Утверждаю

Руководитель предприятия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, печать предприятия)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023г

Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

ОТЧЕТ

по производственной технологической практике

на \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование предприятия)

Исполнитель

студент \_\_ курса \_\_\_\_группы \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (Ф.И.О.)

Руководитель практики

от предприятия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, печать предприятия) (подпись, дата) (Ф.И.О.)

Руководитель практики

от кафедры

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, уч. звание) (подпись, дата) (Ф.И.О.)

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Минск 2023 г.

Оглавление

[Введение 3](#_Toc140933383)

[1 Общая характеристика сайта 4](#_Toc140933384)

[2 Описание используемого ПО 5](#_Toc140933385)

[2.1 Java 5](#_Toc140933386)

[2.2 Selenium 5](#_Toc140933387)

[2.3 IntelliJ IDEA 5](#_Toc140933388)

[3 Разработка проекта 6](#_Toc140933389)

[3.1 Проектирование программного средства 6](#_Toc140933390)

[3.2 Тесты и результаты их выполнения 9](#_Toc140933391)

[Заключение 18](#_Toc140933392)

[Список использованных источников 19](#_Toc140933393)

[Приложение А 20](#_Toc140933394)

[Приложение Б 23](#_Toc140933395)

# **Введение**

Компания "Автотега" – это крупнейшая компания в области продаж контрактных тестированных двигателей и автозапчастей б/у для автомобилей иностранного производства, осуществляющая прямые и регулярные поставки с крупнейших авторазборок Европы.

Более 15 лет успешной работы на рынке б/у автозапчастей позволили им создать крупнейшую складскую и розничная сеть на территории Республики Беларусь и России. Все двигатели и запчасти легально ввезены на территорию Таможенного Союза с крупных авторазборок Германии и Англии, что подтверждается необходимыми документами.

Доклад представляет собой обзор результатов тестирования сайта продажи автозапчастей, с целью выявления его функциональных возможностей, удобства использования и общей надежности. Осуществленное тестирование имеет решающее значение для обеспечения высокого качества пользовательского опыта, а также повышения конкурентоспособности компании на рынке автозапчастей.

Данный отчет содержит подробное описание проведенных тестов, используемых методологий и критериев оценки, что позволяет точно определить сильные и слабые стороны сайта. Это поможет идентифицировать возможные проблемы и недостатки, а также предложить соответствующие рекомендации по их устранению.

Сайт продажи автозапчастей является важным каналом для взаимодействия с клиентами и потенциальными покупателями. Поэтому обеспечение его высокой производительности и надежности не только снижает риски потери клиентов, но и способствует улучшению репутации компании на рынке.

Целью данной производственной практики является разработка приложения для тестирования сайта компании.

Задачи, решение которых позволит достигнуть цели:

* Провести аналитический обзор сайта компании;
* Определить основные характеристики и особенности сайта;
* Определить набор необходимых инструментов;
* Разработать составляющие компоненты разработки для тестов;
* Разработать тесты.

# **1 Общая характеристика сайта**

Тестируемое веб-приложение представляет собой сайт по продаже автозапчастей, написанный с помощью языка гипертекстовой разметки html и стилизованный и помощью css. Интерактивные элементы добавлены с использованием языка программирования JavaScript. Снимок вида главной страницы представлен на рисунке 1.

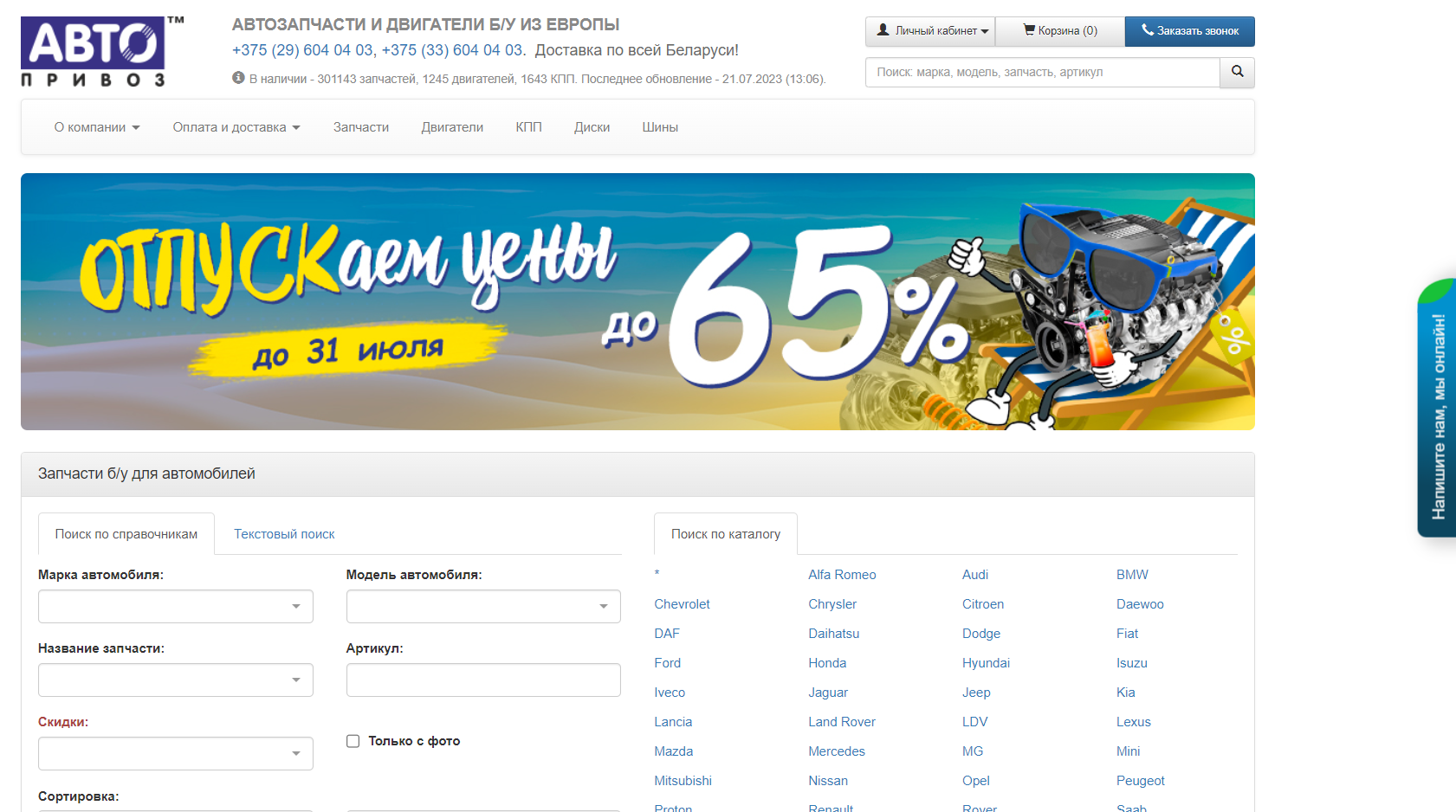


Рисунок 1 – Главная страница сайта

На сайте присутствуют ключевые элементы, характерные для данной категории веб-приложений: каталог товаров, поиск по каталогу и возможность связаться с центром поддержки.

Пользователь имеет возможность регистрации, используя номер мобильного телефона в международном формате и адрес электронной почты, для однозначной его идентификации. После на мобильный телефон клиента придет сгенерированный из случайных символов пароль, который в дальнейшем будет использоваться для входа на сайт. Для заказа какого-либо товара, пользователю необходимо войти на сайт, введя номер телефона или адрес электронной почты и пароль, ранее полученный при регистрации.

Для более быстрого и удобного заказа необходимых деталей из каталога на сайте предусмотрен текстовый поиск по марке машины, бренду производителя и по артикулу самой детали, который у каждой детали является уникальным в пределах каталога. Также присутствует расширенный поиск, с помощью которого можно задать процент скидки, вид сортировки и дополнительные параметры, такие как год выпуска, адрес склада, объем двигателя, тип двигателя, особенности двигателя и тд.

# **2 Описание используемого ПО**

В данном разделе рассматриваются основные программы, которые были использованы в процессе разработки программы тестирования описанного выше сайта.

## **2.1 Java**

Java – это высокоуровневый, объектно-ориентированный, классический язык программирования, разработанный компанией Sun Microsystems (позднее приобретенной Oracle Corporation). Он был представлен в 1995 году и с тех пор стал одним из самых популярных и широко используемых языков программирования.

Одной из ключевых особенностей Java является его "write once, run anywhere" (WORA) подход, что означает, что один и тот же код может быть запущен на различных платформах без необходимости изменений. Это достигается благодаря использованию виртуальной машины Java (JVM), которая интерпретирует байт-код Java, полученный в результате компиляции исходного кода на Java. Таким образом, Java обеспечивает платформенную независимость и переносимость программ.

## **2.2 Selenium**

Selenium – это популярный инструмент для автоматизации тестирования веб-приложений. Он предоставляет программный интерфейс, позволяющий разработчикам и тестировщикам создавать автоматизированные тесты для проверки функциональности веб-приложений на различных браузерах.

WebDriver: Это основной компонент Selenium, который обеспечивает управление браузерами. WebDriver предоставляет простой и интуитивный API для взаимодействия с веб-элементами, такими как кнопки, поля ввода, выпадающие списки и другие. Он позволяет автоматизировать действия пользователя, такие как клики, ввод текста, выбор из списка и т. д.

## **2.3 IntelliJ IDEA**

IntelliJ IDEA – это интегрированная среда разработки (Integrated Development Environment, IDE), разработанная компанией JetBrains, предназначенная для программирования на различных языках, включая Java, Kotlin, Groovy, Scala, JavaScript, TypeScript, HTML, CSS и многие другие. Эта популярная IDE предоставляет разработчикам мощные инструменты и функциональность, которые существенно облегчают процесс разработки программного обеспечения.

# **3 Разработка проекта**

Разработка проекта для тестирования сайта компании включает ряд основных этапов, которые позволяют создать эффективный и надежный тестовый фреймворк для проверки функциональности и качества веб-приложения. В данном разделе рассмотрены основные этапы разработки проекта.

## **3.1 Проектирование программного средства**

Как ранее уже отмечалось, проект создавался в среде разработки IntelliJ IDEA. Проект состоит из таких пакетов, как:

* Common – пакет, который содержит базовый класс для непосредственно тестов;
* Drivers – пакет, содержащий класс драйверов, который отвечает за настройку драйвера;
* Pages – пакет, который содержит базовый класс для классов-страниц и сами классы-страницы;
* Tests – пакет, содержащий непосредственно тесты.

