиля

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

Ожидаемый результат: если пользователь не ввел какие-нибудь из данных, он получит уведомление об ошибке “Заполните поле”, и “Выберите способ оплаты” в том случаи, если пользователь не выбрал способ оплаты.

Шаги для воспроизведения:

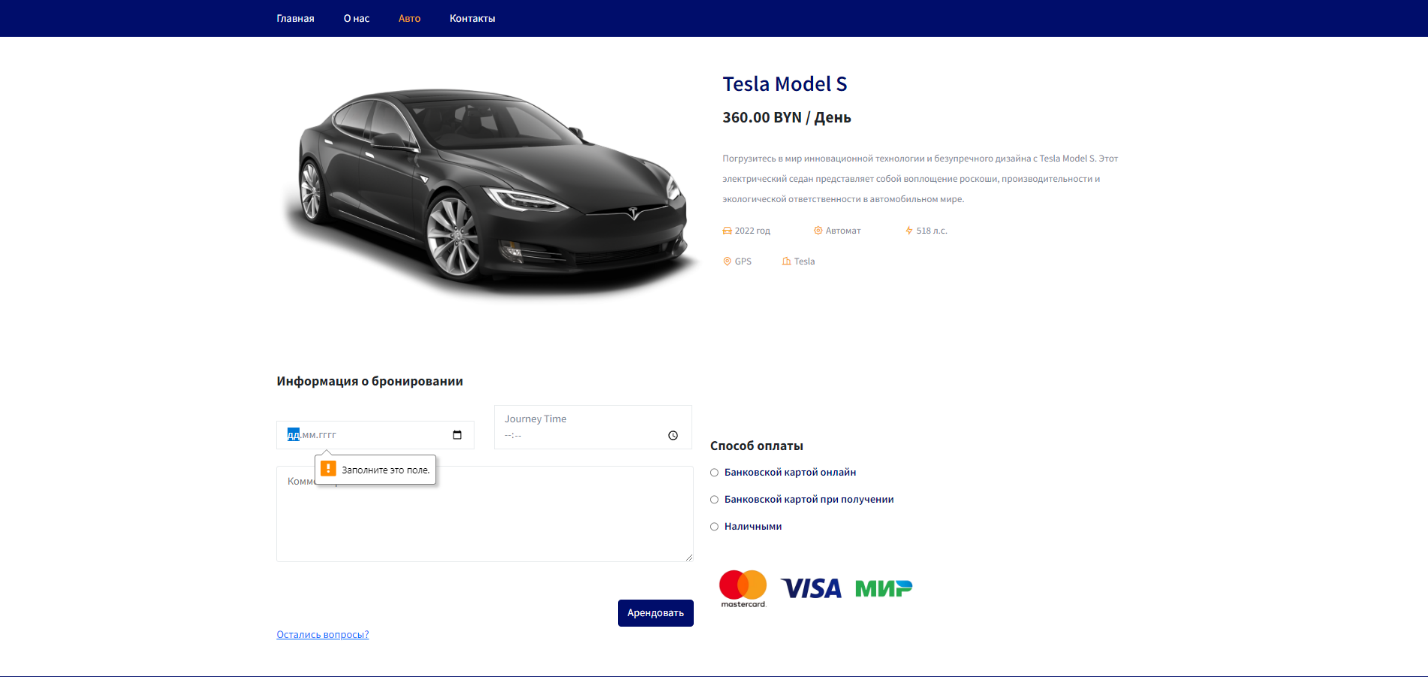
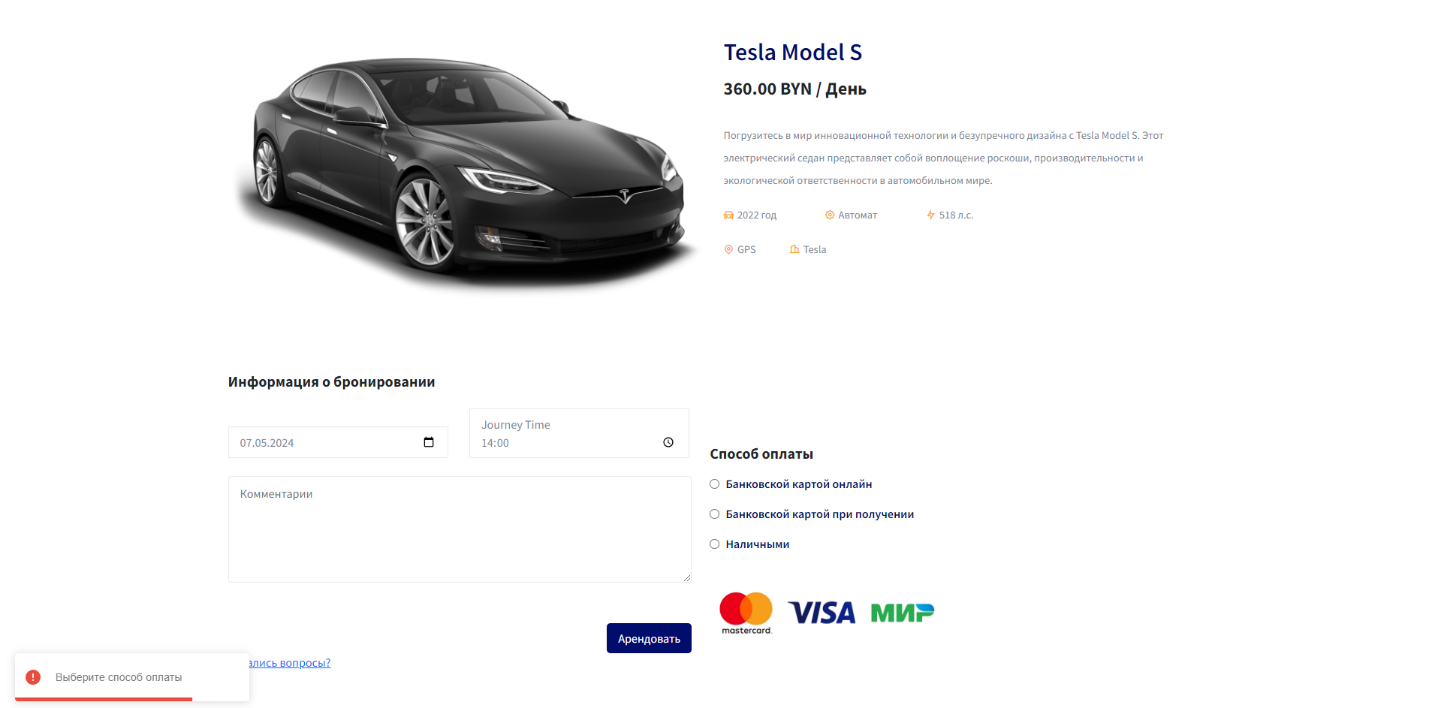
1) Перейти на страницу с автомобилями.

