Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий и анализа данных |

наименование института

|  |
| --- |
| Допускаю к защите  Руководитель:  В.Л. Аршинский |
| И.О. Фамилия |

Распознавание и регистрация идентификационных номеров полувагонов наименование темы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

|  |
| --- |
| Объектно-ориентированное программирование |

|  |
| --- |
| 1.020.00.00 ПЗ |
| обозначение документа |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | ИСМб 19-1 |  |  |  | Солопов Д.Д. |
|  | шифр группы |  | подпись |  | Фамилия И.О |
| Нормоконтроль |  |  |  |  | Аршинский В.Л. |
|  |  |  | подпись |  | Фамилия И.О |

|  |  |
| --- | --- |
| Курсовая работа защищена с оценкой |  |

Иркутск 2021 г.

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ЗАДАНИЕ**

**НА КУРСОВОЙ ПРОЕКТ**

|  |  |  |
| --- | --- | --- |
| По курсу | Объектно-ориентированное программирование | |
| Студенту | Солопову Даниилу Дмитриевичу | |
|  | (фамилия, инициалы) | |
| Тема работы: | Распознавание и регистрация идентификационных номеров полувагонов | | |
| Исходные данные: | | Вариант 20 – Распознавание и регистрация идентификационных номеров полувагонов | |
| Рекомендуемая литература: | | | |
| 1. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированный анализ и проектирование с примерами приложений. Третье издание. М.: "Вильямс", 2010, -720 с.  2. Васильев А. Н. Java. Объектно-ориентированное программирование: для магистров и бакалавров. Базовый курс по объектно-ориентированному программированию / А. Н. Васильев . – СПб.: Питер, 2012. – 395 с. | | | |
|  | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Графическая часть на | - | | листах. | |
| Дата выдачи задания |  | 23 / 11 / 2021 г. | | |
| Задание получил студент | | | |  | |  | Солопов Д.Д. |
|  | | | | подпись | |  | Фамилия И.О. |

|  |  |
| --- | --- |
| Дата представления работы руководителю | 21 / 12 / 2021 г. |
| Руководитель курсовой работы |  |  | Аршинский В.Л. |
|  | подпись | Фамилия И.О. |

Содержание

[Введение 4](#_Toc71684895)

[Перечень условных обозначений и терминов 6](#_Toc71684896)

[Анализ задания и описание предметной области 8](#_Toc71684897)

[Словарь существительных и глаголов 14](#_Toc71684898)

[Проектирование приложения 18](#_Toc71684899)

[Реализация приложения 26](#_Toc71684900)

[Реализация модуля “Центральный сервер” 26](#_Toc71684901)

[Описание реализованного модуля 26](#_Toc71684902)

[Описание реализованных классов 27](#_Toc71684903)

[Спецификация методов реализованных классов 32](#_Toc71684904)

[Тестирование приложения 48](#_Toc71684905)

[Тестирование модуля “Центральный сервер” 48](#_Toc71684906)

[Выбор методики тестирования 48](#_Toc71684907)

[Таблица тестов 49](#_Toc71684908)

[Результаты тестирования 56](#_Toc71684909)

[ссылки 63](#_Toc71684910)

Введение

В настоящее время информационные технологии активно используются в разных сферах жизнедеятельности людей. Наиболее значительную роль информационные технологии играют при замене ручного труда машинным. Существует множество производств, в которых используются компьютеры для моделирования каркасов автомобилей, само их производство (сборка), тестирование моделей, и многое другое. Эффективность и продуктивность процесса производства с помощью компьютерных технологий намного выше, чем при производстве ручным трудом [3].

Широкое распространение получили системы визуальной аналитики, которые способствуют получению информации об объектах [4]. Для примера стоит взять ситуацию, когда необходимо вести учёт автомобилей, которые въезжают и выезжают на парковку для сотрудников определённой компании. Установленная камера будет считывать номера автомобилей и регистрировать прибытие того или иного транспортного средства. Это значительно облегчает труд охранников парковки, поскольку им нет необходимости стоять на улице значительное время и проводить учёт автомобилей. С данной задачей справляется камера и программа, реализующая распознавание средствами алгоритмов машинного обучения или нейронных сетей.

Системы визуальной аналитики достаточно распространены в части регистрации идентификационных номеров вагонов (или полувагонов). Например, существует система SecurOS Transit[7], которая позволяет анализировать номер полувагона с изображения и записывать в базу данных распознанный номер вагона. Данная система может быть использована не только на железной дороге, но и в морских портах, на промышленных предприятиях, на таможенном и пограничном контроле. Существует также SDK для встраивания систем распознавания в свой проект. Например, SDK INTLAB WAGON[2] предоставляет инструменты для анализа изображений содержащих номера вагонов. Данная SDK ориентирована исключительно на номера вагонов и примеры распознавания на официальном сайте данного SDK позволяют оценить этот инструментарий как достаточно надёжный.

В данном курсовом проекте будет осуществлена работа с компьютерным зрением. Как научная дисциплина, компьютерное зрение относится к теории и технологии создания искусственных систем, которые получают информацию из изображений. Компьютерное зрение чаще всего используется в следующих сферах человеческой деятельности[6]:

1) системы видеонаблюдения в офисах, на производстве, в торговых центрах, на улицах;

2) системы управления движущимися машинами, предотвращающие столкновения с препятствием;

3) чтение штрих кодов в торговле и на складских комплексах;

4) технологии дополненной и виртуальной реальности;

5) конвертация бумажных книг и документов в цифровые форматы

Компьютерное зрение с недавнего времени начало использоваться с системой аутентификации Face ID, позволяющая подтверждать личность для разблокировки мобильного устройства и выполнения операций, которые изначально могут быть заблокированы (например, взаимодействие с кредитной картой).

В западных странах компьютерное зрение несколько лет используется для оценки дорожных ситуаций: пробок, загруженности полос. Анализируются целые сцены и ситуации, связанные с чтением дорожных знаков и скоростью их распознавания.

С помощью технологий компьютерного зрения можно разработать информационные системы, способные распознавать текст, идентифицировать предметы и людей, оценивать движения, восстанавливать изображения, выделять на них однородные структуры и элементы, анализировать оптические потоки. Оператор может получить конкретные части изображения, который представляют особый интерес. Например, после анализа изображения полувагона особый интерес вызывает идентификационный номер полувагона, который программа может считать и сохранить в виде данных, для последующей регистрации фактического прибытия на место полувагона.

Регистрация и распознавание номеров достаточно интересная задача. Разработка систем, реализующих решение данной или подобных задач позволяет получить практический опыт разносторонней разработки программного обеспечения. Периодическое изменение направлений деятельности в рамках сферы программирования положительно сказывается на разработчике, поскольку развитый широкий кругозор позволяет решать задачи более разнообразными методиками, а также влияет на качество сгенерированных идей, которые могут быть реализованы.

**Предметной областью** данного курсового проекта является железная дорога, а **объектом исследования** – полувагон, который перемещается из одной точки в другую посредством железнодорожных путей.

**Цель данного курсового проекта** состоит в увеличении скорости разгрузки полувагонов и автоматизированном информировании уполномоченных лиц о прибытии определённого вещества, заранее определённого по накладной.

**Задачи:**

1) Исследование и анализ предметной области.

2) Проектирование информационной системы.

3) Практическая реализация информационной системы.

4) Тестирование информационной системы.

5) Анализ результатов тестирования и подведение итогов.

**Практическая значимость** состоит в увеличении скорости разгрузки полувагонов по приезду на конечную остановку и делегирование на автоматизированную систему решения задач, связанных с распознаванием номеров полувагонов и занесением информации в базу данных.

Перечень условных обозначений и терминов

**Железная дорога** - транспортная трасса постоянного действия, отличающаяся наличием пути (или путей) из закрепленных рельсов, по которым ходят поезда, перевозящие пассажиров, багаж, почту и различные грузы. Понятие "железная дорога" включает в себя не только подвижной состав (локомотивы, пассажирские и грузовые вагоны и т.п.), но и полосу отчуждения земли со всеми сооружениями, постройками, имуществом и правом провоза товаров и пассажиров по ней.

**Грузовая железнодорожная станция** — раздельный пункт сети железных дорог, выполняющий грузовые и коммерческие операции с грузами и грузовыми вагонами, связанные с приёмом к перевозке, взвешиванием, хранением, погрузкой, выгрузкой, сортировкой и выдачей грузов, переработкой контейнеров, оформлением перевозочных документов, формированием передаточных грузовых поездов и отправительских маршрутов, производством маневровой работы по подаче вагонов на грузовые фронты и их уборке, а также с другими техническими операциями.

**Железнодорожный транспорт** – вид транспорта, осуществляющий перевозки грузов по рельсовым путям в вагонах (поездах) с помощью локомотивной тяги.

**Компьютерное зрение (иначе техническое зрение**) — теория и технология создания машин, которые могут производить обнаружение, отслеживание и классификацию объектов.

**Spring Framework (или коротко Spring**) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET.

**OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом)** — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков. Может свободно использоваться в академических и коммерческих целях — распространяется в условиях лицензии BSD.

**Модель (фр. modèle от лат. modulus «мера, аналог, образец**») — система, исследование которой служит средством для получения информации о другой системе; представление некоторого реального процесса, устройства или концепции.

**Логическая модель представления знаний** — модель в представлении знаний.

**Представление знаний** — вопрос, возникающий в когнитологии (науке о мышлении) и информатике, а также в исследовании вопросов, связанных с искусственным интеллектом. В когнитологии он связан с тем, как люди хранят и обрабатывают информацию. В информатике — с подбором представления конкретных и обобщённых знаний, сведений и фактов для накопления и обработки информации в ЭВМ. Главная задача в искусственном интеллекте (ИИ) — научиться хранить знания таким образом, чтобы программы могли осмысленно обрабатывать их и достигнуть тем подобия человеческого интеллекта.

**Проектирование** — процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части (ISO 24765). Результатом проектирования является проект — целостная совокупность моделей, свойств или характеристик, описанных в форме, пригодной для реализации системы.

**MySQL (МФА: [maɪ ˌɛskjuːˈɛl])** — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

**База данных** — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

**Предприятие промышленного железнодорожного транспорта (ППЖТ)** — обособленное транспортное предприятие, либо отдельный транспортный цех завода, шахты, фабрики, электростанции или иного промышленного объекта, в задачу которого входит транспортировка промышленных грузов (продукции, сырья, отходов производства) с одного промышленного объекта на другой, либо, чаще всего, от объекта до станции, входящей в сеть РЖД.

**Вагоноопрокидыватель** — специальное сооружение для механизированной разгрузки вагонов с насыпными и навалочными грузами (рудой, углем, зерном).

Анализ задания и описание предметной области

Понятие железная дорога обозначает оборудованную рельсами полосу земли либо поверхности искусственного сооружения (тоннель, мост, эстакада), которая используется для движения рельсовых транспортных средств. Железная дорога может состоять из одного пути или нескольких. Железные дороги бывают с электрической, дизельной, турбинной, паровой или комбинированной тягой. Особый вид железных дорог — зубчатые. Обычно железные дороги оборудуются системой сигнализации, а железные дороги на электрической тяге — также контактной сетью. Различают железные дороги общего пользования, промышленные железные дороги (подъездные пути предприятий и организаций) и городские железные дороги — метрополитен и трамвай.

Интересуемый вид железной дороги в данном курсовом проекте являются промышленные железные дороги.



Рисунок 1 – Тепловоз (слева) и электровоз (справа) на железнодорожных путях



Рисунок 2 – Железнодорожные пути



Рисунок 3 – Полувагон

На рисунке 3 представлен полувагон. Данные вагоны используются для транспортировки груза не требующих защиты от атмосферных осадков. Кузов полувагона не имеет крыши, что обеспечивает удобства для использования различных средств механизации при погрузке и выгрузке грузов (мостовые и козловые краны, вагоноопрокидыватели и др.). Все универсальные полувагоны имеют люки в металлическом полу для разгрузки сыпучих грузов гравитационным способом.

В России первые полувагоны появились в 1861 году, это были заказанные из-за границы вагоны для Грушевской железной дороги, занимавшейся перевозкой угля. Впоследствии русские заводы скопировали конструкцию и стали изготавливать вагоны самостоятельно. Также переоборудовались под полувагоны крытые вагоны малой ёмкости. Эти вагоны имели малый объём кузова (10 м³) и грузоподъёмность 8,5—10 тонн. В 1890—1900 годах для углевозных дорог было построено большое количество полувагонов. Они имели грузоподъёмность 12,5—15 тонн и имели деревянный кузов и высоту бортов от 70 до 120 см. В Домбровском угольном районе строились и цельнометаллические полувагоны.

Полувагоны перемещаются из одной точки в другую и конечным пунктом назначения, обычно, является завод или промышленное предприятие, которой необходимо определённое вещество. По прибытии полувагоны подлежат осмотру (в том числе, для обнаружения идентификационного номера полувагона и сопоставление его по определённой накладной) и последующей разгрузке с помощью вагоноопрокидывателя.