Структура проекта представлена на рисунке 2.

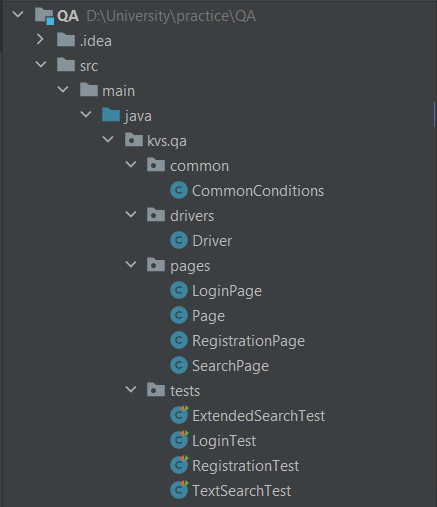


Рисунок 2 – Структура проекта

Код класса «Driver» представлен на листинге 1.

public class Driver {

private static WebDriver driver;

private Driver(){}

public static WebDriver getDriver(){

if (null == driver){

switch (System.getProperty("browser")){

case "chrome": {

WebDriverManager.chromedriver().setup();

driver = new ChromeDriver();

}

default: {

WebDriverManager.chromedriver().setup();

driver = new ChromeDriver();

}

}

driver.manage().window().maximize();

}

return driver;

}

public static void closeDriver(){

driver.quit();

driver = null;

}

}}}

Листинг 1 – Класс «Driver»

Данный код представляет собой Java класс с именем "Driver" в пакете "kvs.qa.drivers", который является утилитой для создания экземпляра WebDriver (браузера) и его закрытия.

Этот класс использует WebDriverManager для управления драйверами браузеров и предоставляет статические методы для получения и закрытия экземпляра WebDriver.

В данном коде объявлен приватный конструктор, чтобы предотвратить создание экземпляров этого класса. Это делается для того, чтобы класс был доступен только через его статические методы.

В этом блоке кода проверяется, был ли уже создан экземпляр WebDriver. Если нет, то с помощью WebDriverManager инициализируется и устанавливается драйвер для браузера Chrome (в данном случае), а затем создается объект ChromeDriver.

Если не указана опция "browser" через системные свойства, то также используется браузер Chrome.

После создания WebDriver происходит максимизация окна браузера.

В этом блоке кода вызывается метод "quit()" для закрытия текущего экземпляра WebDriver, а затем присваивается значение null переменной driver.

Общий смысл этого класса заключается в том, чтобы обеспечить один экземпляр WebDriver на протяжении всего выполнения тестового сценария, что способствует более эффективному управлению браузером и более быстрой и стабильной автоматизации тестирования веб-приложений.

Код класса «Page» представлен на листинге 2.

public class Page {

protected WebDriver driver;

protected Page(WebDriver driver) {

this.driver = driver;

PageFactory.initElements(driver, this);

}

protected static WebElement waitForVisibilityOfElement(WebDriver driver, WebElement element) {

return new WebDriverWait(driver, Duration.ofSeconds(30))

.until(ExpectedConditions.visibilityOf(element));

}

}

}

}

Листинг 2 – Класс «Page»

Данный код представляет собой Java класс с именем "Page" в пакете "kvs.qa.pages", который является базовым классом для всех страниц (Page Objects) в тестовом фреймворке автоматизации тестирования.

Этот класс содержит методы и свойства, которые могут быть общими для всех страниц приложения.

Конструктор принимает объект WebDriver в качестве параметра и сохраняет его в переменную "driver". Затем метод "PageFactory.initElements(driver, this)" инициализирует элементы страницы (Page Objects) для данного объекта WebDriver.

Метод ожидает видимости элемента (WebElement) на странице с помощью явного ожидания (WebDriverWait). Для ожидания видимости элемента используется условие "ExpectedConditions.visibilityOf(element)" из Selenium.

Дополнительно, установлено время ожидания в 30 секунд (может быть настроено по необходимости). Метод возвращает элемент после его видимости на странице.

Общий смысл этого класса заключается в том, что он предоставляет базовую функциональность и методы, которые могут быть общими для всех Page Objects (страниц) в тестовом фреймворке.

Использование этого базового класса позволяет улучшить читаемость и обслуживаемость кода, а также повысить его модульность и повторное использование.

Листинги кода наследников данного класса, а именно страницы, приведены в приложении А.

Код класса «CommonConditions» представлен на листинге 3. Данный код представляет собой Java класс с именем "CommonConditions" в пакете "kvs.qa.common", который содержит методы для настройки и завершения сеанса WebDriver (браузера) перед и после выполнения каждого тестового метода.

public class CommonConditions {

protected WebDriver driver;

@BeforeMethod

public void setUp()

{

System.setProperty("browser", "chrome");

System.setProperty("webdriver.http.factory", "jdk-http-client");

driver = Driver.getDriver();

}

@AfterMethod(alwaysRun = true)

public void stopBrowser()

{

Driver.closeDriver();

}

}}

Листинг 3 – Класс «CommonConditions»

Метод "@BeforeMethod" выполняется перед каждым тестовым методом. В этом методе выполняются предварительные настройки окружения перед запуском тестов, в данном случае, инициализируется объект WebDriver.

Здесь используются системные свойства, устанавливающие браузер (в данном случае "chrome") и фабрику HTTP-клиента (используется JDK HTTP Client). Метод "@AfterMethod" выполняется после каждого тестового метода. Производится завершение работы с браузером и освобождение ресурсов.

В данном блоке кода вызывается метод "closeDriver()" из класса "Driver", который выполняет закрытие текущего экземпляра WebDriver.

Общий смысл данного кода заключается в том, что он обеспечивает инициализацию и завершение работы с браузером перед и после каждым тестовым методом, что является хорошей практикой для обеспечения чистого и надежного окружения при проведении тестирования веб-приложений.

Листинги кода наследников данного класса, а именно тесты, приведены в приложении Б.

## **3.2 Тесты и результаты их выполнения**

Тесты и результаты их выполнения представляют собой ключевую часть процесса тестирования сайта компании. Тесты представляют собой набор тестовых сценариев, которые разработаны для проверки различных функциональных, нагрузочных, безопасностных и других аспектов веб-приложения.

Они могут быть разделены на различные категории, такие как позитивные (ожидаемо успешные) и негативные (ожидаемо с ошибкой), а также на модульные, интеграционные и системные тесты.

Каждый тест содержит набор шагов, которые должны быть выполнены в определенной последовательности, чтобы проверить определенное поведение сайта. Результаты выполнения тестов обрабатываются автоматически, и обзор результатов может быть предоставлен как в форме отчетов, так и в виде журналов выполнения тестов.

Тесты могут быть запущены вручную или автоматически, в зависимости от выбранного подхода.

Автоматизированные тесты обычно запускаются с использованием CI/CD-среды при каждом обновлении кода, чтобы обеспечить непрерывное тестирование и быструю обратную связь.

Ошибки и недостатки в веб-приложении фиксируются как неудачные тесты, а те, которые проходят без ошибок, считаются успешными. При обнаружении ошибок, выявленных в результате тестирования, они регистрируются в системе управления ошибками для дальнейшего исправления разработчиками.

Ошибки могут быть классифицированы по приоритету и серьезности, чтобы команда разработки могла определить порядок их исправления. Тестирование обычно является итеративным процессом, что означает, что тесты могут запускаться несколько раз, чтобы убедиться, что ошибки были исправлены и функциональность улучшена.

После внесения изменений разработчиками повторно запускаются соответствующие тесты для проверки исправлений. Оценка покрытия тестами позволяет определить, насколько широко функциональность сайта была протестирована.

Целью является максимальное покрытие всех ключевых аспектов сайта тестами, чтобы уменьшить риск обнаружения ошибок пользователем в процессе реального использования приложения. Результаты выполнения тестов являются основой для постоянного улучшения качества веб-приложения.

Команда разработки и тестирования должна анализировать результаты и вносить коррективы в тесты и тестовый фреймворк, чтобы обеспечить эффективное и надежное тестирование. Тесты и результаты их выполнения являются критическими элементами для обеспечения высокого качества веб-приложения, удовлетворения потребностей пользователей и успешной работы компании.

Первый тест представляет собой тестирование текстового поиска. Он возможен по артиклю детали автомобиля, по марке производителя, по названию модели автомобиля и по названию самой детали автомобиля. По этим критериям и был произведен поиск.

На рисунке 3 представлен скриншот сайта при выполнении текстового поиска по детали автомобиля.