2) Выбрав нужный автомобиль нажать на кнопку “Арендовать”.

3) Нажать кнопку “Арендовать” без ввода данных.

4) Нажать кнопку “Арендовать” без выбора способа оплаты.

Результат : пользователь получит уведомления, о том что не все поля заполнены.

Рисунок 4. … – Тестирование оформление заказа на автомобиль

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

1. Тестирование модуля вопроса помощи от пользователя.

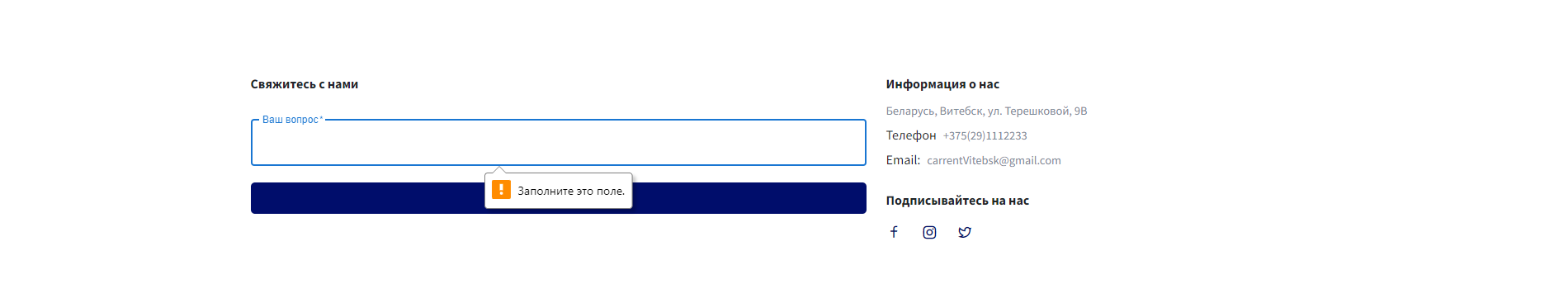