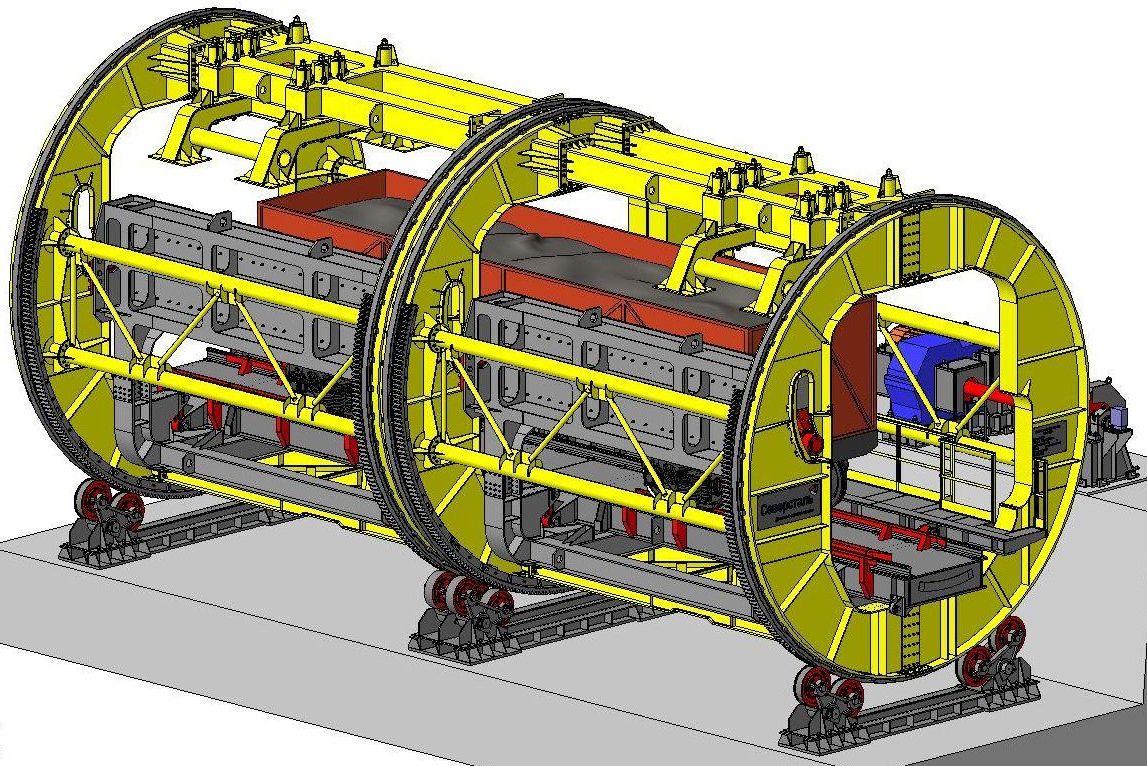


Рисунок 4 – Модель вагоноопрокидывателя

На рисунке 4 представлена модель вагоноопрокидывателя. По данной можно увидеть, каким образом вагоны помещаются в него и каким образом переворачиваются. Вагоны заезжают в специальное отверстие, спроектированное для полувагонов разной длины и размеров, затем, их закрепляют и переворачивают вращательным движением вагоноопрокидывателя за счёт автоматизированных механизмов.



Рисунок 5 – Вагоноопрокидыватель



Рисунок 6 – Вагоноопрокидыватель с двумя въездами

На рисунках 5-6 представлены вагоноопрокидыватели, которые используются на производственных предприятиях.

Существуют промышленные предприятия, которые устанавливают на определённом расстоянии от вагоноопрокидывателя видеокамеры, с возможностью транслирования видеопотока удалённой информационной системе. Вагоны приходят составом, состоящий из определённого числа вагонов. На рисунке 7 представлены камеры, расположенная на определённом расстоянии от разгрузочной/загрузочной станции, для идентификации номера полувагона.



Рисунок 7 – Камеры, установленные для распознавания номеров полувагонов

Наибольший интерес при распознавании изображения полувагона представляет идентификационный номер полувагона, который выделен белыми цифрами определённого размера (см. рис. 8).



Рисунок 8 – Полувагон с идентификационным номером

Идентификационный номер может сопоставлять определённый полувагон накладной, в которой содержится основная информация о полувагоне.

Словарь существительных и глаголов

**Подвижной состав на сети железных дорог** России — совокупность подвижного состава, находящегося в собственности или в оперативном управлении компании Российские железные дороги, других компаний-операторов подвижного состава.

**Полувагон** — железнодорожный грузовой вагон с кузовом без крыши, предназначенный для перевозки грузов, не требующих защиты от атмосферных осадков.

**Видеокамера** — первоначальное значение — комбинация телевизионной передающей камеры и устройства для видеозаписи.

**Цифровое изображение** — двумерное изображение, представленное в цифровом виде. В зависимости от способа описания, изображение может быть растровым или векторным.

**Оптическое распознавание символов (англ. optical character recognition, OCR)** — механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные, использующиеся для представления символов в компьютере (например, в текстовом редакторе). Распознавание широко применяется для преобразования книг и документов в электронный вид, для автоматизации систем учёта в бизнесе или для публикации текста на веб-странице. Оптическое распознавание символов позволяет редактировать текст, осуществлять поиск слов или фраз, хранить его в более компактной форме, демонстрировать или распечатывать материал, не теряя качества, анализировать информацию, а также применять к тексту электронный перевод, форматирование или преобразование в речь. Оптическое распознавание текста является исследуемой проблемой в областях распознавания образов, искусственного интеллекта и компьютерного зрения.

Функциональные требования

С помощью компьютерного зрения необходимо автоматизировать процесс идентификации полувагонов, которые прибыли по накладной, определённого вида (см. рис. 9). Каждый полувагон содержит в себе сыпучую смесь некоторого процентного содержания (обозначающаяся как S^d - %).

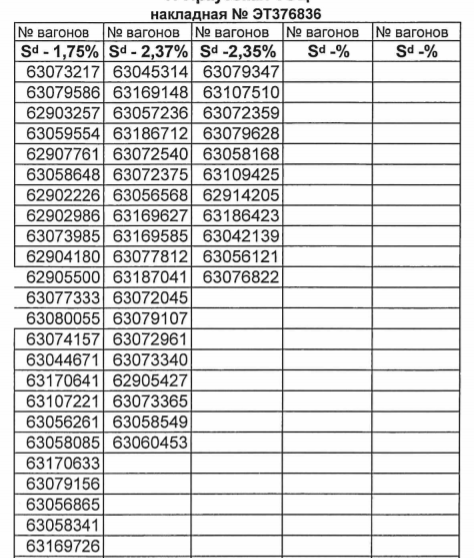


Рисунок 9 – Пример накладной

Накладная поступает на производственное предприятие заранее, то есть, до приезда полувагонов. Вагоны прибывают в хаотичном порядке, однако принадлежат определённому составу.

Таким образом, уполномоченные лица знают о составе некоторые подробности, в том числе – процентное содержание сыпучей смеси в полувагоне.

Необходимо достичь уровня автоматизации, при которого каждый приезжий полувагон будет зафиксирован и разгружен.

Есть необходимость сохранять информацию о приезжающих и разгружаемых полувагонах, по определённой накладной, обусловленная в большей степени отчётностью и возможностью последующего статистического анализа (например, анализ эффективности отправки состава из одной точки страны в другую), но и может в себя включать проверку на корректное контактирование различных предприятий между собой (например, прогнозирование ошибок в накладных, преждевременное их обнаружение или профилактика).

Информационная система должна быть разбита на три модуля:

1) “Центральный сервер” – единственный модуль, который имеет доступ к базе данных. Модуль должен обеспечивать взаимодействие с базой данных других модулей, которым необходимо получать доступ к актуальным данным, расположенным в базе данных для работоспособности.

2) Модуль “Камера”, занимающийся распознаванием изображения, которое должно загружаться пользователем.

3) Модуль “Пользователь” обеспечивает пользовательский интерфейс, с помощью которого можно рассмотреть информацию, содержащуюся в базе данных, добавить накладную и информацию о ней, добавить информацию о полувагонах содержащиеся в определённой накладной.

Модуль “Камера” и “Пользователь” относятся к клиентской части приложения, а модуль “Центральный сервер” к серверной.

Необходимо производить оценку корректности распознавания, для уведомления уполномоченных лиц, в случае плохого распознавания об ошибках для возможности улучшения работы информационной системы и ведения общей статистики по результативности корректного обнаружения и анализа номеров полувагонов в будущем. Помимо информирования о плохом распознавании, должна быть возможность исправить некорректно распознанный номер полувагона корректным и обновить данные в базе данных.

В пользовательском интерфейсе необходимо предоставлять информацию о определённом полувагоне, выбранном пользователем из всех имеющихся (зарегистрированных). С помощью пользовательского интерфейса оператор, который оценивает общую эффективность рабочего процесса, должен иметь возможность менять поведение информационной системы: оператор должен вносить информацию о накладной, а также информацию о полувагонах соответствующих данной накладной в элемент управления. Таким образом, часть информационной системы занимающейся распознаванием будет фиксировать полувагоны, приезжающие уже по определённой конкретной накладной. Сотрудник должен вмешиваться в процесс распознавания только в одном случае – если существует необходимость изменить некорректно распознанный номер. Самостоятельно добавлять информацию о полувагоне запрещено. Необходимо учесть вывод предупреждений оператору при слишком низком уровне успешного распознавания или ошибок в системе, для уменьшения возможных финансовых потерь при некорректных попытках распознавания.

Необходимо развертывать базу данных на сервере посредством SQL-запросов для увеличения кроссплатформенности серверной части приложения и возможности работы в непредвиденных обстоятельствах в лице аппаратного обеспечения ранее не используещегося.

Постановка задач

1) Спроектировать модуль “Центральный сервер” и практически реализовать функционал данного модуля, описанный в функциональных требованиях.

2) Спроектировать модуль “Камера” и практически реализовать функционал данного модуля, описанный в функциональных требованиях.

3) Спроектировать модуль “Пользователь” и практически реализовать функционал данного модуля, описанный в функциональных требованиях.

Проектирование приложения

Проектирование модели базы данных

Перед проектированием логической, физической модели базы данных и ER-диаграммы, имеется необходимость спроектировать процесс взаимодействия данных, в самой проектируемой базе данных для способствования визуальной и интуитивной интерпретации идеи проектирования, которая была предложена.

При прибытии полувагона на место, где располагается камера, будут считаны данные о полувагоне, имеющие следующую структуру:

1. Уникальный идентификатор полувагона.

2. Дата прибытия полувагона

3. Уровень корректного распознавания.

4. Путь к распознанному изображению, хранящемуся на сервере локально.

Данная структура включает в себя основную информацию, которую мы получаем о полувагоне при первом визуальном контакте. Никакой другой информации для решения текущих поставленных задач не требуется.

Далее, после занесения информации о полувагоне в таблицу содержащую информацию о всех прибывших полувагонах, в таблице, содержащей все полувагоны по определённой накладной, должно происходить изменение отметки о прибытии (с “не прибыл” на “прибыл”) и вычисление фактического порядка полувагона, относительно исходного списка всех полувагонов по данной накладной. Таблица “Регистрация” вбирает в себя информацию о всех уникальных идентификаторах полувагонов, прибывающих по одной конкретной накладной.

При прибытии накладной, на конечное место приезда каждого отдельного полувагона по данной накладной, сотрудник, имеющий определённые полномочия к данной информации, имеет возможность внести уникальные идентификационные номера полувагонов в таблицу “Регистрации”, для каждой отдельной накладной. По одной накладной может прибыть любое число полувагонов, с разными или одинаковыми показателями S%-d (процент сыпучести груза).

Сущность “Накладные” никак не взаимодействует с сущностью “Полувагоны”, поскольку сущность “Полувагоны” решает задачу только фиксации определённой информации о полувагоне, которую можно будет получить через сущность “Регистрации”, осуществляя поиск по тем записям, где уникальный идентификатор присутствует и также присутствует отметка о прибытии этого полувагона. Уникальные идентификаторы полувагонов известны ещё до прибытия самих полувагонов – они содержаться в накладных, которые прибывают раньше полувагонов. Информация о полувагонах доступна сразу после получения полувагонов. Однако, общий вид полувагона будет получен только по фактическому прибытию полувагона.

Сущность “Регистрация” содержит внешний ключ на сущность “Накладные”, поскольку каждая конкретная накладная обязательно должна содержать информацию о общем числе полувагонах, их идентификаторах, порядковых номерах в составе, проценте сыпучести груза (S%-d) и другое. Таким образом, отношения между сущностью “Накладные” и “Регистрация” можно оценить как 1:N по степени связи и Н … О по классу принадлежности (множество сущностей “Накладные” содержится в одной сущности “Регистрация”).

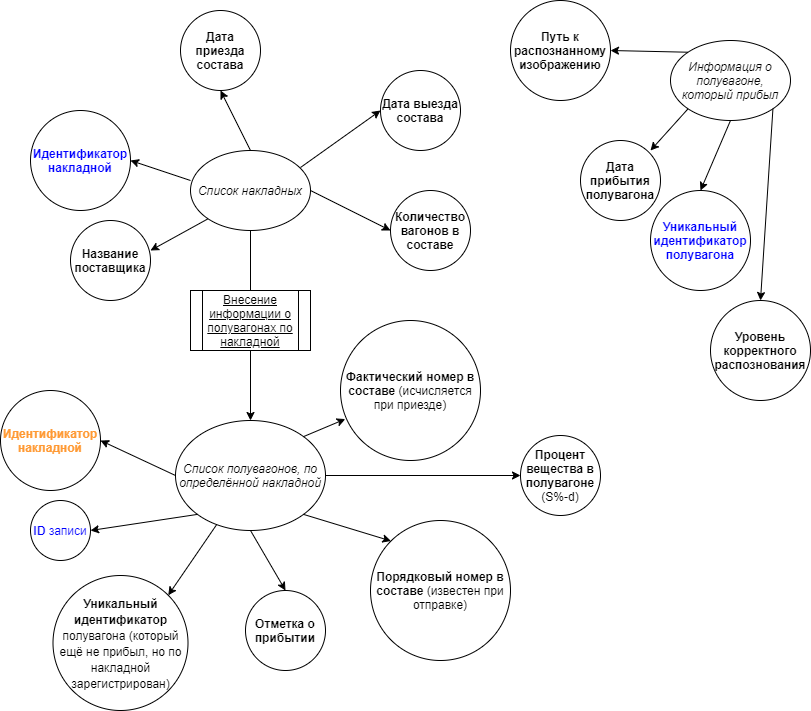


Рисунок 1 – Логическая модель базы данных

Рисунок 2 – ER-Диаграмма

Для проектирования физической модели базы данных, следует оценить эффективность использования того или иного типа данных. Так как будет использоваться база данных MySQL, то отталкиваться имеет смысл именно от ограничений на типы данных именной данной базы данных.