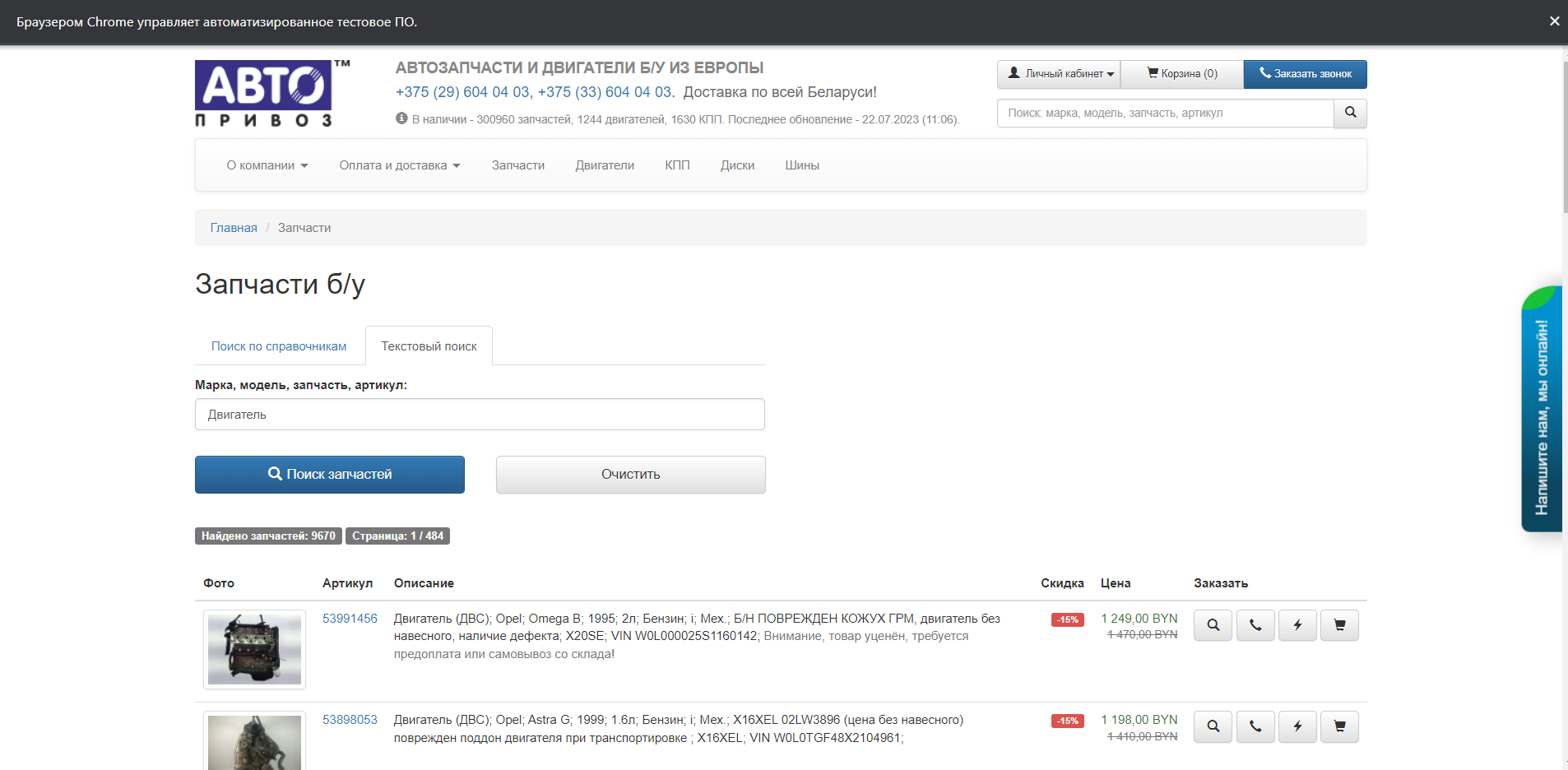


Рисунок 3 – Текстовый поиск по названию детали

Также был произведен поиск с пустым текстовым полем (рисунок 4).

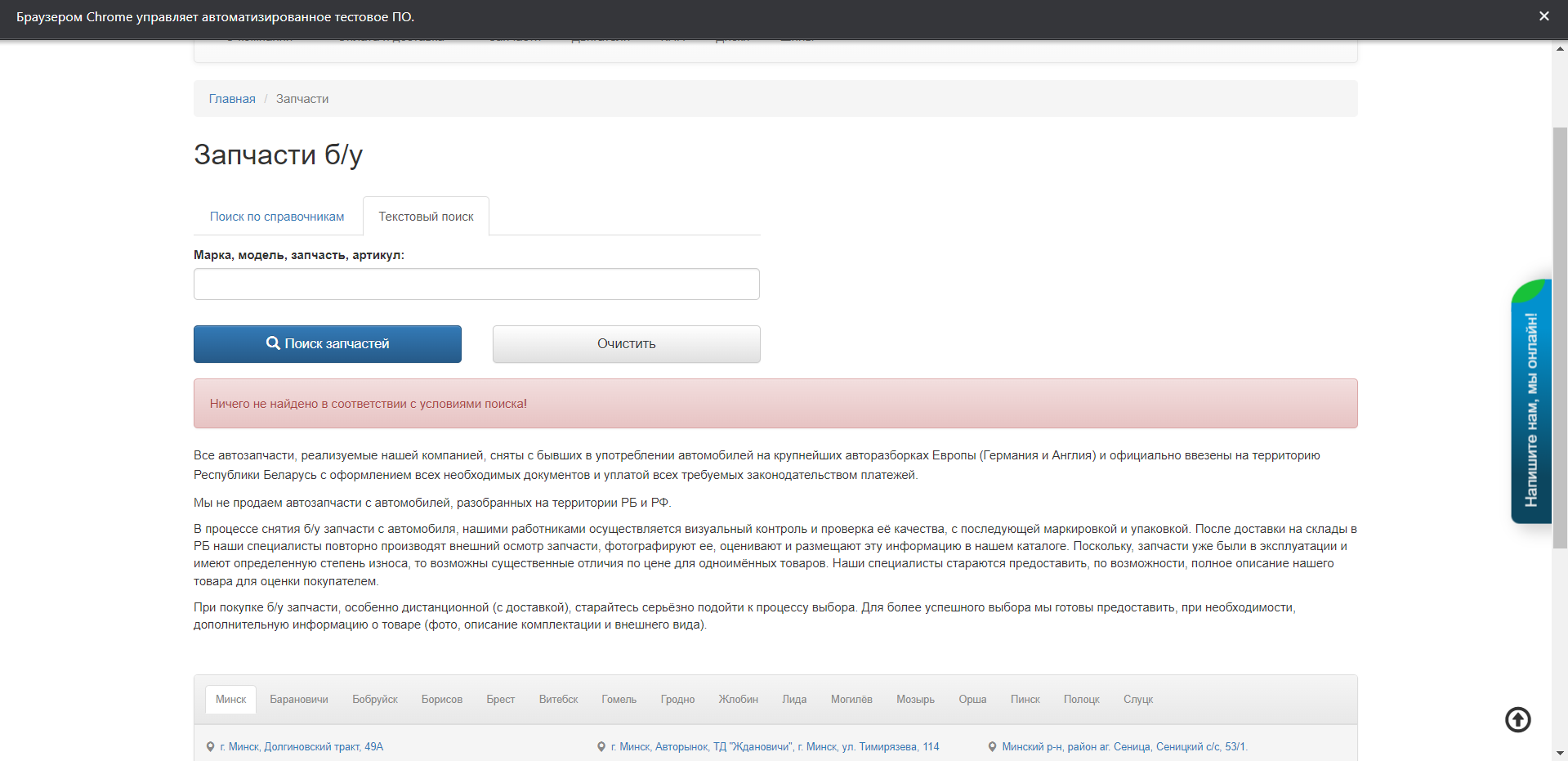


Рисунок 4 – Пустой поиск

. Как и ожидалось, поиск не дал никаких результатов. Кроме того, появилось предупреждение «Ничего не найдено в соответствии с условиями поиска».

Была протестирована кнопка очищения полей (рисунок 5).

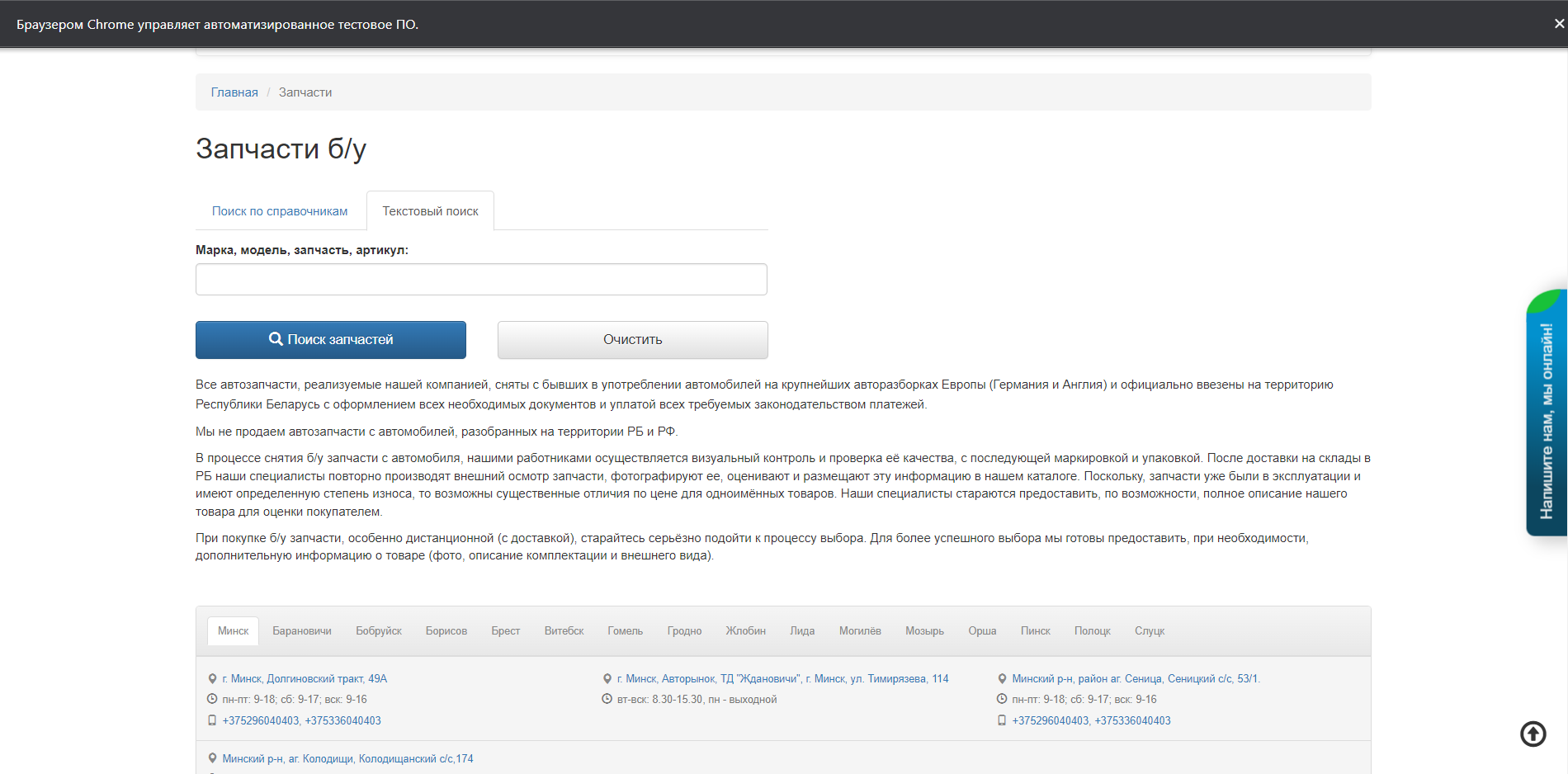


Рисунок 5 – Очищение полей

Следующий тест был проведен для расширенного поиска (рисунок 6).