Ожидаемый результат: пользователь, если у него остались вопросы, или у него возникли проблемы, может задать вопрос.

Шаги воспроизведения:

1) Перейти на страницу “Контакты”

2) Написать свой вопрос или проблему.

3)Нажать кнопку “Отправить сообщение”.

Результат: если поле было заполнено, пользователь получит уведомление о том, что его вопрос или проблема, была принята и с ним свяжутся, а если поле не было заполнено, пользователь получит уведомление об ошибке “Заполните поле”.

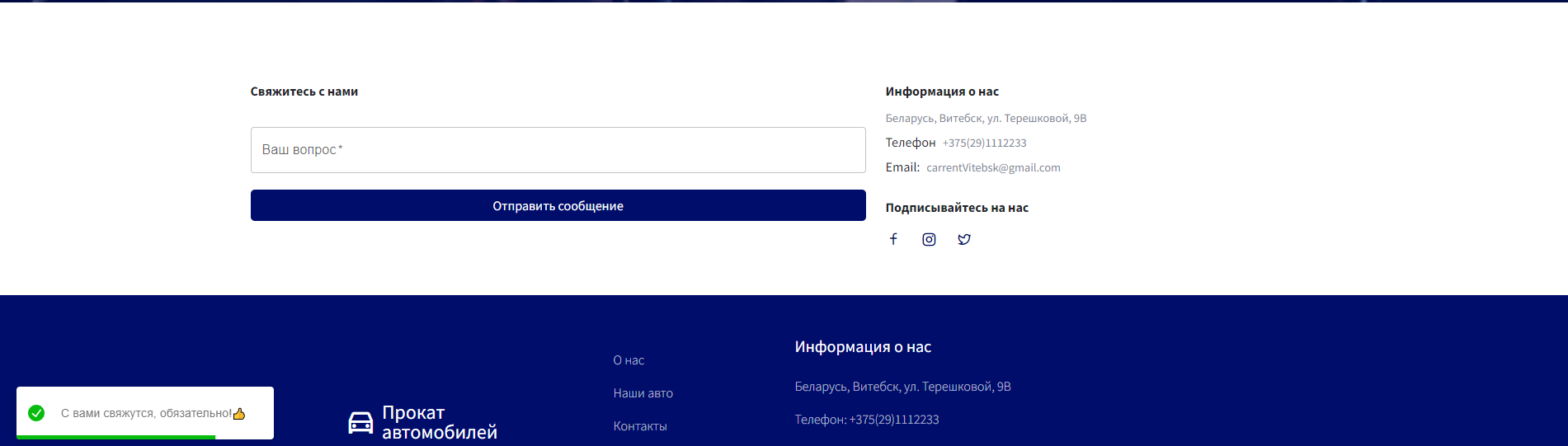
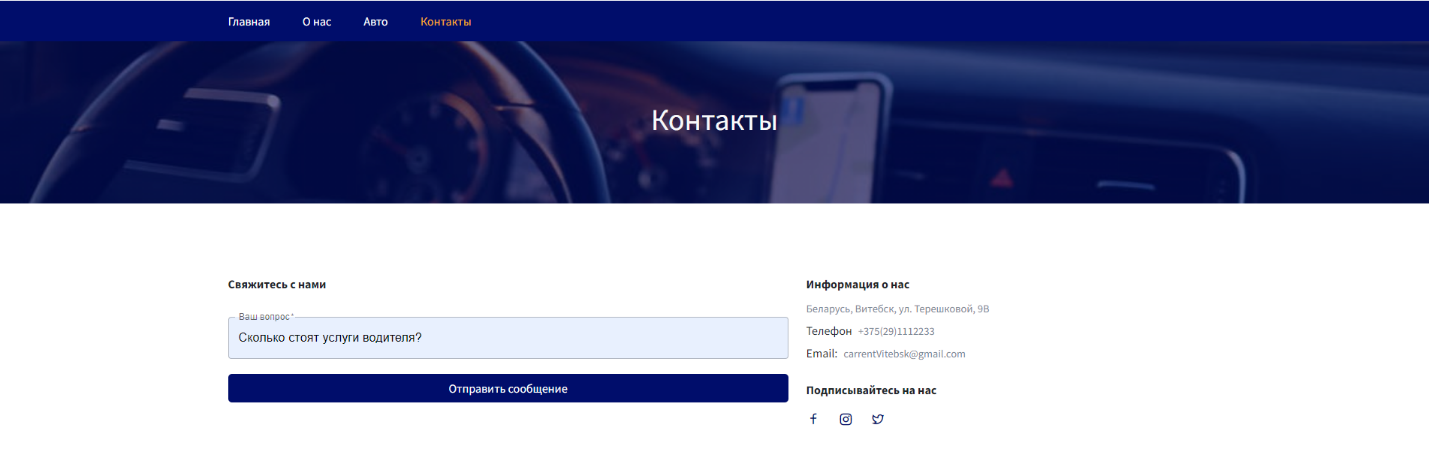