**Рассмотрим сущность “Полувагоны”:**

Для начала, стоит ознакомиться с информацией, относительно актуального количества полувагонов в Российской Федерации на момент 2020-2021 года.

Количество общего парка полувагонов на сети РЖД в июле 2020 года выросло на 3,7% и достигло 565 641 ед. (год назад аналогичный показатель составил 545 428 ед.). Рабочий парк полувагонов за год увеличился на 1,6%, с 492 207 ед. в июле 2019-го до 499 959 ед. в июле 2020-го. Динамика роста отмечается в сегменте прочих видов вагонов: общий парк в июле 2020-го составил 247 345 ед., что на 6,9% выше июля 2019 года (231 450 ед.), при этом рабочий парк в годовой динамике также вырос на 5,8% с 187 306 до 198 112 ед. [5]

С помощью статистических данных, приведённых выше, можно сделать вывод о том, что использование целочисленного типа INT для представления номеров полувагонов вполне обосновано, т.к. общая численность полувагонов не превышает даже 1 млн., а идентификационный номер каждого отдельного полувагона состоит, как правило, из 8 цифр, в то же время тип данных INT представляет целые числа от -2147483648 до 2147483647, занимает 4 байта (в базе данных MySQL).

Для хранения даты прибытия полувагона на место остановки будет использован тип данных DATE, который хранит даты с 1 января 1000 года до 31 деабря 9999 года (c "1000-01-01" до "9999-12-31"). По умолчанию для хранения используется формат yyyy-mm-dd. Занимает 3 байта.

Путь к распознаваемому изображению имеет смысл хранить с типом данных VARCHAR с ограничением на длину пути 255. Данный тип может представлять русские символы, корректно отображать их как в самой базе данных, так и имеется возможность эти данные корректно считывать из базы данных.

Данные о уровне корректного распознавания будут храниться в типе данных DOUBLE, который хранит дробные числа с плавающей точкой двойной точности от -1.7976 \* 10308 до 1.7976 \* 10308, занимает 8 байт. Может принимать форму DOUBLE (M, D), где M - общее количество цифр, а D - количество цифр после запятой. Чем меньше значение уровня корректного распознавания, тем лучше было распознано изображение.

**Рассмотрим сущность “Накладные”:**

Уникальный идентификатор накладной имеет смысл хранить с типом данных VARCHAR, с ограничением на количество символом – 20. Данного типа данных с ограничением более чем достаточно, чтобы хранить большое множество разнообразных накладных, которых может быть больше, чем самих полувагонов.

Название поставщика будет храниться с типом данных NVARCHAR, с ограничением на количество символов – 100. Данного типа данных с ограничением более чем достаточно, для хранения названия поставщика.

Общее количество полувагонов в составе будет храниться с типом данных SMALLINT, который представляет целые числа от -32768 до 32767 и занимает 2 байта.

Все атрибуты с датами будут храниться с типом данных DATE, уже упоминаемый ранее.

**Рассмотрим сущность “Регистрация”:**

Уникальный идентификатор накладной и уникальный идентификатор полувагона будут храниться без изменения, однако следует учесть, что на накладную поле будет являться внешним ключом, по отношению к таблице “Регистрация”, но другое поле – идентификатор полувагона – не будет являться внешним по отношению к таблице “Регистрация” на таблицу “Полувагоны”, поскольку данные о идентификаторе полувагонов по накладной могут существовать и тогда, когда таких данных нет в таблице “Полувагоны”.

Отметка о прибытии будет храниться с типом данных BOOL, который фактически не представляет отдельный тип, а является лишь псевдонимом для типа TINYINT (1) и может хранить два значения 0 и 1. Однако данный тип может также в качестве значения принимать встроенные константы TRUE (представляет число 1) и FALSE (предоставляет число 0).

ID записи будет храниться в типе данных INT, который был упомянут выше.

Фактический и порядковый номера в составе будут храниться в типе данных SMALLINT.

Процент вещества в полувагоне будет храниться в типе данных FLOAT, который хранит дробные числа с плавающей точкой одинарной точности от -3.4028 \* 1038 до 3.4028 \* 1038 и занимает 4 байта.

Может принимать форму FLOAT (M, D), где M - общее количество цифр, а D - количество цифр после запятой.

Рассмотрев все сущности и форматы представления данных, которые необходимы, была спроектирована физическая модель базы данных (см. рис. 3).

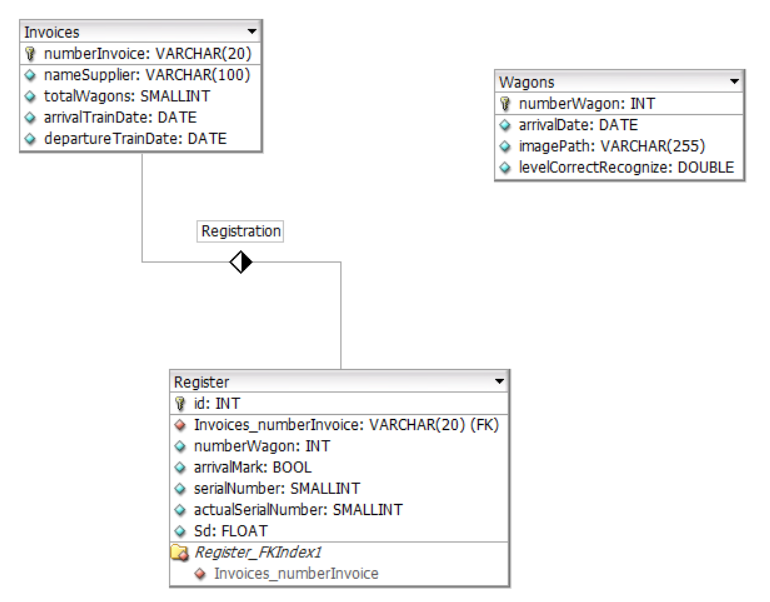


Рисунок 3 – Физическая модель базы данных

Проектирование модуля “Центральный сервер”

Для реализации серверной части будет использован фреймворк Spring MVC, позволяющий достаточно быстро организовать работу с серверной частью и обеспечить быстрый доступ к базе данных, расположенной на сервере. База данных, к которой будет подключена серверная часть и к которой будет обеспечивать доступ – MySQL.

Модуль “Центральный сервер” должен реализовывать паттерн DAO (Data Access Object), который определяет взаимодействие системы с базой данных MySQL.

Взаимодействие клиентской частью приложения с базой данных, расположенной на сервере, будет реализовываться с помощью протокола HTTP, используя POST и GET запросы, поэтому контроллеры будут работать исключительно с данными типами запросов.

POST запросы должны реализовывать логику изменения данных в базе данных. Например, для добавления, удаления или обновления определённой записи в таблице, содержащей зарегистрированные полувагоны по их прибытию.

GET запросы должны реализовывать логику загрузки данных с сервера, на клиентскую часть приложения. Например, загрузка изображения полувагона с локального хранилища сервера на клиентскую часть изображения.

Серверная часть разбита на определённое множество пакетов, которые будут содержать в себе классы, реализующие определённый функционал для работоспособности серверной части приложения согласно паттерну DAO. В соответствии с этим, для каждого пакета будет представлена UML-диаграмма классов и интерфейсов, содержащихся в этих пакетах.

Функционал пакетов серверной части приложения:

1) **com.server.configs** : классы, осуществляющие настройку конфигурации для взаимодействия с базой данных

2) **com.server.controllers** : классы, контролирующие обращения к серверу по средствам HTTP протокола. Основные запросы: GET и POST. Так же, может содержать контроллеры, которые обрабатывают ошибки, возникающие в других контроллерах.

3) **com.server.database.dao** : интерфейсы и классы, реализующие логику паттерна DAO (интерфейс и реализацию обращения к базе данных на низком уровне)

4) **com.server.database.elements** : классы, экземпляры объектов которых будут единицами данных для таблиц, в базе данных и локальном хранилище. С помощью данных классов возможен возврат данных посредством GET-запросов и общее взаимодействие с данными из базы данных.

5) **com.server.database.mappres** : классы, реализующие интерфейс RowMapper<T> для построчного чтения данных из определённых таблиц в базе данных, где T – тип единицы данных для соответствующей таблицы.

6) **com.server.database.requests** : классы, использующиеся для обработки POST-запросов, полученных от клиентской части приложения.

7) **com.server.database.services** : интерфейсы и классы, определяющие сервис для взаимодействия с базой данных.

8) **com.server.exceptions** : классы, определяющие виды исключений возникающие при различных обращениях к базе данных.

9) **com.server.settings** : классы, определяющие настройку подключения к базе данных.

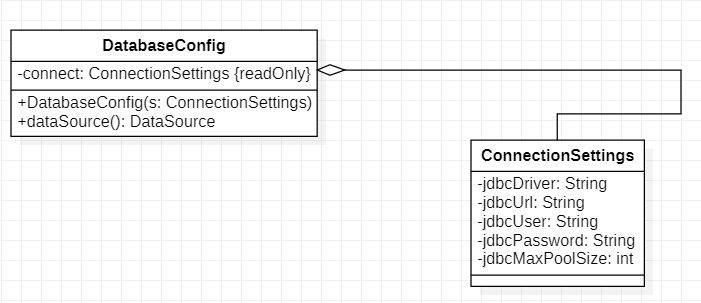


Рисунок 4 – Общая UML-диаграмма классов серверной части, занимающейся настройкой и конфигурированием базы данных

Рисунок 5 – Общая UML - диаграмма классов серверной части, реализующая паттерн DAO и маршрутизацию на основе HTTP-протокола

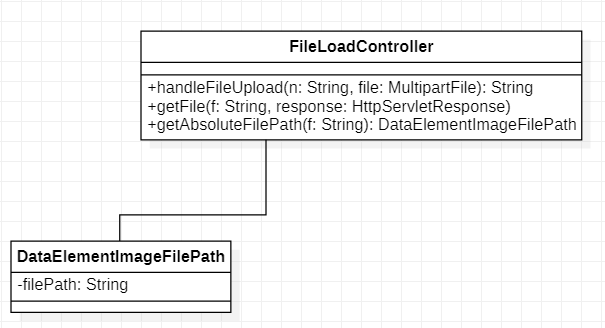


Рисунок 6 – Общая UML – диаграмма классов серверной части, реализующая логику загрузки файлов с сервера на клиентскую часть приложения и обратно

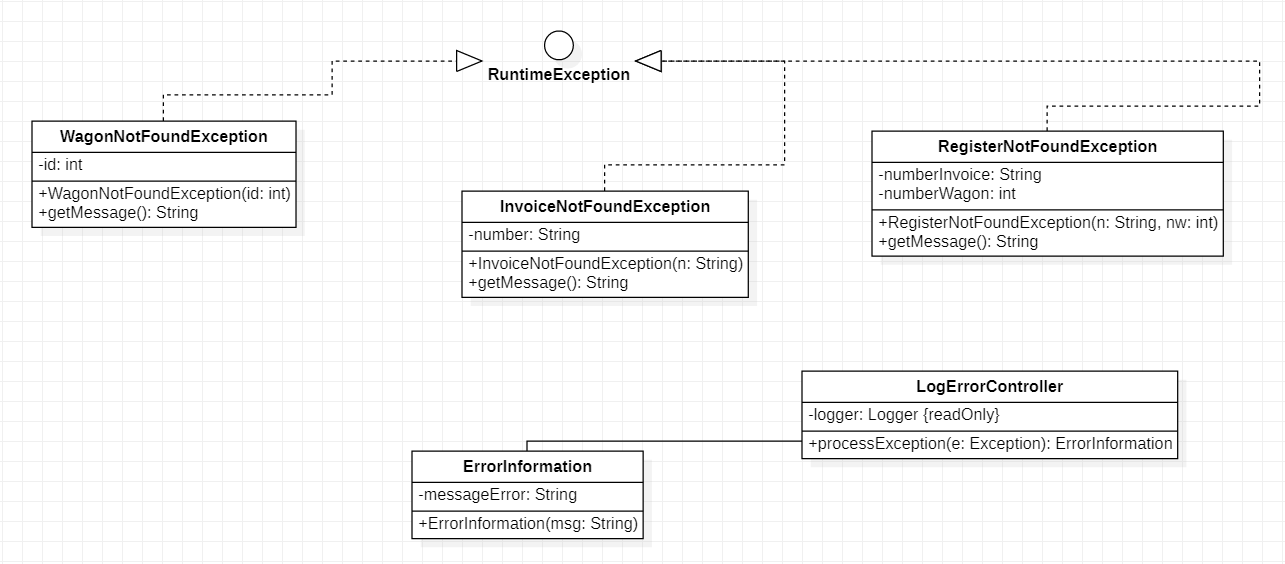


Рисунок 7 – Общая UML – диаграмма классов серверной части, реализующая обработку ошибок, возникающих внутри сервера

Реализация приложения

Реализация модуля “Центральный сервер”

Описание реализованного модуля

Серверная часть была написана с использованием фреймворка Spring. В серверной части была развернута база данных MySQL, с определёнными первоначальными настройками. Данный модуль единственный, кто имеет прямой доступ к базе данных. Клиентское приложение осуществляет доступ к базе данных только через интерфейс, который предоставляет сервер. Данный интерфейс состоит из POST/GET запросов, которые посылаются из клиентской части приложения серверной. Сервер обрабатывает входные данные в формате JSON (или MultipartFile) при POST запросах и возвращает данные в формате JSON при GET запросах со стороны клиента.