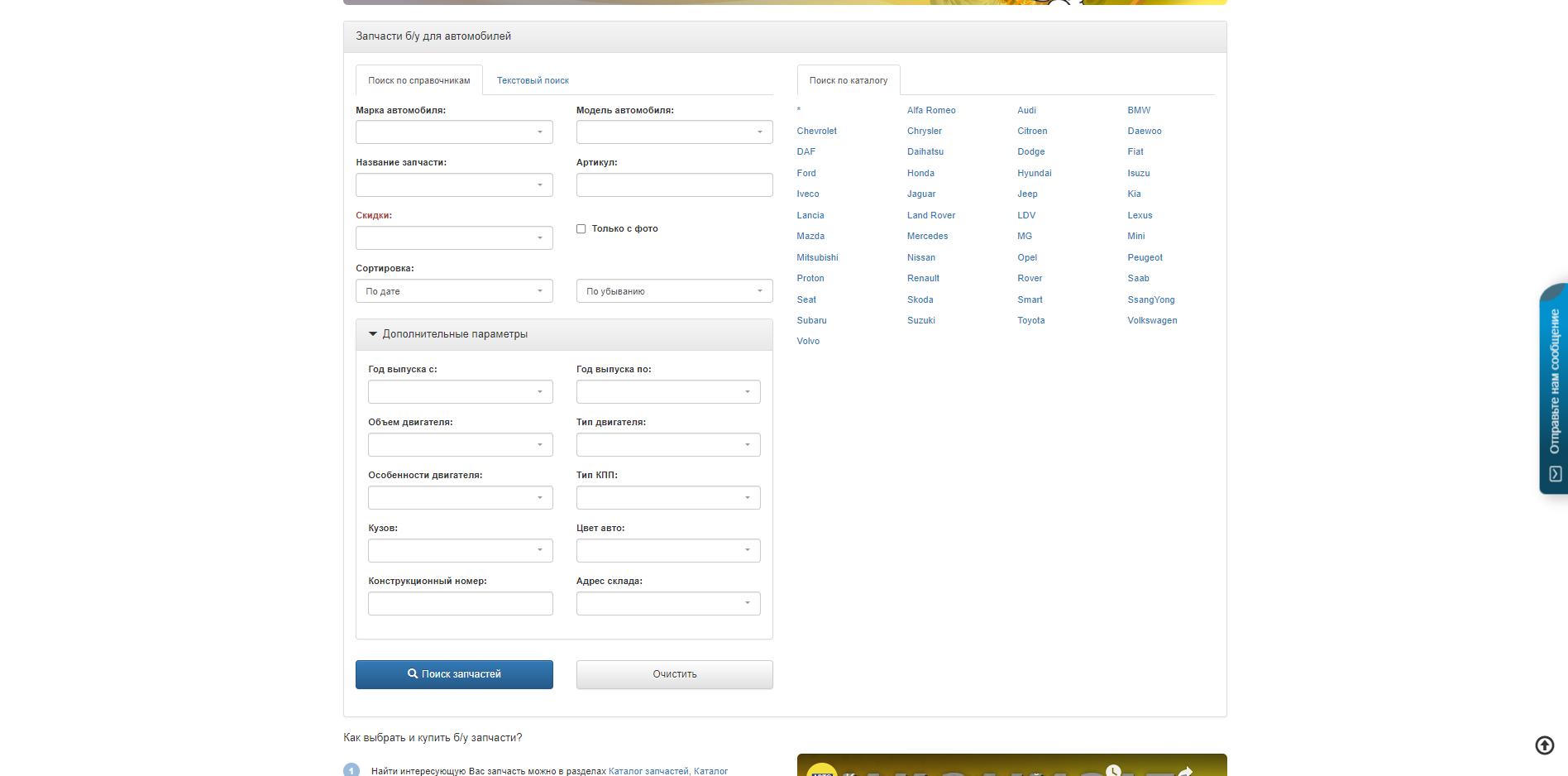


Рисунок 6 – Расширенный поиск

Расширенный поиск производится по марке автомобиля, модели автомобиля, названию запчасти, артикулу, скидке, результат отбирается только с фото. Также можно задать дополнительные параметры, такие как период выпуска по годам, объем двигателя, тип двигателя, особенности двигателя, тип кпп, кузов, цвет авто, конструкционный номер и адрес склада. После ввода всех необходимых полей, как и ожидалось, результат будет представлять собой нужную покупателю деталь (рисунок 7).

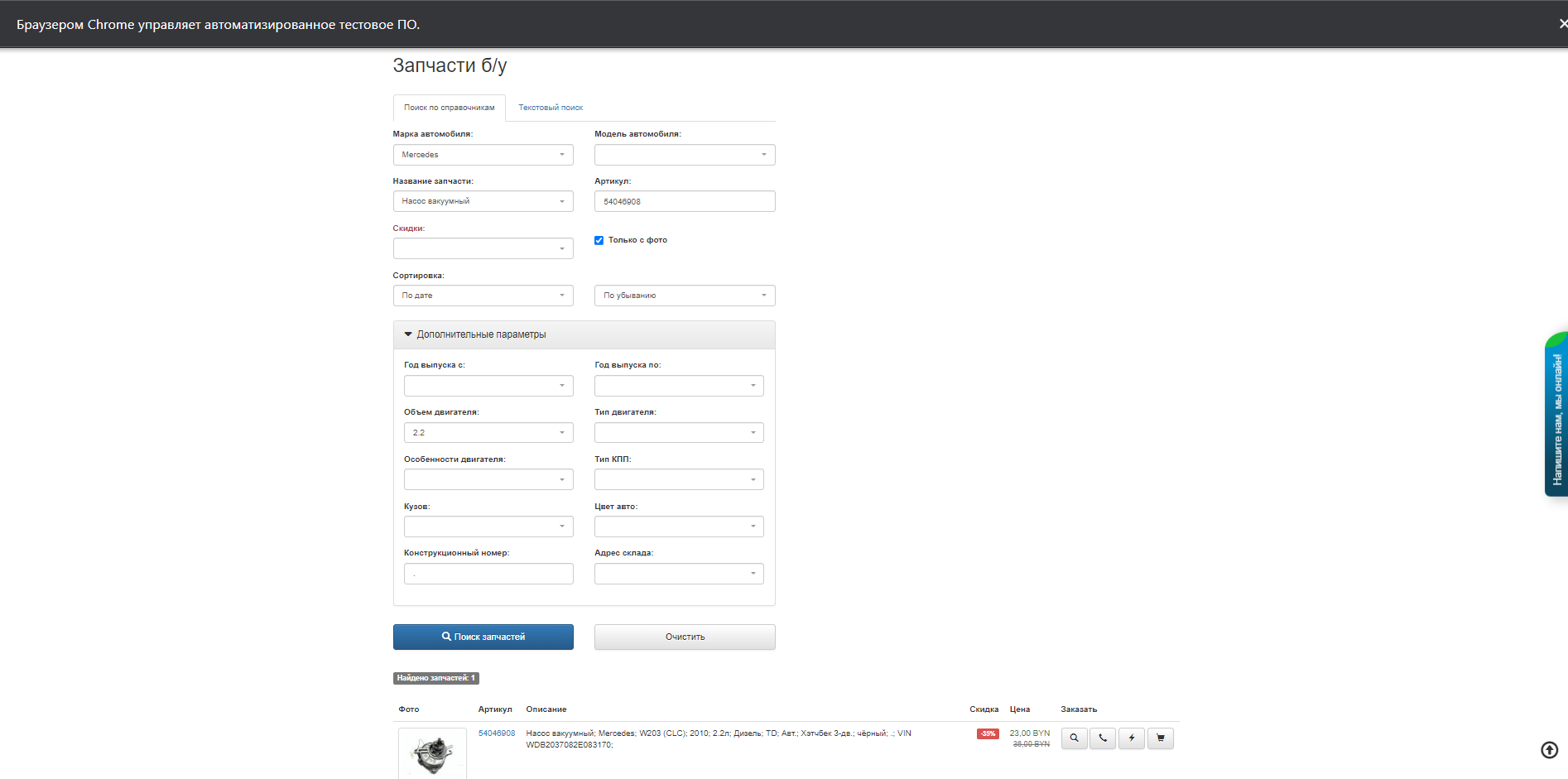


Рисунок 7 – Поиск детали по нескольким полям

Проведем поиск лишь по одному полю. Ожидается, что найдутся все детали, удовлетворяющие условию. Согласно рисунку 8, так оно и происходит.

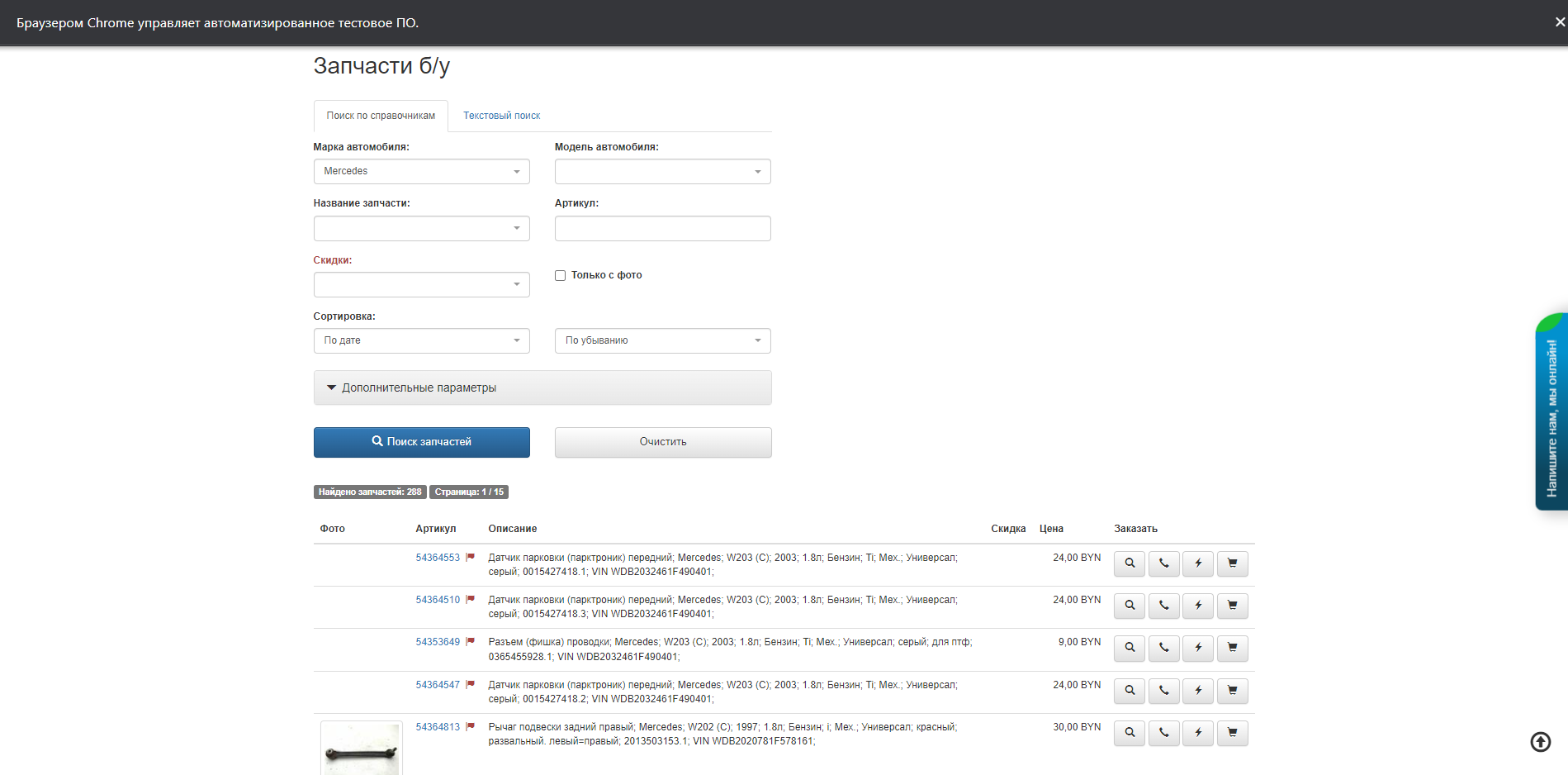


Рисунок 8 – Поиск детали по одному полю

Следующий тест проверяет одну из самых важных функций сайта, а именно, регистрацию. Для начала введем пустые значения для имени пользователя и мобильного телефона в международном формате. Не введем адрес электронной почты, это необязательное поле для заполнения, но поставим галочку но согласие на обработку персональных данных. На рисунке 9 видно, что регистрация не произошла, а пользователю было показано предупреждение о правильности заполнения формы.