Рисунок 4. … – Тестирование модуля вопроса помощи

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

**4.4 Функциональное тестирование**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

Тестирование и отладка программы являются наиважнейшими этапами разработки любых программных продуктов. Под тестированием понимают проверку соответствия между реальным и ожидаемым поведением программы, осуществляемую на конечном наборе тестов, выбранном определенным образом. В случае выявления некорректной работы программного продукта необходимо проведение работ по их исправлению.

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

На основе функций, которые должны быть протестированы в разработанной СУЛ составлен чек-лист (checklist), приведен в таблице 5.1.

Таблица 5.1 – Чек-лист

|  |  |  |
| --- | --- | --- |
| Тестируемый модуль | Тестируемая функция | Результат |
| 1 | 2 | 3 |
| Вход в систему | Вход в систему под различными пользователями | Выполнено успешно |
| Регистрация | Регистрация нового пользователя | Выполнено успешно |
| Мой аккаунт | Отображение истории заказов | Выполнено успешно |
| Отображение истории вопросов помощи |
| Просмотр/Редактирование профиля |
| Выход из системы |
| Каталог автомобилей | Редактирование автомобиля | Выполнено успешно |
| Фильтрация автомобилей |
| Страница оформления заказа | Оформление заказа | Выполнено успешно |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Вопросы пользователей | Поддержка обратной связи с помощью почты и/или телефона | Выполнено успешно |

Окончание таблицы 5.1

Тест-кейс (test case) – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

На основе составленного чек-листа разработан набор тест-кейсов:

Таблица 5.2 – Тест-кейсы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Модуль | Описание теста / предварительные действия | Детализация / ввод данных | Ожидаемый результат | Статус («выполнено успешно», «выполнение завершилось ошибкой») |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | Вход и регистрация в системе | **Регистрация /**  На странице входа нажать кнопку «Нет аккаунта? Зарегистрируйтесь» | В предложенных для заполнения полях ввести фамилию, имя, телефон, email и придуманный пароль, нажать кнопку «Зарегистрироваться» | Пользователь зарегистрирован, появляется строка с данным пользователем на вкладке «Пользователи» в админ-панели. | выполнено успешно |
| 2 | **Регистрация при некорректных данных /** На странице входа нажать кнопку «Нет аккаунта? Зарегистрируйтесь» | В предложенных для заполнения полях ввести фамилию, имя, телефон, адрес почты: «qwerty» и придуманный пароль, нажать кнопку «Регистрация» | Регистрация не осуществляется. Появляется сообщение «Адрес электронной почты должен содержать символ «@». В адресе «qwerty» отсутствует символ «@». | выполнено успешно |

Продолжение таблицы 5.2

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 3 |  | **Проверка входа под существующим пользователем** / На странице входа нажать кнопку «Войти» | В предложенных для заполнения полях ввести телефон и пароль, нажать кнопку «Войти» | Осуществляется вход в систему под данным пользователем. | выполнено успешно |
| 4 |  | **Проверка входа под несуществующим пользователем** / На странице входа нажать кнопку «Войти» | В предложенных для заполнения полях ввести телефон и пароль, нажать кнопку «Войти» | Появляется сообщение об ошибке | выполнено успешно |
| 5 | Проверка работы меню | **Проверка работы меню** | Перейти по пунктам меню (ссылкам) «Главная»,  «О нас», «Авто», «Контакты». | При переходе по данным пунктам меню отображаются каталоги автомобилей и страницы с дополнительной информацией | выполнено успешно |
| 6 | Каталог и товары | **Проверка добавления автомобиля /** 1) Зайти под пользователем с правами администратора;  2) Зайти в админ-панель;  3) Перейти на вкладку «Автомобили» | 1) Нажать кнопку «Добавить автомобиль», ввести данные в поля:  - бренд;  - название;  - модель;  - цена  - мощность;  - задать изображение товара;  - описание  2) Нажать кнопку «Добавить автомобиль». | Добавленный автомобиль отображается в списке автомобилей. | выполнено успешно |

Окончание таблицы 5.2

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» ДП.019 1-40 05 01-01 РПЗ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 |  | **Проверка редактирования автомобиля /**  1) Зайти под пользователем с правами администратора;  2) Зайти в админ-панель;  3) Перейти на вкладку «Автомобили» | 1) Навести мышкой на необходимый для редактирования автомобиль, и нажать на кнопку «Редактировать» | Происходит редактирование товара. | выполнено успешно |
|  | Фильтрация | **Фильтрация товаров по цене /**  Перейти в каталог автомобилей | Сверху над автомобилями выбрать фильтр по цене min/max; | Автомобили отображаются в зависимости от выбранного фильтра цены | выполнено успешно |
| 9 | Вопросы пользователей | **Проверка обратной связи /**  Перейти на страницу «Контакты» | 1) Ввести данные в поля:  - Ваш вопрос: Сколько стоят услуги водителя?  2) Нажать кнопку «Отправить сообщение». | Появляется сообщение «С вами обязательно свяжутся». В админ- панели во складе «Сообщения», админ увидит вопрос пользователя | выполнено  успешно |
| 10 | Оформление заказа | **Оформления заказа /**  Перейти на страницу автомобиля, и нажать кнопку «Арендовать» | 1) Ввести данные в поля:  - Дата: 07.05.2024;  - Время: 14.00;  - Комментарии: услуги водителя;  - Способ оплаты: Банковской картой онлайн  2)Нажать кнопку «Арендовать». | Появляется сообщение об успешном принятии заказа, заказ отображается на вкладке «Заказы» у администратора. Также заказ отображается в личном кабинете пользователя на вкладке «История моих заказов» | выполнено успешно |