Взаимодействие с базой данных осуществляется с помощью реализованного абстрактного интерфейса Data Access Object. Настройка и конфигурирование базы данных осуществляется с помощью высокопроизводительной библиотеки пула соединений HikariCP. Данная библиотека в основном используется в классе DataBaseConfig.

База данных развёртывается на сервере автоматически. Для этого были использованы представления SQL команд в виде строковых констант Java. Используя SQL команды серверная часть создаёт таблицы реляционной базы данных даже если базы данных до этого не существовало. То есть, достаточно запустить серверную часть на машине, чтобы взаимодействовать с базой данных удалённо посредством интерфейса предоставленного серверной частью в виде POST/GET запросов.

Для считывания данных из базы данных используется интерфейс RowMapper<T>, который позволяет считать данные определённой структуры из таблицы.

Множество классов сервера были разделены на отдельные пакеты, в которых содержатся классы, относящиеся к той или иной части серверного приложения. Например, классы DAO и классы, экземпляры объектов которых используются для десериализации данных из формата JSON, содержатся в различных пакетах. Полное деление классов серверной части представлено на рисунке **X**

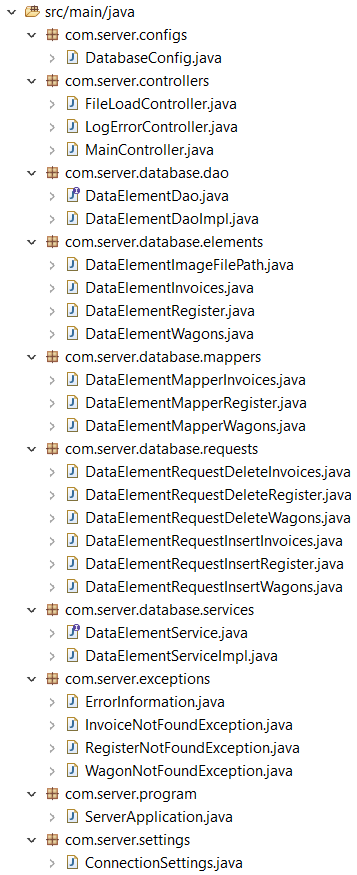


Рисунок X – Пакеты серверной части приложения

Описание реализованных классов

Класс **FileLoadController** является контроллером, который обеспечивает обработку POST/GET запросов к серверу для загрузки файла в локальное хранилище сервера, так и его загрузки с сервера в локальное хранилище клиентского приложения (для обеспечения визуального контакта с распознанным изображением). Для передачи данных по протоколу HTTP на сервер и с сервера был выбран формат JSON и подключены соответствующие библиотеки для сериализации и десериализации данных, с помощью которых обеспечиваются взаимосвязи между серверной частью и клиентским приложением. Однако, POST-запросы для контроллера FileLoadController обладают уникальным поведением – они не поддерживают JSON формат, данные передаются в виде формата multipart/form-data, который позволяет передавать файлы с сервера на клиентское приложение и обратно. Один GET-запрос всё же поддерживает JSON формат – это метод getAbsoluteFilePath, который возвращает абсолютный путь к файлу, расположенному в локальном хранилище сервера.

Поддерживаемые POST-запросы контроллером FileLoadController:

1) http://localhost:8080/upload (метод обработки: handleFileUpload)

Поддерживаемые GET-запросов контроллером FileLoadController:

1) http://localhost:8080/load/?fileName=”название файла” (метод обработки: getFile)

2) http://localhost:8080/load/filepath (метод обработки: getAbsoluteFilePath)

Для сериализации был создан класс **DataElementImageFilePath** (необходим для обработки одного GET-запроса, с адресом http://localhost:8080/load/filepath). Для данного класса таблица спецификации не требуется. Данный класс содержит одно единственное поле с спецификатором доступа private и геттеры/сеттеры для изменения значений, а также конструктор с одним параметром.

Таблица спецификации методов класса FileLoadController, способствующих обработке POST/GET запросов для загрузки файла на сервер, а также загрузки файла с сервера, представлена в таблице 1.

Экземпляры объектов классов **DataElementInvoices**, **DataElementRegister** и **DataElementWagons** являются единицами данных в технологии программирования ORM, которая используется совместно с моделью проектирования DAO (Data Access Object).

Для данных классов таблица спецификации не предусмотрена, так как функциональность данных классов не несёт в себе никакой смысловой нагрузки. В данных классах определены поля определённого типа, со спецификаторами доступа private, а также конструктор, геттеры и сеттеры для соответствующих полей.

В классе DataElementInvoices содержатся данные, которые были выделены для таблицы Invoices в базе данных (см. рис. 3). В классе DataElementRegister содержатся данные, которые были выделены для таблицы Register (см. рис. 3). В классе DataElementWagons содержатся данные, которые были выделены для таблицы Wagons в базе данных (см. рис. 3).

Классы **WagonNotFoundException**, **RegisterNotFoundException**, **InvoiceNotFoundException**, **ErrorInformation** и **LogErrorController** являются серверной частью, реализующие обработку ошибок, возникающие внутри сервера, а также трансляцию данных ошибок клиентской части приложения для уведомления пользователя о возникновении данных ошибок.

Классы с постфиксом NotFoundException возникают тогда, когда не найдена определённая запись в базе данных по соответствующим уникальным идентификаторам соответствующих таблиц. Например, для таблицы Wagons уникальным идентификатором будет numberWagon, и в случае, когда не было найдено записи с определённым numberWagon, данное исключение (WagonNotFoundException) возникает, а LogErrorController логгирует данную ошибку и отправляет сообщение об ошибке обратно клиентской части приложения, в случае, если клиентское приложение обратилось с GET или POST запросом к контроллеру, обеспечивающий взаимодействие с базой данных (MainController).

Таблиц спецификаций для данных классов также не предусмотрено, поскольку данные классы содержат лишь информацию об ошибке и геттеры с сеттерами. Информация об ошибках у всех классов разная. А контроллер LogErrorController имеет один единственный метод processException, которые обрабатывает исключение и посылает клиентской части приложения информацию о внутренней ошибке сервера.

Классы **DatabaseConfig** и **ConnectionSettings** реализуют логику настройки и конфигурирования базы данных. Для упрощения конфигурирования, был использован встроенный в фреймворк Spring класс – HikariConfig. Данный класс используется для инициализации источника данных, и его использование осуществляется в методе dataSource() класса DatabaseConfig. Для данных классов таблица спецификации методов не предусмотрена, поскольку данные классы содержат лишь определённые поля с геттерами и сеттерами, исключением является лишь класс DatabaseConfig, который содержит один метод dataSource(), который также не несёт в себе большой смысловой нагрузки – он возвращает настройки подключения к базе данных (источник данных).

Экземпляры объектов классов **DataElementMapperInvoices**, **DataElementMapperRegister** и **DataElementMapperWagons** используются для чтения данных из базы данных с использованием технологии ORM (осуществляется построчное чтение из каждой таблицы в базе данных). Данные классы реализуют интерфейс RowMapper<T> и переопределяют один единственный метод mapRow для чтения данных.

Таблица спецификаций для классов, имплементирующие интерфейс RowMapper представлена в таблице 2.

Экземпляры объектов классов **DataElementRequestDeleteInvoices**, **DataElementRequestDeleteRegister**, **DataElementRequestDeleteWagons**, **DataElementRequestInsertInvoices**, **DataElementRequestInsertRegister** и **DataElementRequestInsertWagons** используются для обмена данными между клиентской и серверной частями приложения посредством использования формата JSON, по протоколу HTTP.

Для данных классов также не предусмотрена таблица спецификаций. Классы с постфиксом Delete(Register/Invoices/Wagons) представляют данные, которые необходимо отправить с клиентской части приложение на сервер для удаления определённой записи, а классы с постфиксом Insert(Register/Invoice/Wagons) наоборот, для записи или обновления записей в базе данных.

Классы **DataElementDaoImpl** и **DataElementServiceImpl** в совокупности реализуют интерфейс DAO.

Таблица спецификаций для данных классов представлена в таблице 3.

Класс **MainController** является контроллером, который обеспечивает взаимодействие между серверной и клиентской частью приложения посредством обмена данными через HTTP протокол, используя POST/GET запросы. Данный класс является главным контроллером серверной части приложения и предоставляет интерфейс для взаимодействия с базой данных. Через POST/GET запросы передаются данные в формате JSON. Абсолютно все адреса обрабатывают данные JSON формата, исключений нет. Например, для добавления полувагона необходимо передать по адресу http://localhost:8080/database/wagons/insert данные в формате JSON, таким образом, чтобы можно было их десериализовать в экземпляры объекта класса DataElementRequestInsertWagons.

Поддерживаемые POST-запросы для контроллера MainController:

1) http://localhost:8080/database/settings/sizenumberwagon - задание длины номера полувагона

2) http://localhost:8080/database/settings/minsizenumberinvoice - задание длины номера накладной

3) http://localhost:8080/database/wagons/insert - добавление информации о полувагоне (после регистрации камерой)

4) http://localhost:8080/database/wagons/update - обновление определённой записи о полувагоне в базе данных

5) http://localhost:8080/database/wagons/delete - удаление информации об определённом полувагоне

6) http://localhost:8080/database/invoices/insert - добавление информации об определённой накладной

7) http://localhost:8080/database/invoices/update - обновление информации об определённой накладной

8) http://localhost:8080/database/invoices/delete - удаление информации об определённой накладной

9) http://localhost:8080/database/register/insert - добавление информации о соответствии полувагона накладной

10) http://localhost:8080/database/register/update - обновление информации о соответствии полувагона накладной

11) http://localhost:8080/database/register/delete - удаление информации о соответствии полувагона накладной

Поддерживаемые GET-запросы для контроллера MainController:

1) http://localhost:8080/database/wagons/get?numberWagon=(номер полувагона) – информация о конкретном полувагоне с определённым номером.

2) http://localhost:8080/database/wagons/get/all - информация обо всех полувагонах, которые были зарегистрированы камерой.

3) http://localhost:8080/database/invoices/get/?numberInvoice=”номер накладной” – информация об определённой накладной с уникальным идентификатором.

4) http://localhost:8080/database/invoices/get/all - информация обо всех накладных, которые поступили на предприятие

5) http://localhost:8080/database/register/get/?numberInvoice=”номер накладной”&numberWagon=(номер полувагона) – информация об определённом зарегистрированном по данной накладной полувагоне (о полувагоне, соответствующем определённой накладной)

6) http://localhost:8080/database/register/get/all - получение информации обо всех записях, являющиеся соответствием полувагона определённой накладной

7) http://localhost:8080/database/register/get/all/numbers - получение информации обо всех номерах полувагона, которые зарегистрированы в таблице соответствия номера полувагона определённой накладной

8) http://localhost:8080/database/register/numberwagon/is - возвращает информацию о существовании (не существовании) в таблице соответствия определённого номера полувагона.

Таблица спецификации для данного класса представлена в таблице 4.

Класс **ServerApplication** содержит в себе главный метод запуска серверной части приложения main и в целом является классом, который настраивает общую конфигурацию сервера и руководит поиском и настройкой бинов. С данного класса начинается выполнения всей программы.

Таблица спецификации для данного класса представлена в таблице 5.