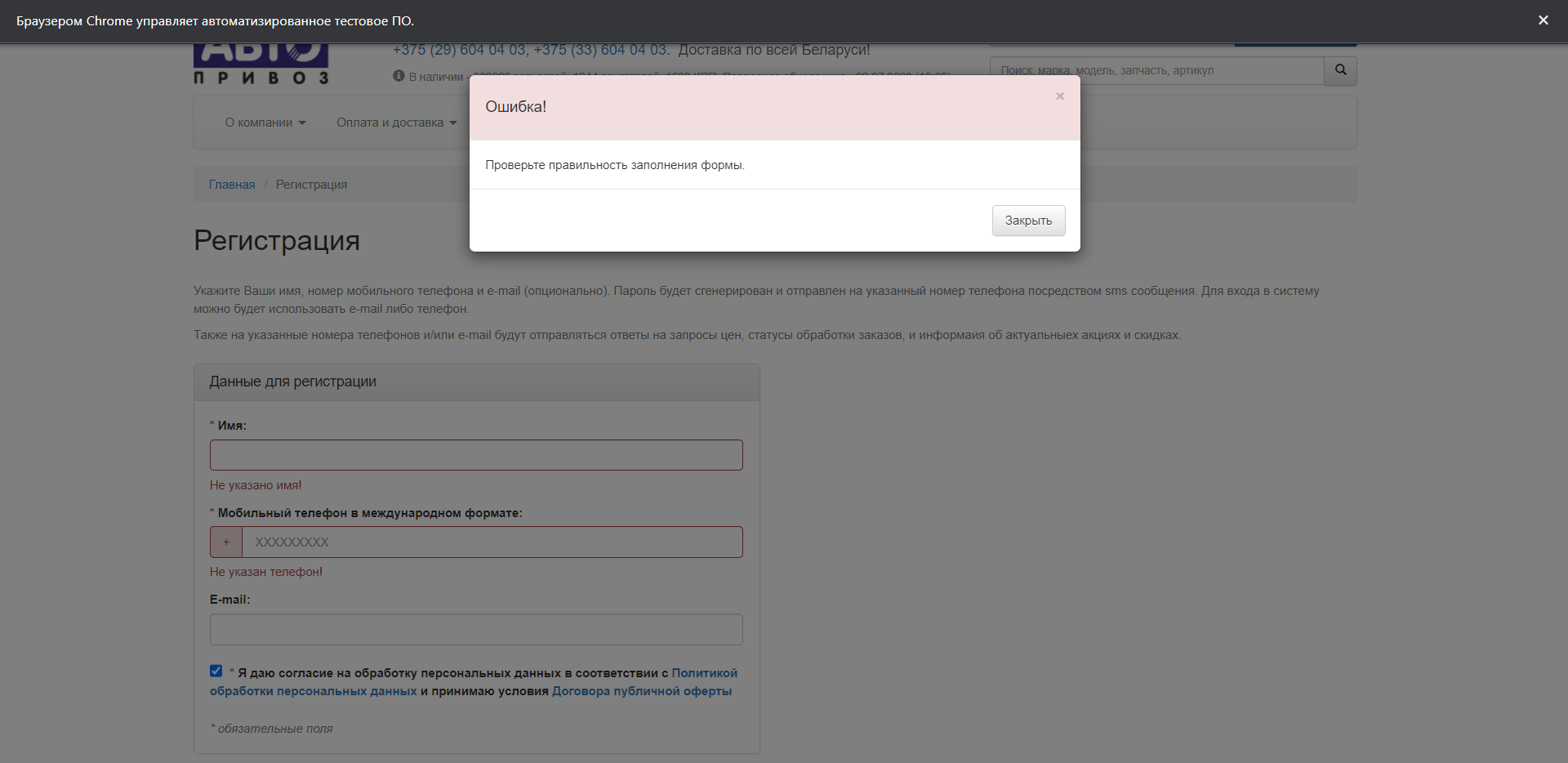


Рисунок 9 – Регистрация с пустыми полями

Не введем имя, но введем номер мобильного телефона в международном формате. Не поставим согласие на обработку персональных данных. Как видно из рисунка 10, пользователю выдается предупреждение, но мобильный телефон никак не проверяется, хотя очевидно, что он не может быть таким.

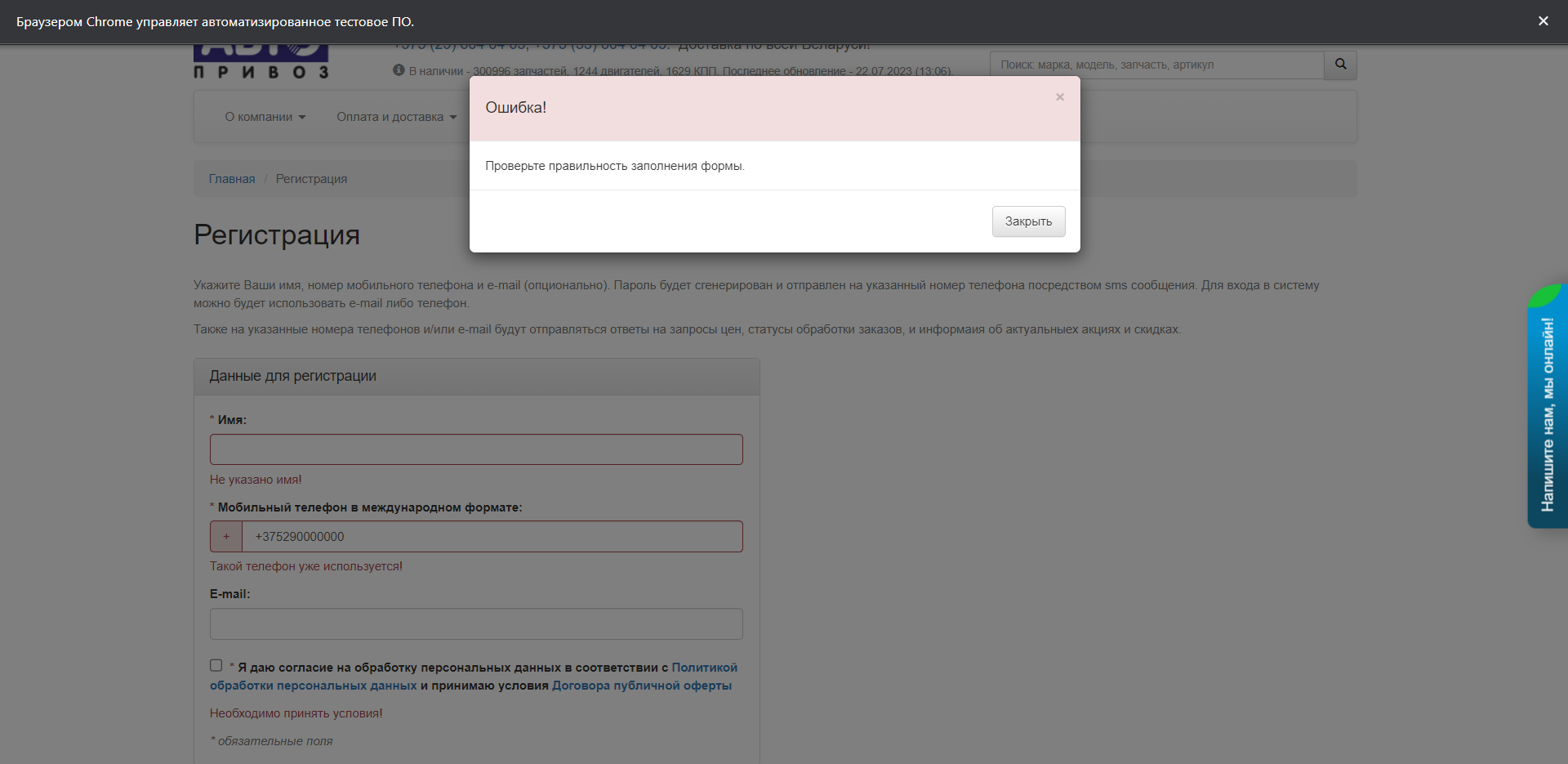


Рисунок 10 – Регистрация без ввода имени

Регистрация без ввода номера телефона также не прохдит. Появляется то же предупреждение о неверном заполнении формы, как и в предыдущих случаях (рисунок 11).

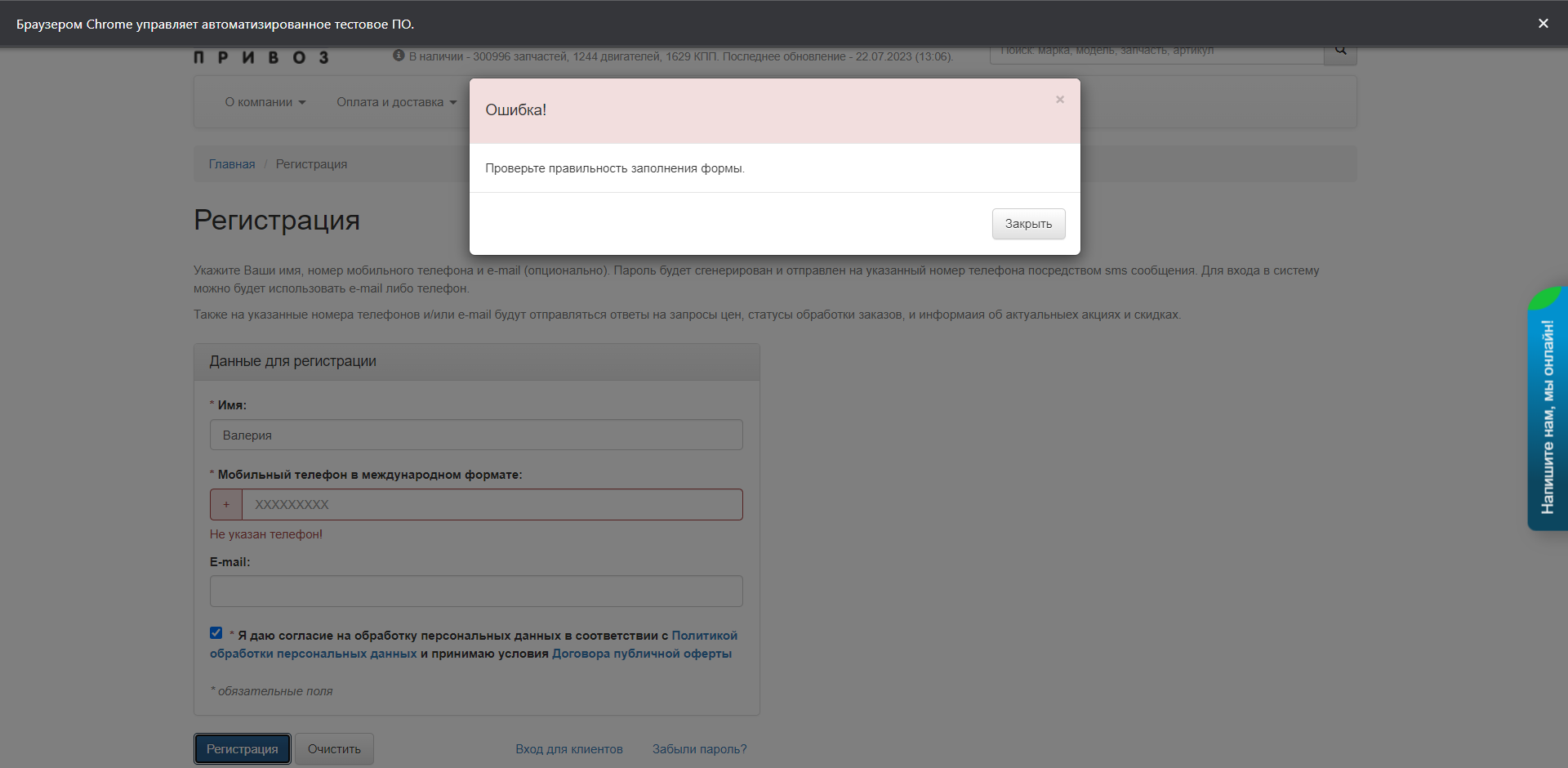


Рисунок 11 – Регистрация без ввода мобильного телефона

Проверим работоспособность кнопки очистки формы. Как видим из рисунка 12, кнопка является рабочей.

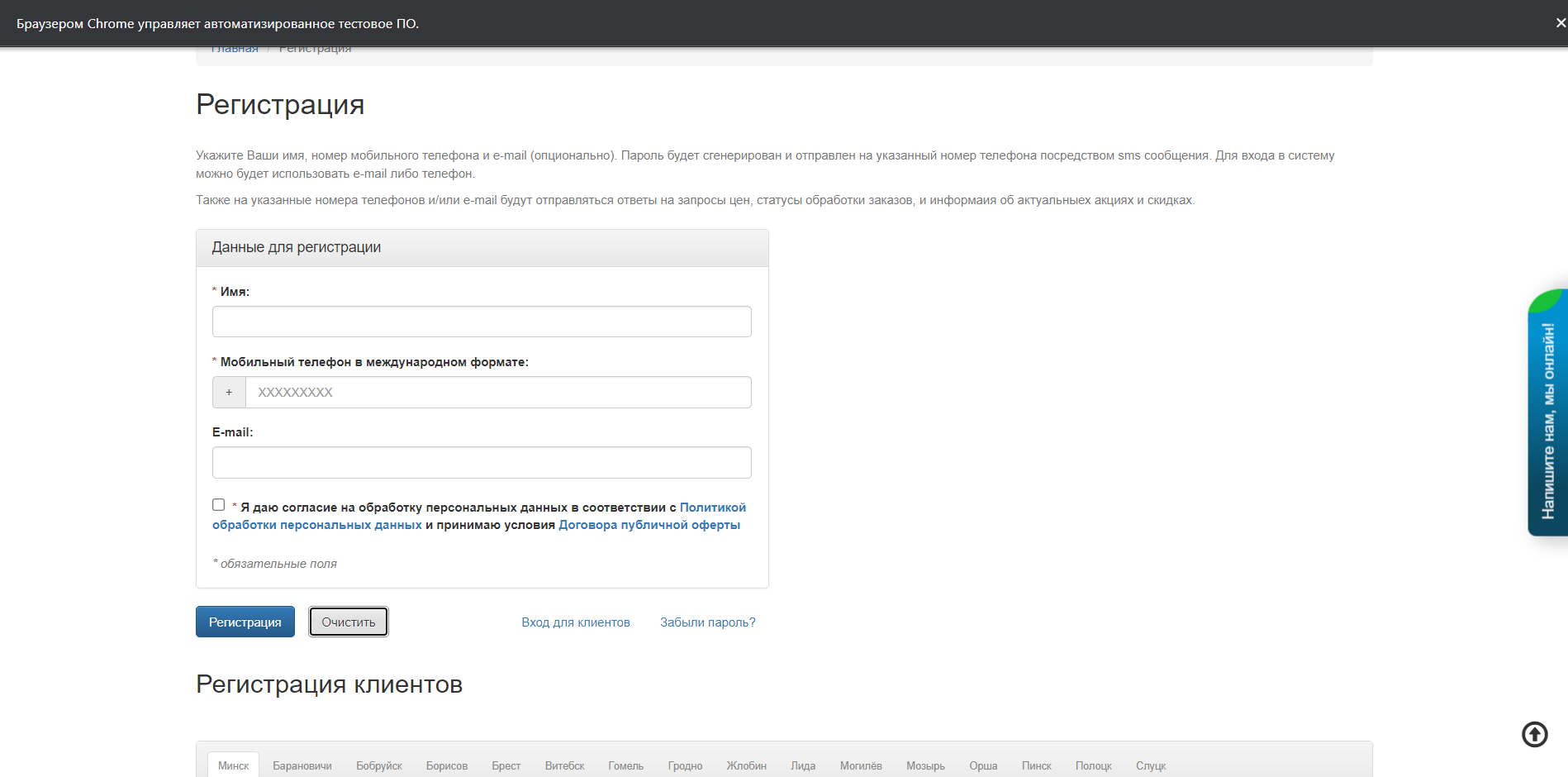


Рисунок 12 – Очистка формы

Наконец введем верные данные, имя пользователя и номер телефона в международном формате, а именно в формате Республики Беларусь. Дадим соглашение на обработку персональных данных (рисунок 13).

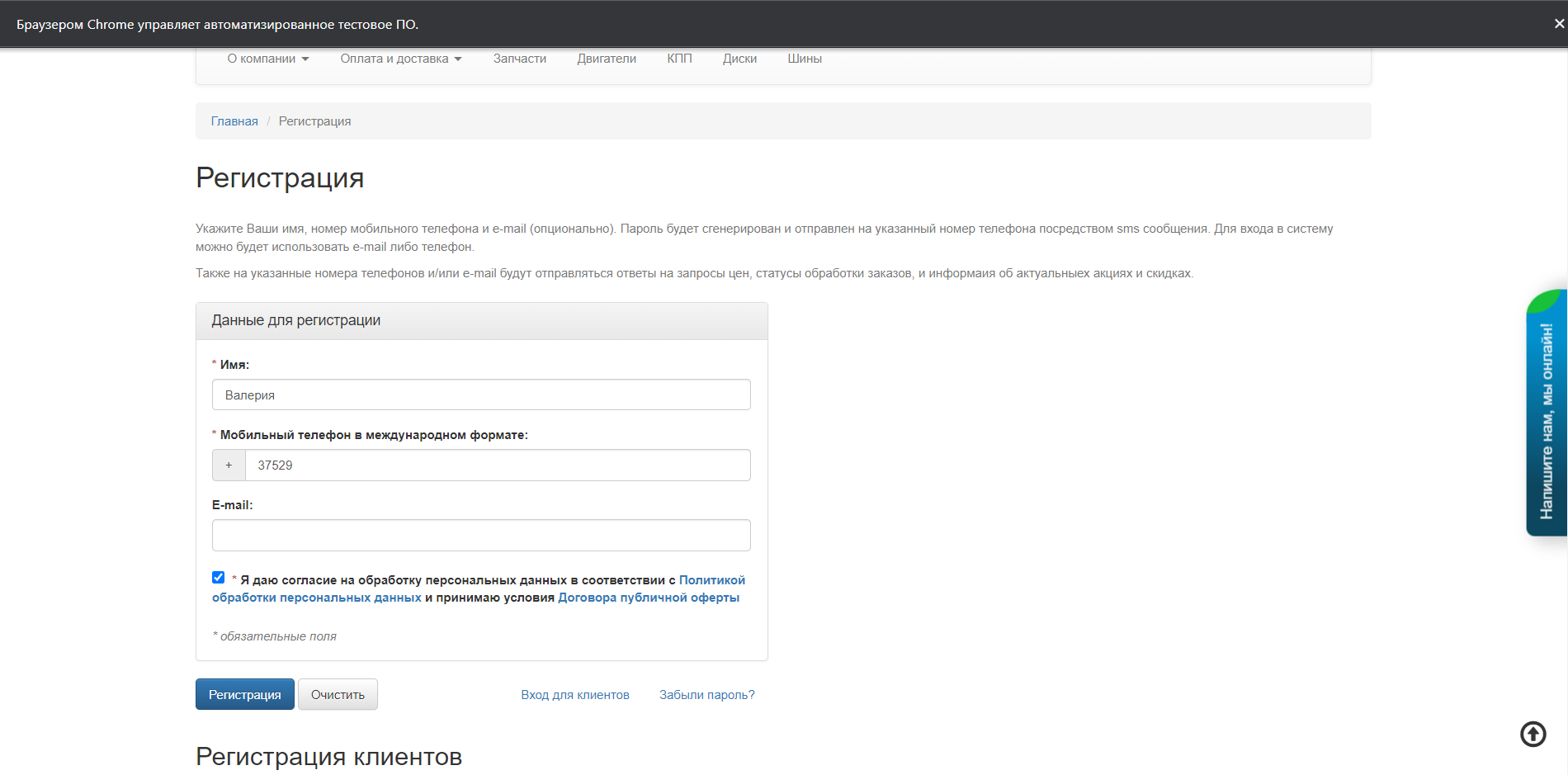


Рисунок 13 – Экран успешной регистрации

Затем на указанный номер телефона пришло смс сообщение со случайно сгенерированным паролем. С его помощью пользователь сможет войти в систему. Проверим возможность авторизации.