Спецификация методов реализованных классов

Таблица 1 – Таблица спецификации методов для класса FileLoadController

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Назначение** | **Тип данных** | **Выходные параметры** | **Назначение** | **Тип данных** |
| handleFileUpload | name  file | Имя файла  Данные файла | String  MultipartFile | path | Абсолютный путь к файлу в локальном хранилище сервера | String |
| getFile | fileName  response | Имя файла  Данные файла | String  HttpResponseServlet | - | - | - |
| getAbsoluteFilePath | fileName | Имя файла | String | data | Абсолютный путь к файлу в локальном хранилище сервера | DataElementImageFilePath |

Таблица 2 – Таблица спецификации методов для классов, реализующих интерфейс RowMapper<T> для чтения данных из таблиц в базе данных MySQL согласно технологии ORM

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Назначение** | **Тип данных** | **Выходные параметры** | **Назначение** | **Тип данных** |
| **Класс DataElementMapperInvoices** | | | | | | |
| mapRow | rs  rowNum | Результат считывания одной строки  Номер строки | ResultSet  int | data | Считанные данные в виде экземпляра объекта класса | DataElementInvoice |
| **Класс DataElementMapperRegister** | | | | | | |
| mapRow | rs  rowNum | Результат считывания одной строки  Номер строки | ResultSet  int | data | Считанные данные в виде экземпляра объекта класса | DataElementRegister |
| **Класс DataElementMapperWagons** | | | | | | |
| mapRow | rs  rowNum | Результат считывания одной строки  Номер строки | ResultSet  int | data | Считанные данные в виде экземпляра объекта класса | DataElementWagons |

Таблица 3 – Таблица спецификации методов для классов, реализующих интерфейс Data Access Object

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Назначение** | **Тип данных** | **Выходные параметры** | **Назначение** | **Тип данных** |
| **Класс DataElementDaoImpl** | | | | | | |
| DataElementDaoImpl | MapW  MapI  MapR  jTempl | Считыватель полувагонов  Считыватель накладных  Считыватель из таблицы регистрации  Параметр шаблона | DataElementMapperWagons  DataElementMapperInvoices  DataElementMapperRegister  NamedParameterJdbcTemplate | - | - | - |
| getDataElementWagonsByNumber | number | Номер полувагона | int | data | Данные о полувагоне | Optional<DataElementWagons> |
| getDataElementWagonsAll | - | - | - | data | Данные о полувагонах | List<Optional<DataElementWagons>> |

Продолжение таблицы 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| insertDataElementWagons | numberWagon  arrivalDate  imagePath  levelCorrect | Номер полувагона  Дата прибытия  Путь к изображению  Уровень корректного распознавания | int  String  String  double | - | - | - |
| updateDataElementWagons | numberWagon  arrivalDate  imagePath  levelCorrect | Номер полувагона  Дата прибытия  Путь к изображению  Уровень корректного распознавания | int  String  String  double | - | - | - |
| deleteDataElementWagons | numberWagon | Номер полувагона | int | - | - | - |
| getDataElementInvoicesById | numberInvoices | Номер накладной | String | data | Информация о накладной | Optional<DataElementInvoices> |
| getDataElementInvoicesAll | - | - | - | data | Информация о накладных | List<Optional<DataElementInvoices>> |

Продолжение таблицы 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| insertDataElementInvoices | numberInvoice  nameSupplier  totalWagons  arrivalTrainDate  departureTrainDate | Номер накладной  Название поставщика  Общее число полувагонов  Дата прибытия полувагонов  Дата отправки полувагонов | String  String  Short  String  String | - | - | - |
| updateDataElementInvoices | numberInvoice  nameSupplier  totalWagons  arrivalTrainDate  departureTrainDate | Номер накладной  Название поставщика  Общее число полувагонов  Дата прибытия полувагонов  Дата отправки полувагонов | String  String  Short  String  String | - | - | - |
| deleteDataElementInvoices | numberInvoice | Номер накладной | String | - | - | - |

Продолжение таблицы 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| getDataElementRegisterById | numberInvoice  numberWagon | Номер накладной  Номер полувагона | String  int | data | Данные о соответствии полувагона накладной | Optional<DataElementRegister> |
| getDataElementRegisterAll | - | - | - | data | Данные о всех соответствиях полувагонов накладным | List<Optional<DataElementRegister>> |
| insertDataElementRegister | numberInvoice  numberWagon  serialNumber  sD | Номер накладной  Номер полувагона  Порядковый номер в составе на момент отправки  Процент сыпучести вещества | String  int  Short  float | - | - | - |

Продолжение таблицы 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| insertDataElementRegister | numberInvoice  numberWagon  serialNumber  sD | Номер накладной  Номер полувагона  Порядковый номер в составе на момент отправки  Процент сыпучести вещества | String  int  Short  float | - | - | - |
| deleteDataElementRegister | numberInvoice  numberWagon | Номер накладной  Номер полувагона | String  int | - | - | - |
| updateDataElementRegisterActualNumber | numberInvoice  numberWagon  actualSerialNumber | Номер накладной  Номер полувагона  Фактический порядковый номер полувагона в составе, вычисленный при регистрации камерой | String  int  short | - | - | - |
| **Класс DataElementServiceImpl** | | | | | | |
| DataElementServiceImpl | dDao | Экземпляр объекта реализующего интерфейс DAO | DataElementDao | - | - | - |

Продолжение таблицы 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| getDataElementWagonsAll | - | - | - | data | Информация обо всех полувагонах зарегистрированных камерой | List<DataElementWagons> |
| getDataElementInvoicesAll | - | - | - | data | Информация обо всех накладных | List<DataElementInvoices> |
| getDataElementRegisterAll | - | - | - | data | Информация обо всех соответствиях полувагонов накладным | List<DataElementRegister> |
| getDataElementWagons | numberWagon | Номер полувагона | int | data | Информация о полувагоне | DataElementWagons |
| getDataElementInvoices | numberInvoice | Номер накладной | String | data | Информация о накладной | DataElementInvoices |
| getDataElementRegister | numberInvoice  numberWagon | Номер накладной  Номер полувагона | String  int | data | Информации о соответствии полувагона накладной | DataElementRegister |
| insertDataElementWagons | numberWagon  arrivalDate  imagePath  levelCorrect | Номер полувагона  Дата прибытия  Путь к изображению  Уровень корректного распознавания | int  String  String  double | - | - | - |

Продолжение таблицы 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| insertDataElementInvoices | numberInvoice  nameSupplier  totalWagons  arrivalTrainDate  departureTrainDate | Номер накладной  Название поставщика  Общее число полувагонов  Дата прибытия состава  Дата отправки состава | String  String  Short  String  String | - | - | - |
| insertDataElementRegister | numberInvoice  numberWagon  serialNumber  sD | Номер накладной  Номер полувагона  Порядковый номер в составе  Процент сыпучести вещества | String  int  short  float | - | - | - |
| updateDataElementWagons | numberWagon  arrivalDate  imagePath  levelCorrect | Номер полувагона  Дата прибытия  Путь к изображению  Уровень корректного распознавания | int  String  String  double | - | - | - |
| updateDataElementInvoices | numberInvoice  nameSupplier  totalWagons  arrivalTrainDate  departureTrainDate | Номер накладной  Название поставщика  Общее число полувагонов  Дата прибытия состава  Дата отправки состава | String  String  Short  String  String | - | - | - |

Продолжение таблицы 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| updateDataElementRegister | numberInvoice  numberWagon  serialNumber  sD | Номер накладной  Номер полувагона  Порядковый номер в составе  Процент сыпучести вещества | String  int  short  float | - | - | - |
| deleteDataElementWagons | numberWagon | Номер полувагона | int | - | - | - |
| deleteDataElementInvoices | numberInvoice | Номер накладной | String | - | - | - |
| deleteDataElementRegister | numberInvoice  numberWagon | Номер накладной  Номер полувагона | String  int | - | - | - |
| updateDataElementRegisterActualNumber | numberInvoice  numberWagon  actualSerialNumber | Номер накладной  Номер полувагона  Фактический порядковый номер полувагона в составе | String  Int  short | - | - | - |

Таблица 4 – Таблица спецификации методов для класса MainController

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Назначение** | **Тип данных** | **Выходные параметры** | **Назначение** | **Тип данных** |
| MainController | element | Экземпляр объекта сервиса, обеспечивающее возможность взаимодействовать с базой данных посредством использования интерфейса DAO | DataElemenService | - | - | - |
| updateMaxSizeNumberWagon | request | Десериализованные данные, характеризующие длину номера полувагона | SizeNumberWagon | - | - | - |
| updateMaxSizeNumberInvoice | request | Десериализованные данные, характеризующие длину номера накладной | SizeMinNumberInvoice | - | - | - |
| getDataElementWagons | numberWagon | Номер полувагона | int | data | Данные полувагона | DataElementWagons |

Продолжение таблицы 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| getDataElementWagonsAll | - | - | - | data | Информация обо всех полувагонах | List<DataElementWagons> |
| insertDataElementWagons | request | Десериализованные данные, характеризующие информацию о полувагоне | DataElementRequestInsertWagons | - | - | - |
| updateDataElementWagons | request | Десериализованные данные, характеризующие информацию о полувагоне | DataElementRequestInsert | - | - | - |
| deleteDataElementWagons | request | Десериализованные данные, характеризующие информацию о полувагоне, который нужно удалить | DataElementRequestDelete | - | - | - |
| getDataElementInvoices | numberInvoice | Номер накладной | String | data | Данные накладной | DataElementInvoices |

Продолжение таблицы 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| getDataElementInvoicesAlll | - | - | - | data | Информация обо всех накладных | List<DataElementInvoices> |
| insertDataElementInvoices | request | Десериализованные данные, характеризующие информацию о накладной | DataElementRequestInsertInvoices | - | - | - |
| updateDataElementInvoices | request | Десериализованные данные, характеризующие информацию о накладной | DataElementRequestInsertInvoices | - | - | - |
| deleteDataElementInvoices | request | Десериализованные данные, характеризующие информацию о накладной которую нужно удалить | DataElementRequestDeleteInvoices | - | - | - |

Продолжение таблицы 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| getDataElementRegister | numberInvoice  numberWagon | Номер накладной  Номер полувагона | String  int | data | Информация о соответствии полувагона накладной | DataElementRegister |
| getDataElementRegisterAll | - | - | - | data | Информация обо всех соответствиях полувагонов накладным | List<DataElementRegister> |
| getDataNumberWagonRegisterAll | - | - | - | data | Информация обо всех номерах, которые есть в таблице соответствия полувагона накладной | List<DataElementNumberWagon> |

Продолжение таблицы 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| insertDataElementRegister | request | Десериализованные данные, характеризующие соответствие полувагона определённой накладной | DataElementRequestInsertRegister | - | - | - |
| updateDataElementRegister | request | Десериализованные данные, характеризующие соответствие полувагона определённой накладной | DataElementRequestInsertRegister | - | - | - |
| isNumberWagon | numberWagon | Номер полувагона | Integer | data | Результат поиска номера полувагона в таблице соответствия | DataAnswer |
| deleteDataElementRegister | request | Десериализованные данные характеризующие запись в таблице соответствия для удаления | DataElementRequestDeleteRegister | - | - | - |

Таблица 5 – Таблица спецификации методов для класса ServerApplication

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Назначение** | **Тип данных** | **Выходные параметры** | **Назначение** | **Тип данных** |
| main | args | Аргументы для изменения поведения программы | String[] | - | - | - |
| multipartConfigElement | - | - | - | factory | Конфигурация для передачи файла с сервера и на сервер (в локальное хранилище) | MultipartConfigElement |

Тестирование приложения

Тестирование модуля “Центральный сервер”

Выбор методики тестирования

Тестирование серверной части приложения на корректность взаимодействия с базой данных для наиболее достоверной наглядности будет осуществлено ручным способом, с использованием онлайн сервиса для тестирования – REQBIN [1]. Данный сервис позволяет отправлять POST/GET запросы серверной части приложения и показывает результат возвращаемого значения (ответ сервера на запрос).

Тестированию будут подвергнуты контроллеры MainController, FileLoadController и LogErrorController, который в случае возникновения внутренней ошибки на стороне сервера, будет возвращать сообщение с ошибкой в формате JSON (которое будет отображаться в клиентской части приложения при возникновении ошибки).

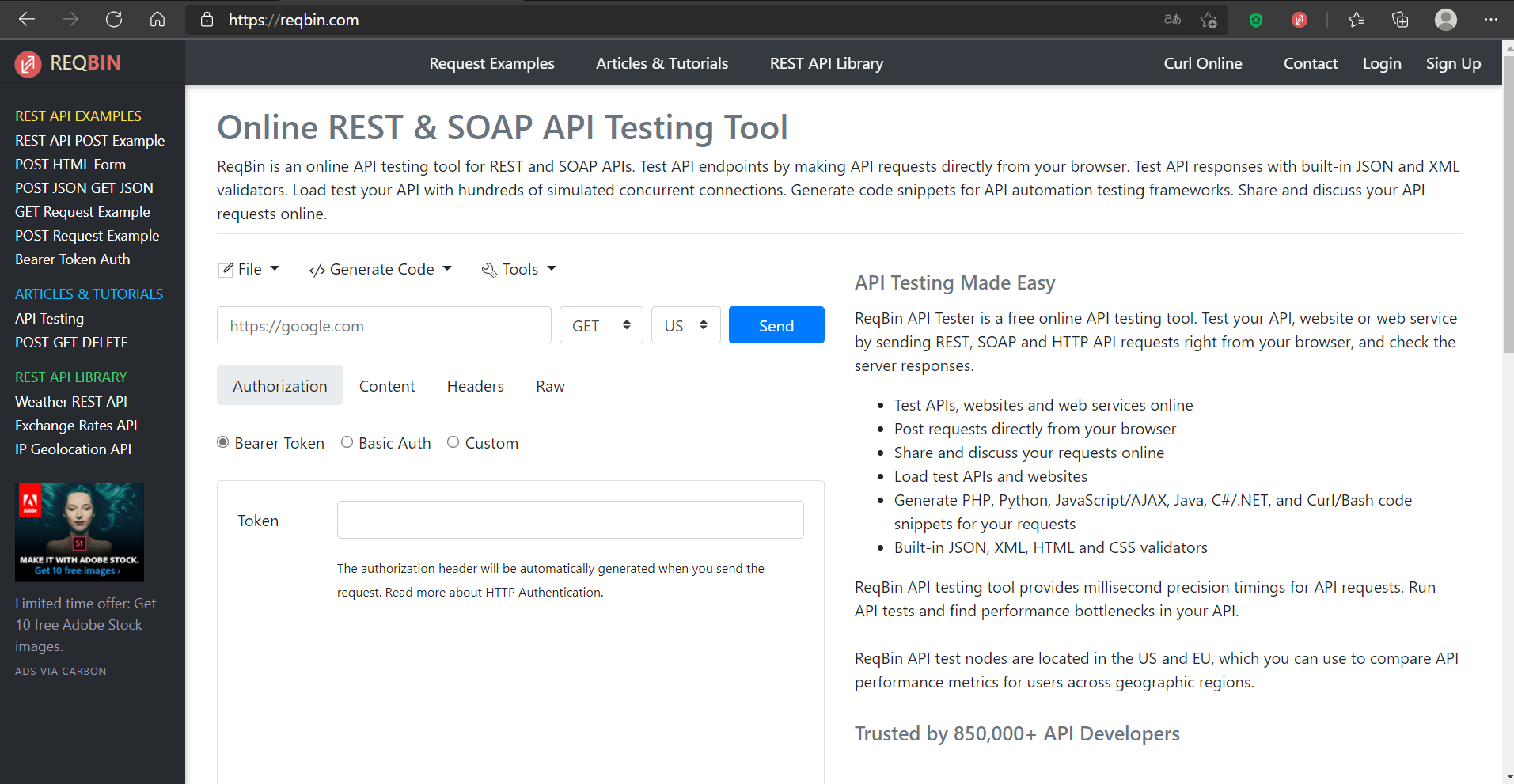


Рисунок X – Интерфейс онлайн сервиса для тестирования обработки запросов

Таблица тестов

В таблице X представлена таблица тестов. Входные и выходные данные представлены в формате JSON. Будем считать, что в случае удачной обработки POST-запроса онлайн сервис REQBIN не будет ничего возвращать в строку результата (она используется только для GET-запросов), однако, при удачной обработке будет возвращён статус 202 (ACCEPTED), который можно увидеть после отправки POST-запроса. В случае ошибки обработки POST-запроса в результате онлайн сервис покажет строку с ошибкой в формате JSON

Таблица X – Таблица тестов для модуля “Центральный сервер”

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер теста** | **Назначение** | **Входные данные** | **Выходные данные** |
| **Таблица Invoices** | | | |
| 1 | Добавление накладной в таблицу Invoices  POST-запрос:  http://localhost:8080/database/invoices/insert | {  "numberInvoice":"140i,  "nameSupplier":"ООО Вагон",  "totalWagons":10,  "arrivalTrainDate":"2018-10-15",  "departureTrainDate":"2017-06-27"  } | (см. рис. X) |
| 2 | Изменение данных накладной в таблице Invoices  POST-запрос:  http://localhost:8080/database/invoices/update | {  "numberInvoice":"140i",  "nameSupplier":"ООО Полувагон",  "totalWagons":20,  "arrivalTrainDate":"2020-10-07",  "departureTrainDate":"2017-06-27"  } | (см. рис. X) |

Продолжение таблицы X

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | Информация о накладной с определённым номером  GET-запрос:  http://localhost:8080/database/invoices/get/?numberInvoice=140i | numberInvoice=”140i” | {  "numberInvoice": "140i",  "nameSupplier": "ООО Полувагон",  "totalWagons": 20,  "arrivalTrainDate": "2020-10-07",  "departureTrainDate": "2017-06-27"  } (см. рис. X) |
| 4 | Информация о накладной с определённым номером  GET-запрос:  http://localhost:8080/database/invoices/get/?numberInvoice=ааа | numberInvoice=”ааа” | {  "message": "Записи с номером накладной ааа не найдено!"  } (см. рис. X) |
| 5 | Изменение данных накладной в таблице Invoices  POST-запрос:  http://localhost:8080/database/invoices/update | {  "numberInvoice":"140i",  "nameSupplier":"ООО Полувагон",  "totalWagons":20,  "arrivalTrainDate":"2000-10-07",  "departureTrainDate":"2017-06-27"  } | {  "message": "Дата приезда состава не может быть раньше даты отправки состава!"  } (см. рис. X) |

Продолжение таблицы X

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | Изменение данных накладной в таблице Invoices  POST-запрос:  http://localhost:8080/database/invoices/update | {  "numberInvoice":"140i",  "nameSupplier":"ООО Полувагон",  "totalWagons":-27,  "arrivalTrainDate":"2018-10-07",  "departureTrainDate":"2017-06-27"  } | {  "message": "Общее число прибывающих полувагонов по накладной не может быть меньше либо равно нулю!"  } (см. рис. X) |
| 7 | Удаление данных об определённой накладной в таблице Invoices  POST-запрос:  http://localhost:8080/database/invoices/delete | {  "numberInvoice":"140i"  } | (см. рис. X-X) |
| **Таблица Register** | | | |
| 8 | Добавление данных о соответствии полувагона определённой накладной  POST-запрос:  http://localhost:8080/database/register/insert | {  "fkNumberInvoice":"140i",  "numberWagon":12345678,  "serialNumber":1,  "sD":1.5  } | (см. рис. X) |
| 9 | Обновление данных о соответствии полувагона определённой накладной  POST-запрос:  http://localhost:8080/database/register/update | {  "fkNumberInvoice":"140i",  "numberWagon":12345678,  "serialNumber":1,  "sD":2.8  } | (см. рис. X) |

Продолжение таблицы X

|  |  |  |  |
| --- | --- | --- | --- |
| 10 | Информация об определённом соответствии полувагона определённой накладной  GET-запрос:  http://localhost:8080/database/register/get/?fkNumberInvoice=140i&numberWagon=12345678 | fkNumberInvoice=”140i”  numberWagon=12345678 | {  "fkNumberInvoice": "140i",  "numberWagon": 12345678,  "arrivalMark": false,  "serialNumber": 1,  "actualSerialNumber": 0,  "sD": 2.8  } (см. рис. X) |
| 11 | Обновление данных о соответствии полувагона определённой накладной  POST-запрос:  http://localhost:8080/database/register/update | {  "fkNumberInvoice":"140i",  "numberWagon":12345678,  "serialNumber":4,  "sD":2.8  } | {  "message": "Порядковый номер полувагона не изменяется! Чтобы изменить порядковый номер полувагона необходимо удалить запись с данным порядковым номером и добавить на её основе новую, с новым порядковым номером!"  }(см. рис. X) |

Продолжение таблицы X

|  |  |  |  |
| --- | --- | --- | --- |
| 12 | Обновление данных о соответствии полувагона определённой накладной  POST-запрос:  http://localhost:8080/database/register/update | {  "fkNumberInvoice":"140i",  "numberWagon":123456,  "serialNumber":1,  "sD":2.8  } | {  "message": "Номер полувагона должен состоять из 8 цифр!"  } (см. рис. X) |
| 13 | Информация обо всех соответствиях полувагонов определённым накладным  POST-запрос:  http://localhost:8080/database/register/get/all | - | [{  "fkNumberInvoice": "140i",  "numberWagon": 12345678,  "arrivalMark": false,  "serialNumber": 1,  "actualSerialNumber": 0,  "sD": 2.8  }, {  "fkNumberInvoice": "140i",  "numberWagon": 87654321,  "arrivalMark": false,  "serialNumber": 2,  "actualSerialNumber": 0,  "sD": 4.7  }] (см. рис. X) |
| 14 | Информация обо всех номерах полувагонов, которые занесены в таблицу соответствия  POST-запрос:  http://localhost:8080/database/register/get/all/numbers | - | [{  "numberWagon": 12345678  }, {  "numberWagon": 87654321  }] (см. рис. X) |

Продолжение таблицы X

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица Wagons** | | | |
| 15 | Добавление информации о полувагоне в таблицу Wagons  POST-запрос:  http://localhost:8080/database/wagons/insert | {  "numberWagon":12345678,  "arrivalDate":"2020-01-01",  "imagePath":"  "levelCorrectRecognize": 0.98  } | (см. рис. X) |
| 16 | Изменение информации о полувагоне в таблице Wagons:  POST-запрос:  http://localhost:8080/database/wagons/update | {  "numberWagon":12345678,  "arrivalDate":"2018-01-01",  "imagePath":"  "levelCorrectRecognize": 1.40  } | (см. рис. X) |
| 17 | Добавление информации о полувагоне в таблицу Wagons  POST-запрос:  http://localhost:8080/database/wagons/insert | {  "numberWagon":12345679,  "arrivalDate":"2014-01-01",  "imagePath":"  "levelCorrectRecognize": 0.45  } | {  "message": "Полувагон с данным идентификационным номером не зарегистрирован ни в одной накладной!"  } (см. рис. X) |

Продолжение таблицы X

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица Wagons** | | | |
| 18 | Добавление информации о полувагоне в таблицу Wagons  POST-запрос:  http://localhost:8080/database/wagons/insert | {  "numberWagon":87654321,  "arrivalDate":"1984-01-27",  "imagePath":"  "levelCorrectRecognize": 0.45  } | {  "message": "Прибытие полувагона не может быть раньше, чем отправка всего состава по данной накладной!"  } (см. рис. X) |
| 19 | Информация обо всех соответствиях полувагонов определённым накладным  POST-запрос:  http://localhost:8080/database/register/get/all | - | (см. рис. X) |
| 20 | Удаление записи о полувагоне из таблицы Wagons  POST-запрос: | {  "numberWagon":12345678  } | (см. рис. X-X) |