Введем пустые поля и нажмем кнопку «Войти» (рисунок 14).

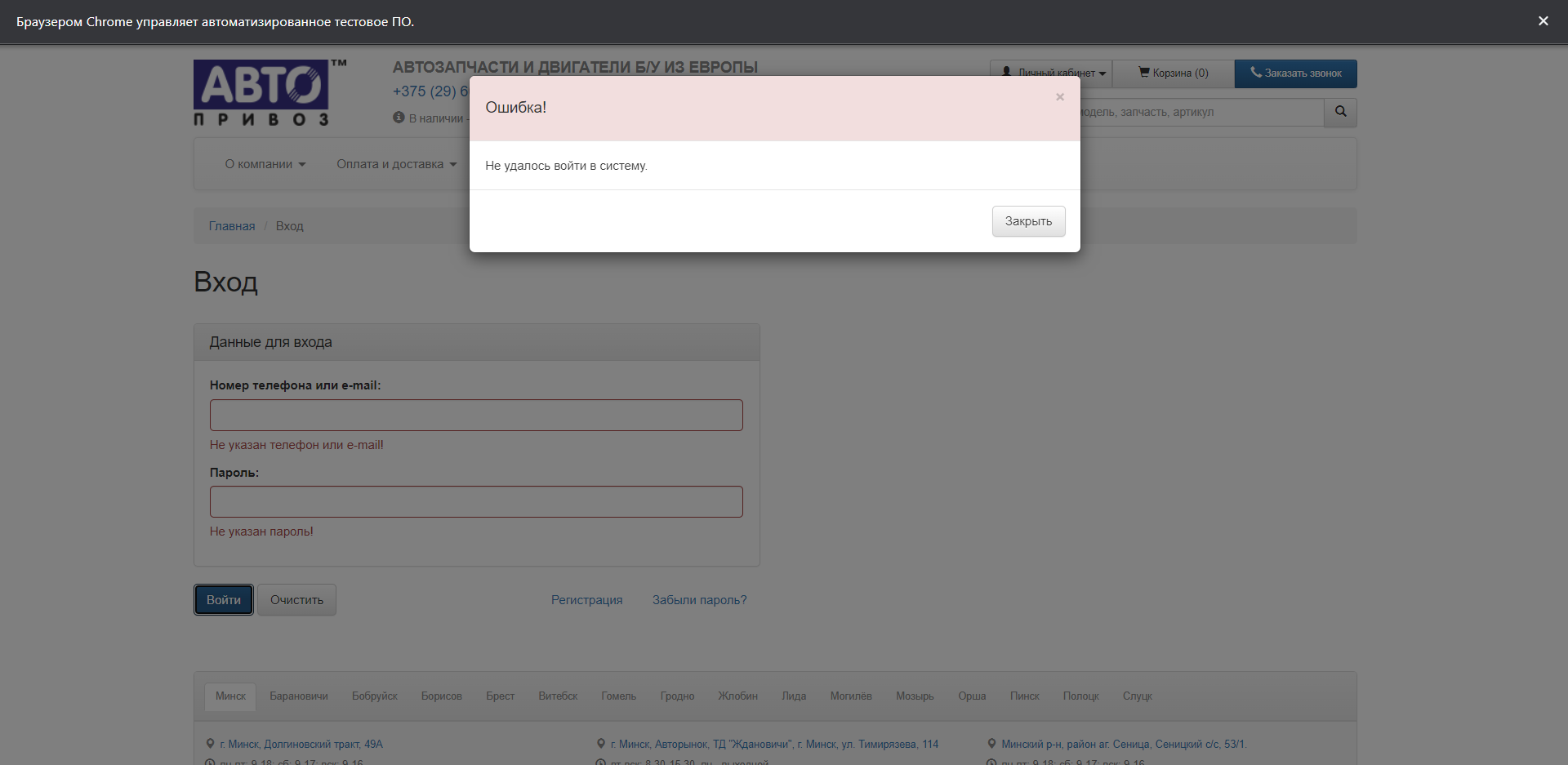


Рисунок 14 – Авторизация без данных

Как видно из рисунка выше вход в систему не выполняется, появляется окно предупреждения.

Выполним вход, не введя пароль. Как видно из рисунка 15, вход не выполнится, появится окно предупреждения.

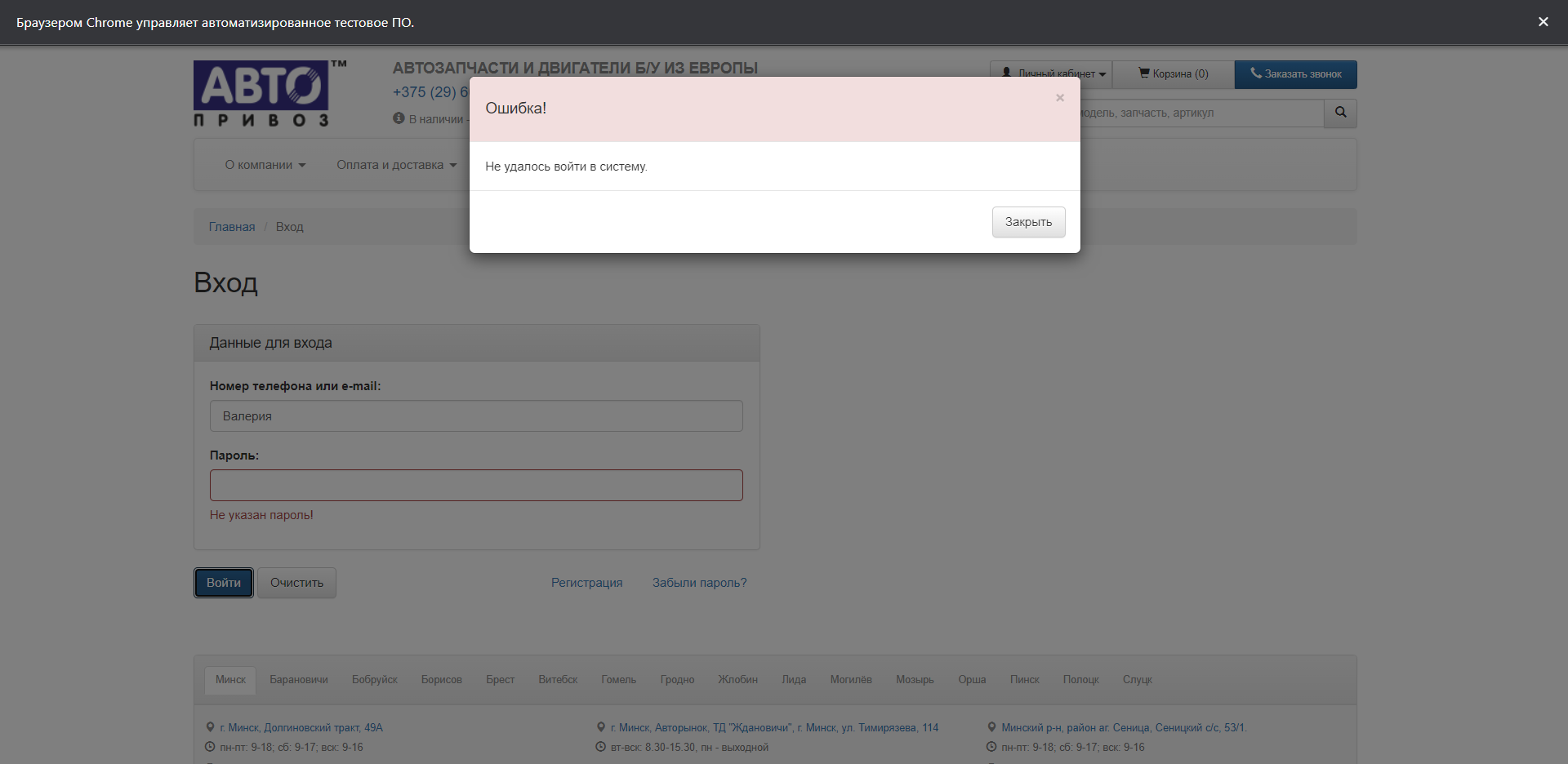


Рисунок 15 – Вход без пароля

Введем правильный номер телефона, но неверный пароль (рисунок 16).