Результаты тестирования



Рисунок X – Результат выполнения теста №1 из таблицы тестирования №X

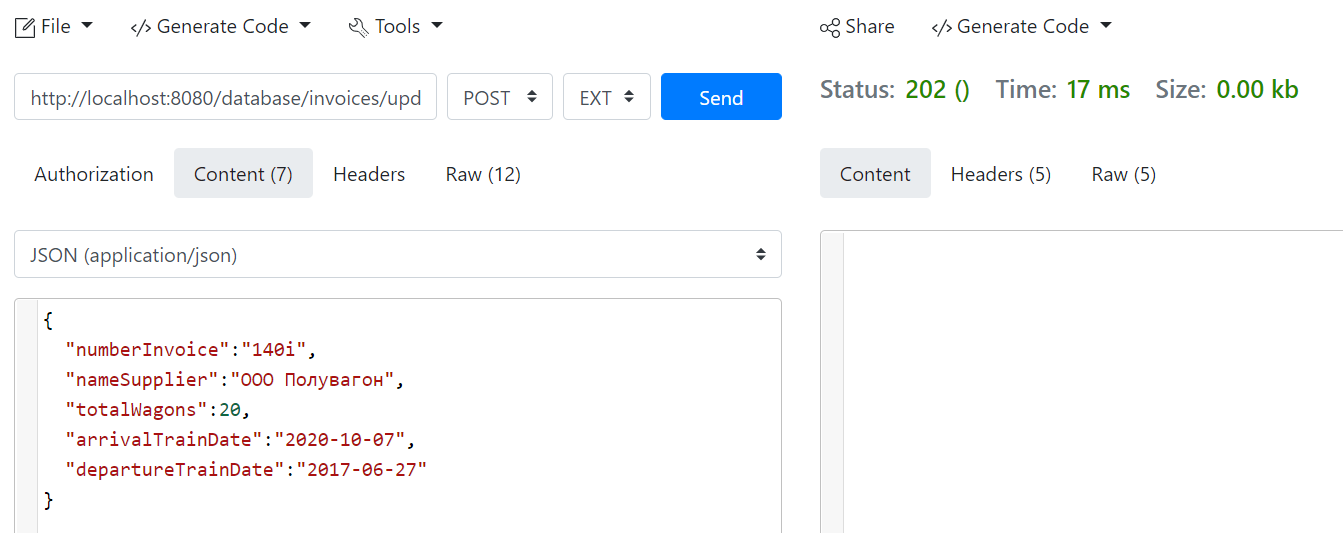


Рисунок X – Результат выполнения теста №2 из таблицы тестирования №X

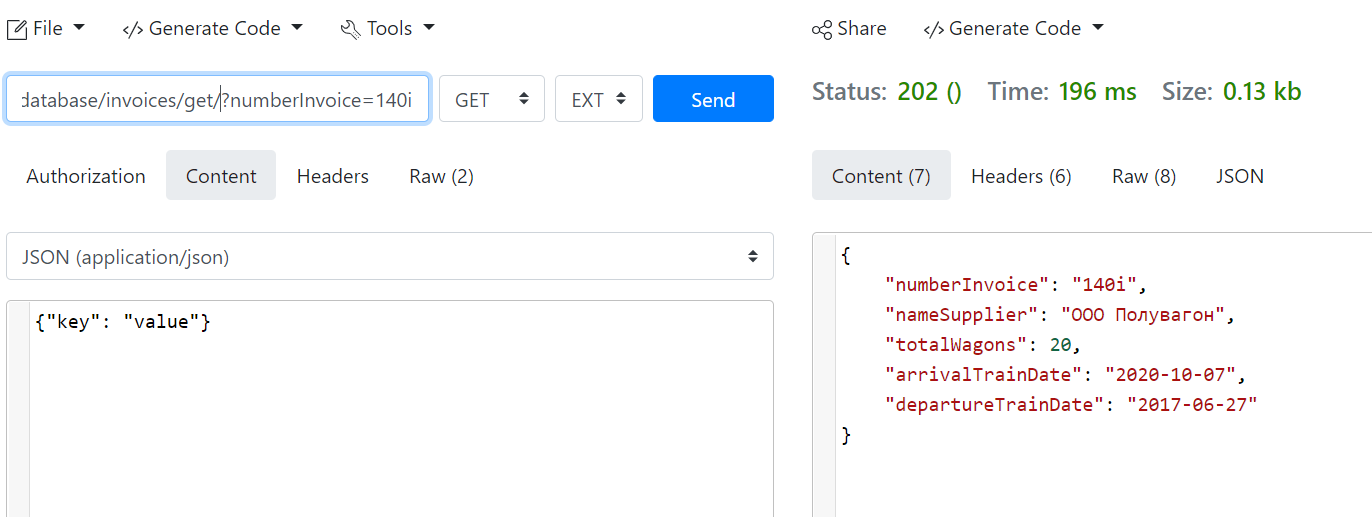


Рисунок X – Результат выполнения теста №3 из таблицы тестирования №X

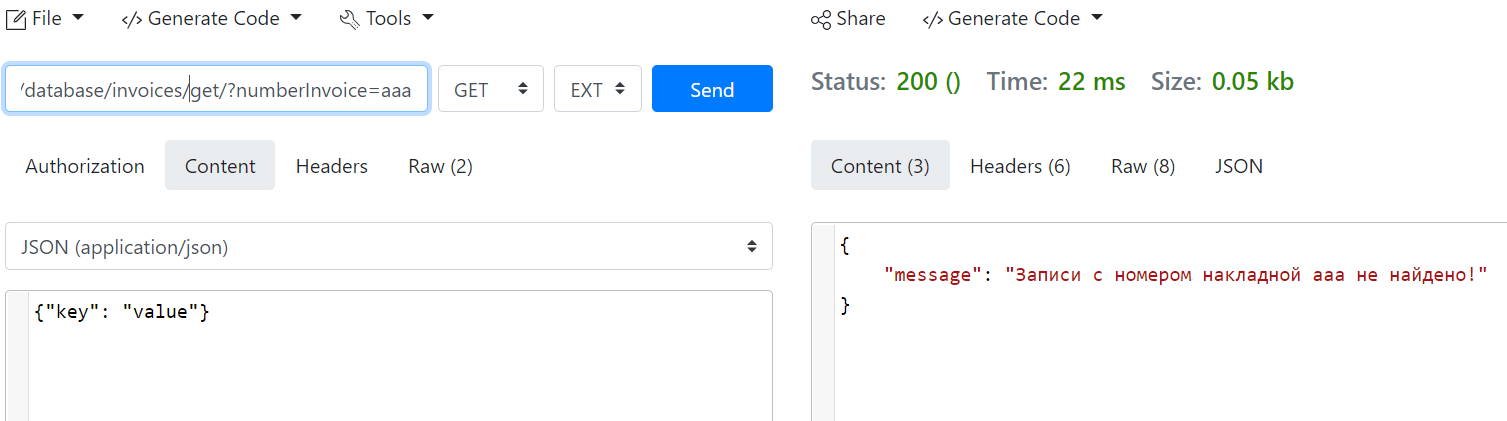


Рисунок X – Результат выполнения теста №4 из таблицы тестирования №X

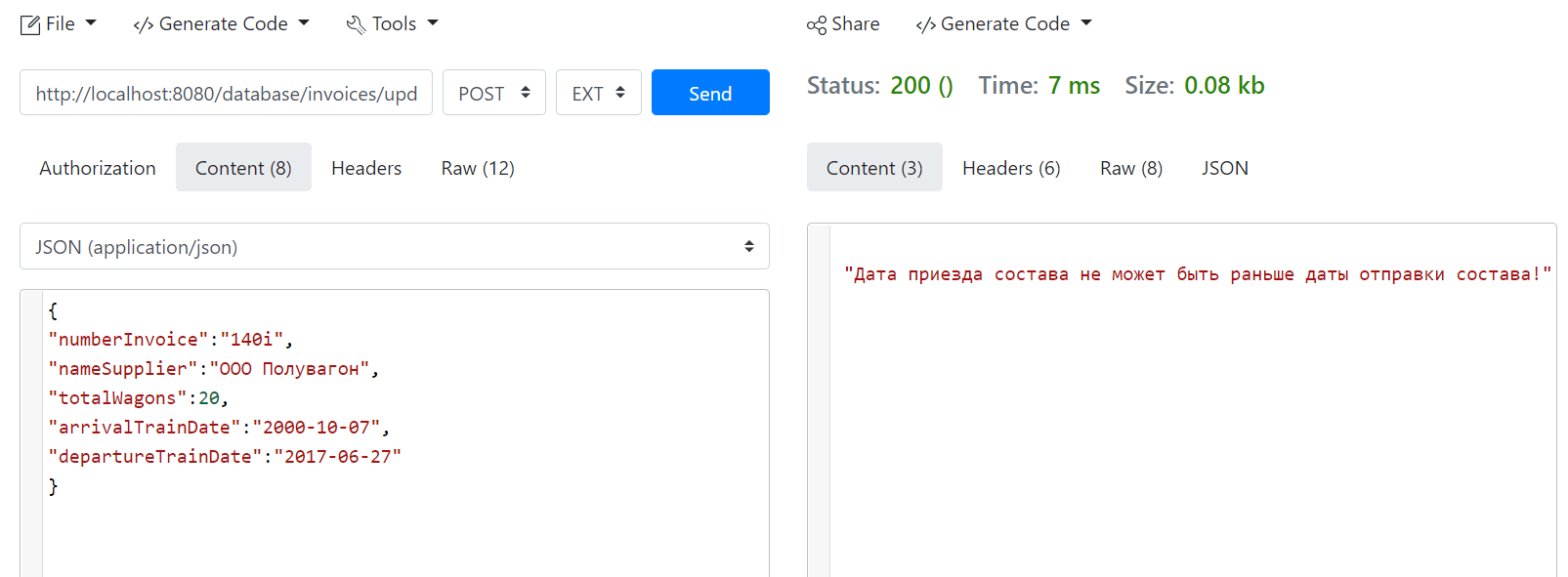


Рисунок X – Результат выполнения теста №5 из таблицы тестирования №X

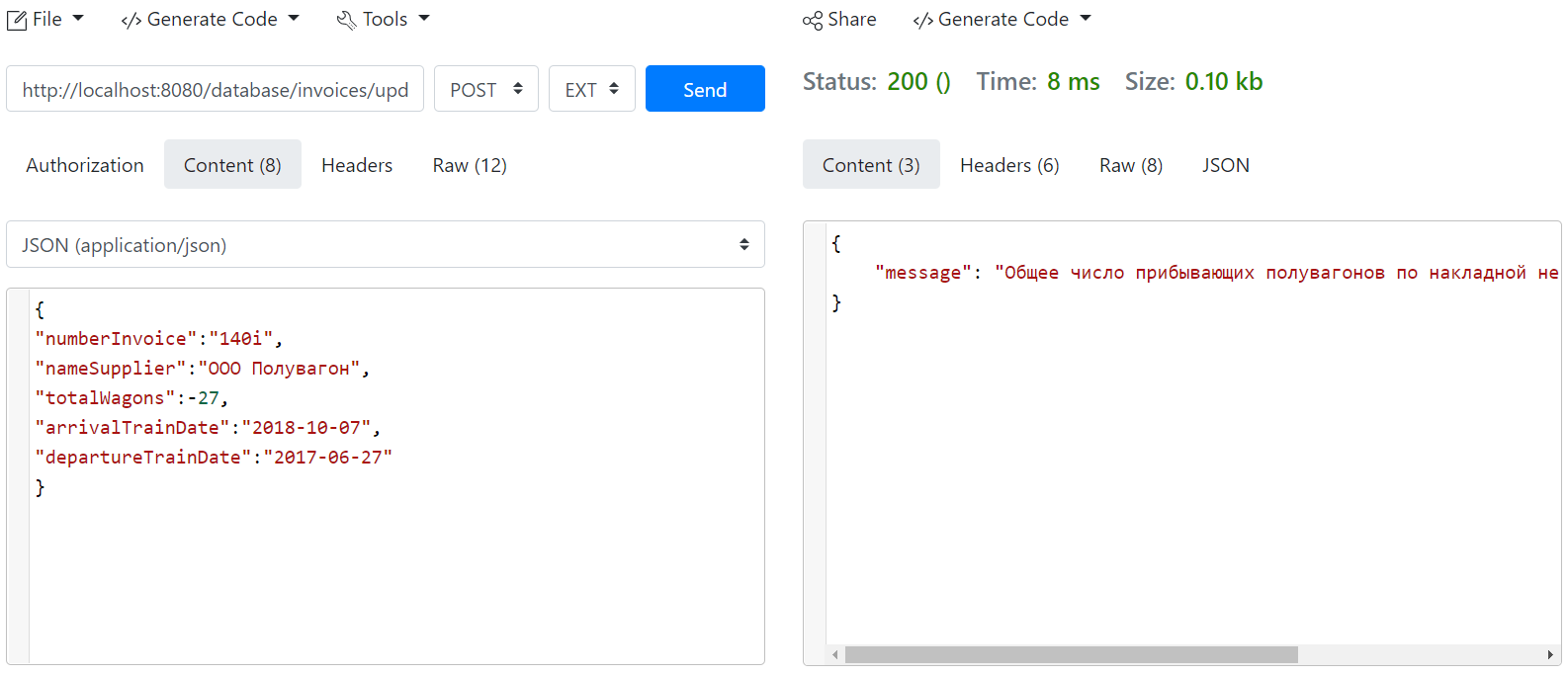


Рисунок X – Результат выполнения теста №6 из таблицы тестирования №X

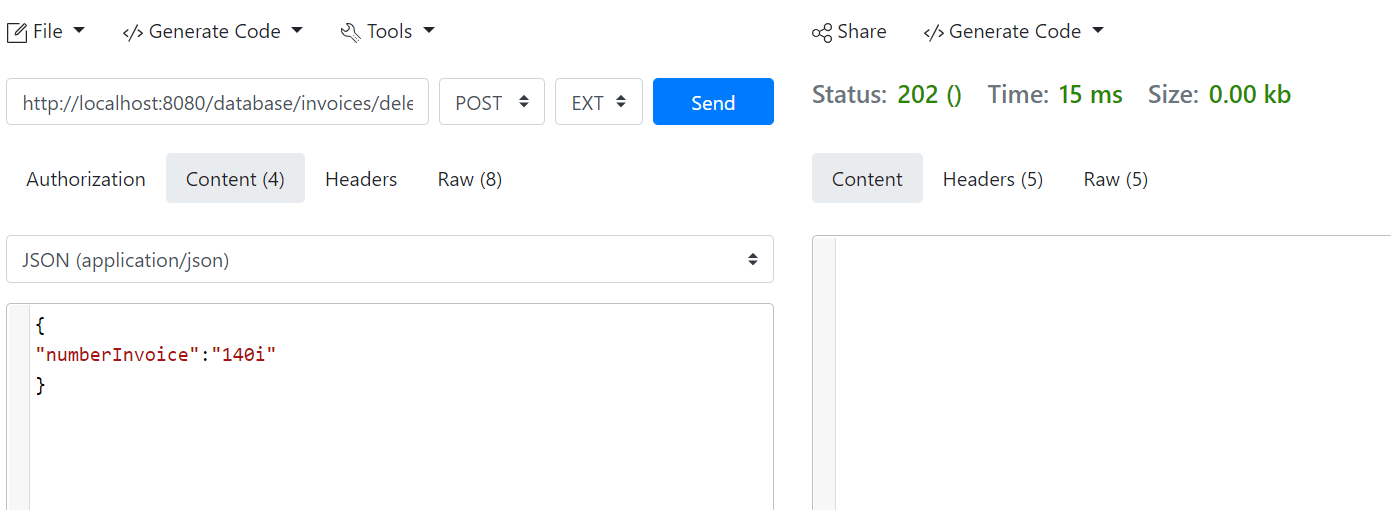


Рисунок X – Результат выполнения теста №7 из таблицы тестирования №X

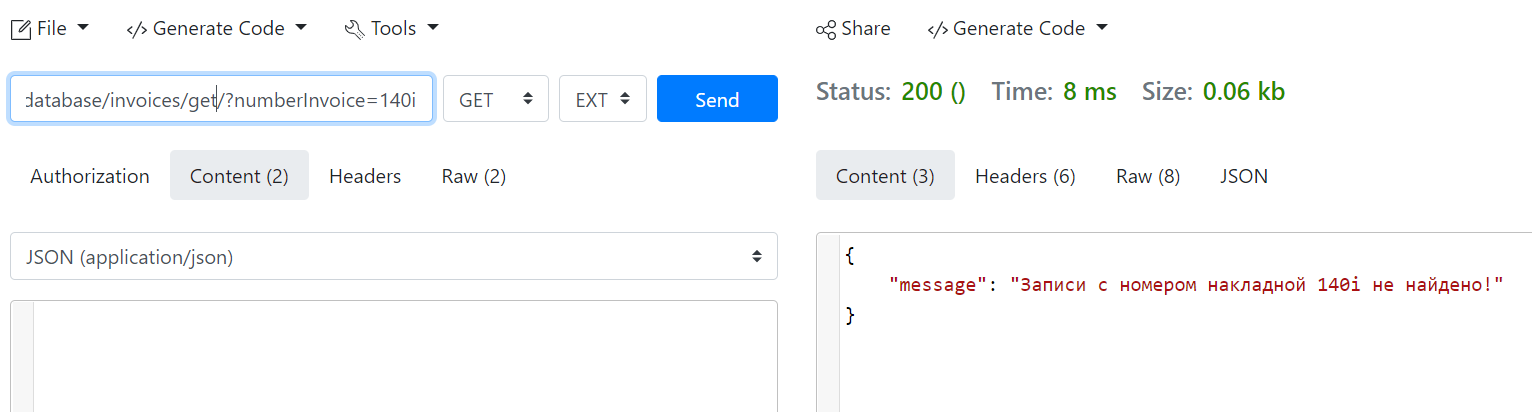


Рисунок X – Проверка на то, что после удаления записи накладной (см. рис. X) её больше нет в базе данных