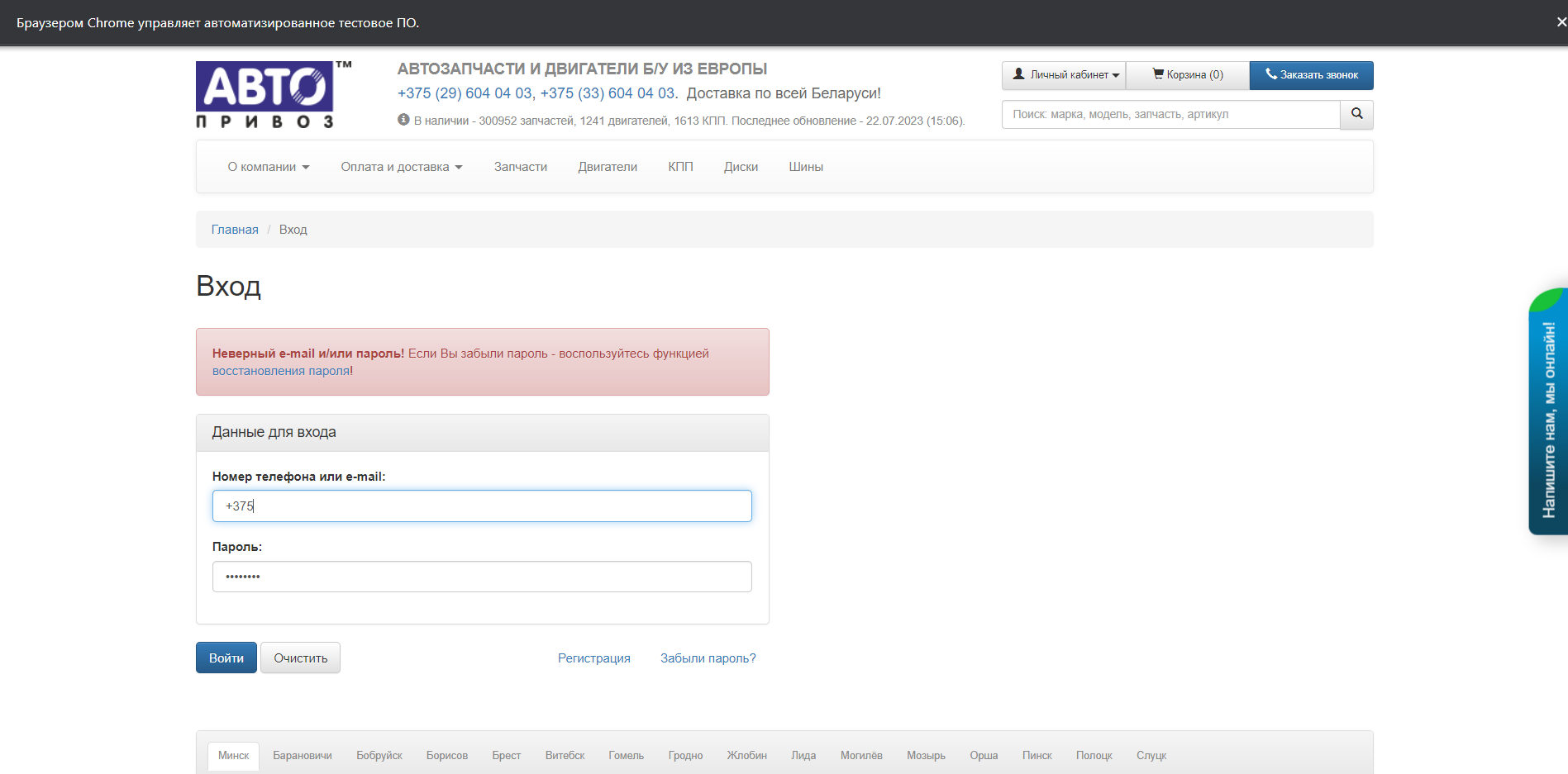


Рисунок 16 – Вход с неверным паролем

Как видно из рисунка выше при вводе неверного пароля появляется предупреждение «Неверный e-mail и/или пароль».

# **Заключение**

В ходе прохождения производственной практики в компании «Автотега» были изучены принципы разработки автоматизированного тестирования и освоино использование языка программирования Java в сочетании с инструментом автоматизации тестирования Selenium. В результате работы были разработаны тесты, которые проверяют основной функционал сайта компании «Автотега».

Были ознакомлены с основными принципами автоматизации тестирования и понятиями, такими как Page Object Model (POM), тест-кейсы, локаторы элементов, явное ожидание и т. д. Были изучены основы работы с Selenium WebDriver, который предоставляет возможность управления веб-браузером для автоматизированного тестирования веб-приложений.

Был создан проект автоматизированного тестирования и настроен с помощью средств сборки проектов, таких как Maven.

Были разработаны тестовые сценарии, которые проверяют основной функционал сайта компании «Автотега», например, регистрацию пользователя, поиск товаров и т. д. Для написания тестов использовался Selenium WebDriver для управления браузером и выполнения действий на веб-страницах, а также использовались различные методы ожидания, чтобы синхронизировать действия с динамическим содержимым страницы.

Для тестирования были созданы тестовые данные, которые используются в тестах для воспроизведения различных сценариев. Тестовое окружение было настроено таким образом, чтобы тесты запускались на определенных браузерах (Chrome) для проверки кросс-браузерной совместимости сайта.

Результатом работы является проект по тестированию сайта компании. Прохождение производственной практики в компании «Автотега» предоставило ценный опыт в области разработки автоматизированных тестов и позволит применить полученные знания на практике для тестирования будущих веб-приложений.

# **Список использованных источников**

1. Руководство по Java [Электронный ресурс]. Режим доступа: https://metanit.com/java/tutorial/. Дата доступа: 12.07.2023.

2.  Руководство по Selenium Webdriver [Электронный ресурс]. Режим доступа: https://comaqa.gitbook.io/selenium-webdriver-lectures/selenium-webdriver.-vvedenie/webdriver.-obzor-i-princip-raboty. Дата доступа: 12.07.2023.

3.  Руководство по Intellij Idea [Электронный ресурс]. Режим доступа: https://blog.jetbrains.com/ru/. Дата доступа: 12.07.2023.

# **Приложение А**

Класс LoginPage

public class LoginPage extends Page{

public LoginPage(WebDriver driver) {

super(driver);

}

private String URL = "https://www.autopriwos.by/login.html";

@FindBy(id = "\_submit\_button")

private WebElement submitButton;

@FindBy(id = "\_clear\_button")

private WebElement clearButton;

public LoginPage login(String phoneNumber, String password, boolean submit, boolean clear) {

driver.get(URL);

WebElement phoneNumberInput = driver.findElement(By.id("data[email]"));

waitForVisibilityOfElement(driver, phoneNumberInput);

phoneNumberInput.sendKeys(phoneNumber);

WebElement passwordInput = driver.findElement(By.id("data[password]"));

passwordInput.sendKeys(password);

if(submit) {

submitButton.click();

}

if(clear) {

clearButton.click();

}

return this;

}

Класс RegistrationPage

public class RegistrationPage extends Page {

public RegistrationPage(WebDriver driver) {

super(driver);

}

private String URL = "https://www.autopriwos.by/register.html";

@FindBy(id = "\_submit\_button")

private WebElement submitButton;

@FindBy(id = "\_clear\_button")

private WebElement clearButton;

public RegistrationPage register(String name, String number, String email, Boolean agreement, Boolean submit, Boolean clear) {

driver.get(URL);

waitForVisibilityOfElement(driver, submitButton);

WebElement nameInput = driver.findElement(By.id("data[name]"));

nameInput.sendKeys(name);

WebElement numberInput = driver.findElement(By.id("data[phone]"));

numberInput.sendKeys(number);

if(agreement) {

WebElement agreeCheckbox = driver.findElement(By.name("data[agree]"));

agreeCheckbox.click();

}

if(submit) {

submitButton.click();

}

if(clear) {

clearButton.click();

}

return this;

}

Класс SearchPage

public class SearchPage extends Page {

public SearchPage(WebDriver driver) {

super(driver);

}

private final String URL = "https://www.autopriwos.by";

@FindBy(linkText = "Текстовый поиск")

private WebElement textSearch;

@FindBy(id = "filter[textsearch]")

private WebElement searchField;

@FindBy(id = "search\_button")

private WebElement searchButton;

@FindBy(id = "search\_clear\_button")

private WebElement clearSearchFieldButton;

public SearchPage Search(String searchText) throws InterruptedException {

driver.get(URL);

waitForVisibilityOfElement(driver, textSearch);

textSearch.click();

searchField.sendKeys(searchText);

searchButton.click();

return this;

}

public SearchPage extendedSearch(String brand, String model, String detail,

String article, Boolean withPhoto, String constrn,

String engineCapacity) {

driver.get(URL);

WebElement extendedButton = driver.findElement(By.className("icon-spacer"));

waitForVisibilityOfElement(driver, extendedButton);

extendedButton.click();

Select brandSelect = new Select(driver.findElement(By.id("filter[auto\_brand]")));

brandSelect.selectByVisibleText(brand);

WebElement modelInput = driver.findElement(By.id("filter[auto\_model]"));

modelInput.sendKeys(model);

Select detailSelect = new Select(driver.findElement(By.id("filter[part\_type]")));

detailSelect.selectByVisibleText(detail);

WebElement articleInput = driver.findElement(By.id("filter[article]"));

articleInput.sendKeys(article);

if (withPhoto) {

WebElement withPhotoCheckbox = driver.findElement(By.id("filter[with\_photo]"));

withPhotoCheckbox.click();

}

WebElement constrnInput = driver.findElement(By.name("filter[constrn]"));

constrnInput.sendKeys(constrn);

Select engineCapacitySelect = new Select(driver.findElement(By.id("filter[engine\_capacity]")));

engineCapacitySelect.selectByVisibleText(engineCapacity);

searchButton.click();

return this;

}

public SearchPage ClearSearchField() throws InterruptedException {

clearSearchFieldButton.click();

return this;

}

# **Приложение Б**

Класс TextSearchTest

public class TextSearchTest extends CommonConditions {

private SearchPage searchPage;

@BeforeMethod

public void browserSetup() throws InterruptedException {

searchPage = new SearchPage(driver);

}

@Test

public void TextSearch() {

try {

searchPage.Search(""); // empty search

searchPage.Search("54295850"); // search by article

searchPage.Search("Kia"); // search by brand

searchPage.Search("Carnival"); // search by car model

searchPage.Search("Двигатель"); // search by car detail

searchPage.ClearSearchField();

} catch (InterruptedException e) {

throw new RuntimeException(e);

}

}

}}

Класс ExtendedSearchTest

public class ExtendedSearchTest extends CommonConditions {

private SearchPage searchPage;

@BeforeMethod

public void browserSetup() {

searchPage = new SearchPage(driver);

}

@Test

public void extendedSearch() throws InterruptedException {

searchPage.extendedSearch("Mercedes", "W203 (CLC)", "Насос вакуумный",

"54046908", true, ".", "2.2");

}

}

Класс RegistrationTest

public class RegistrationTest extends CommonConditions {

private RegistrationPage registrationPage;

@BeforeMethod

public void browserSetup() {

registrationPage = new RegistrationPage(driver);

}

@Test

public void registration() {

registrationPage.register("", "", "",true, true, false);

registrationPage.register("", "+375290000000", "someemail@mail.ru",true, false, true);

registrationPage.register("Валерия", "", "someemail@mail.ru", true, false, true);

registrationPage.register("Валерия", "+375290000000", "someemail@mail.ru", true, false, true);

registrationPage.register("Валерия", "+375290000000", "someemail@mail.ru", false, true, false);

}

}

Класс LoginTest

public class LoginTest extends CommonConditions {

private LoginPage loginPage;

@BeforeMethod

public void browserSetup() {

loginPage = new LoginPage(driver);

}

@Test

public void login() {

loginPage.login("", "", true, false);

loginPage.login("", "12345678", true, false);

loginPage.login("+3750000000", "12345678", false, true);

loginPage.login("+3750000000", "12345678", true, false);

}

}