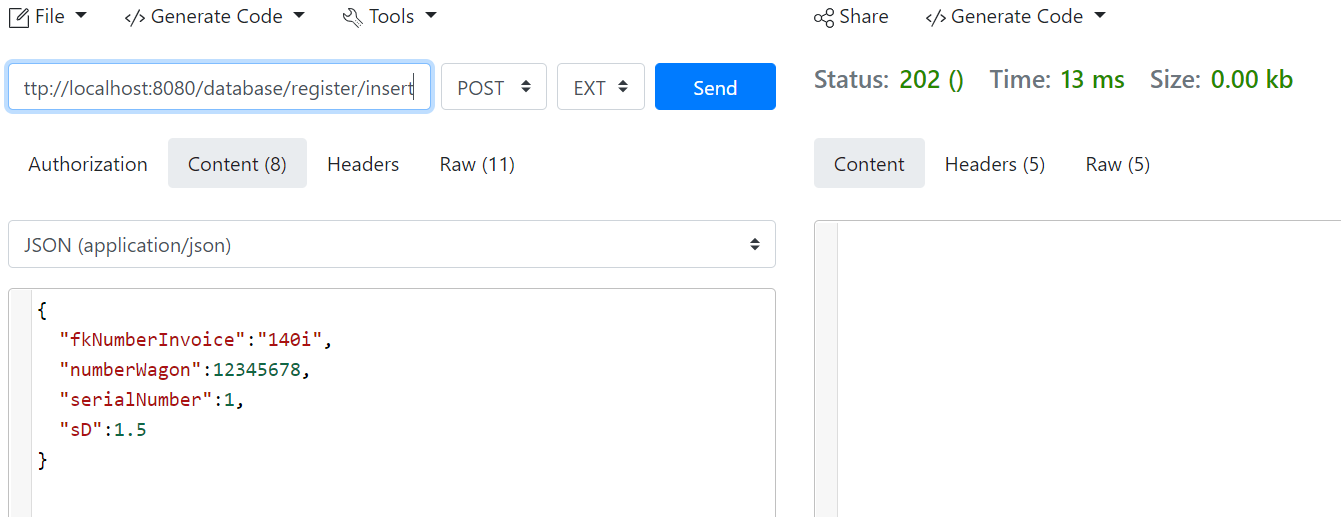


Рисунок X – Результат выполнения теста №8 из таблицы тестирования №X

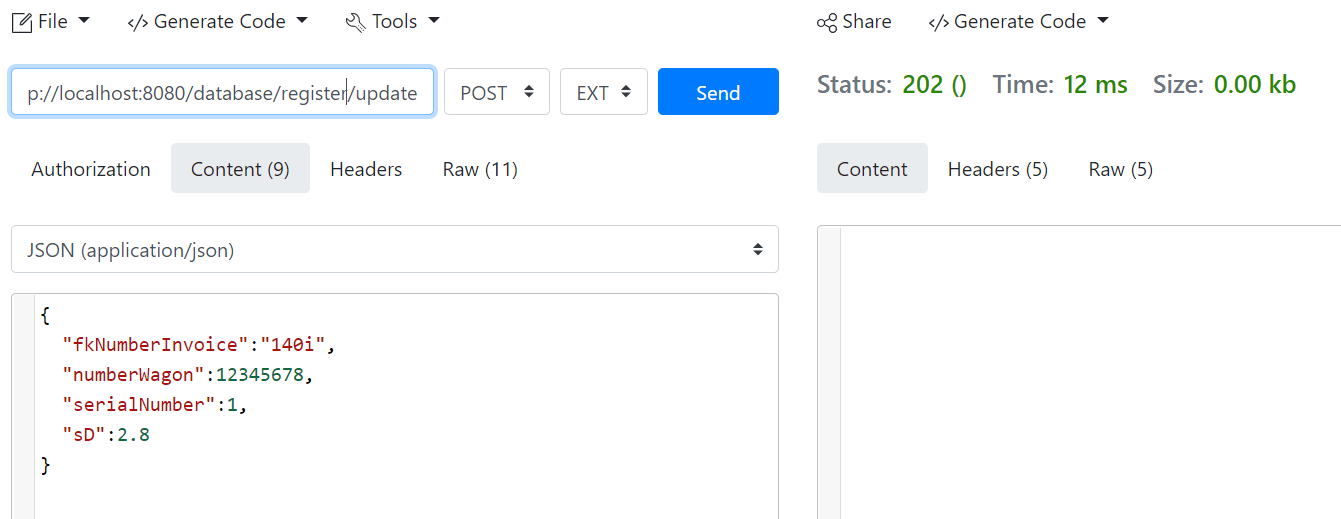


Рисунок X – Результат выполнения теста №9 из таблицы тестирования №X

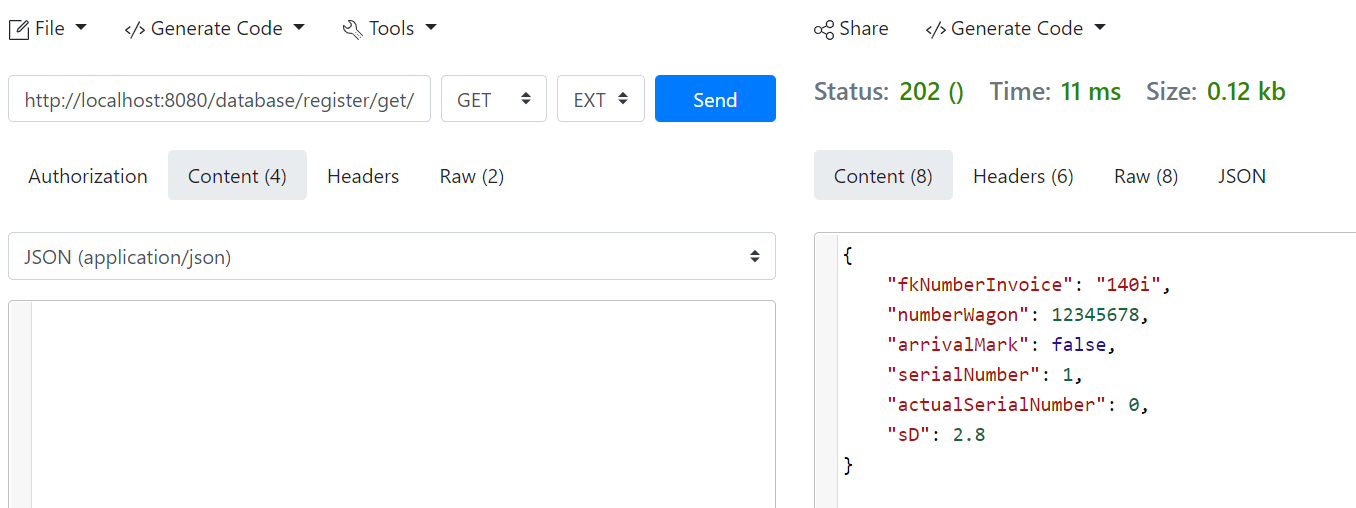


Рисунок X – Результат выполнения теста №10 из таблицы тестирования №X

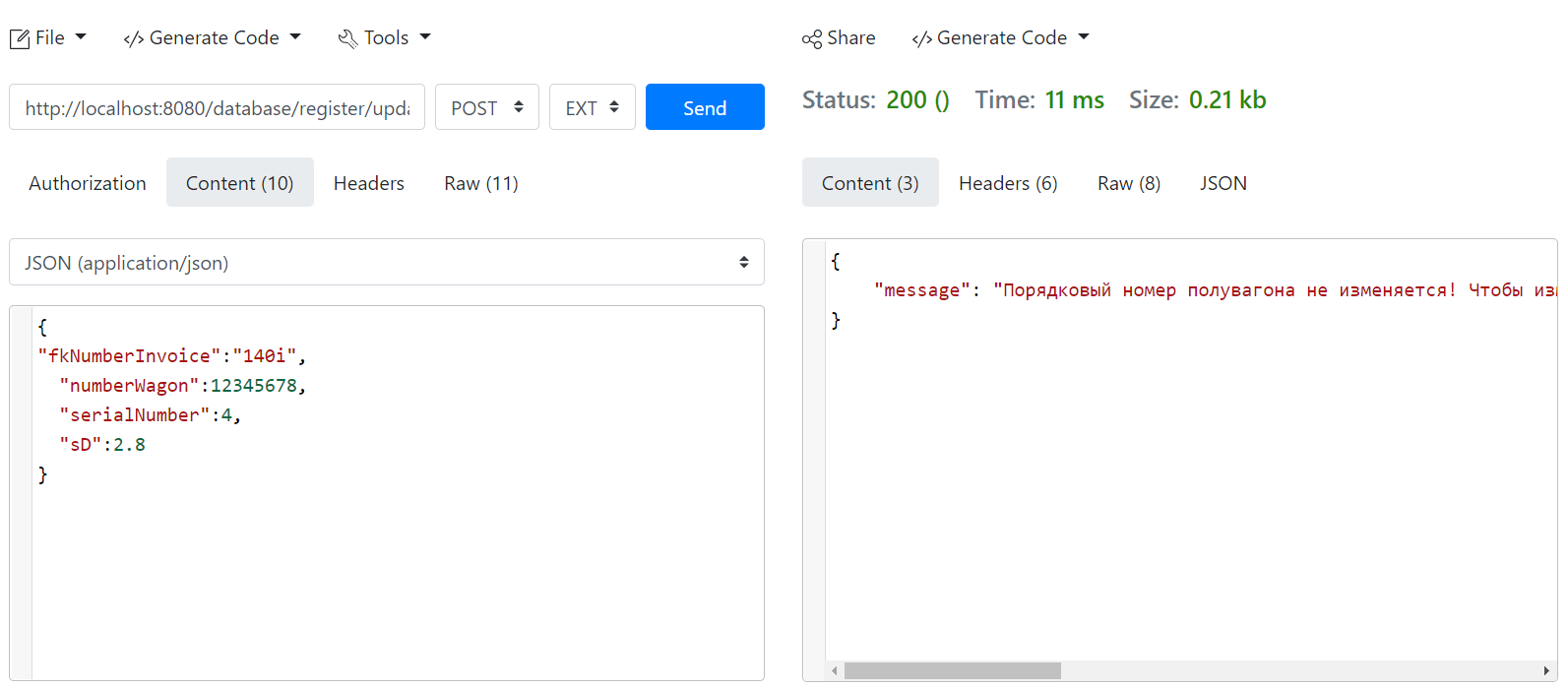


Рисунок X – Результат выполнения теста №11 из таблицы тестирования №X

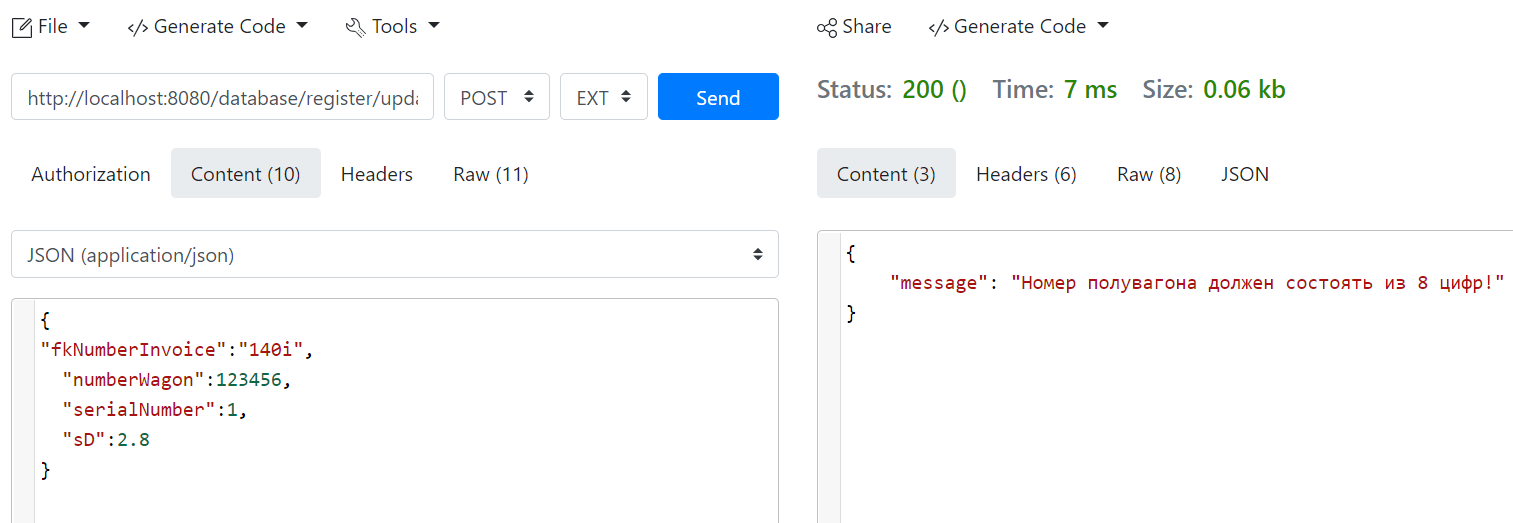


Рисунок X – Результат выполнения теста №12 из таблицы тестирования №X

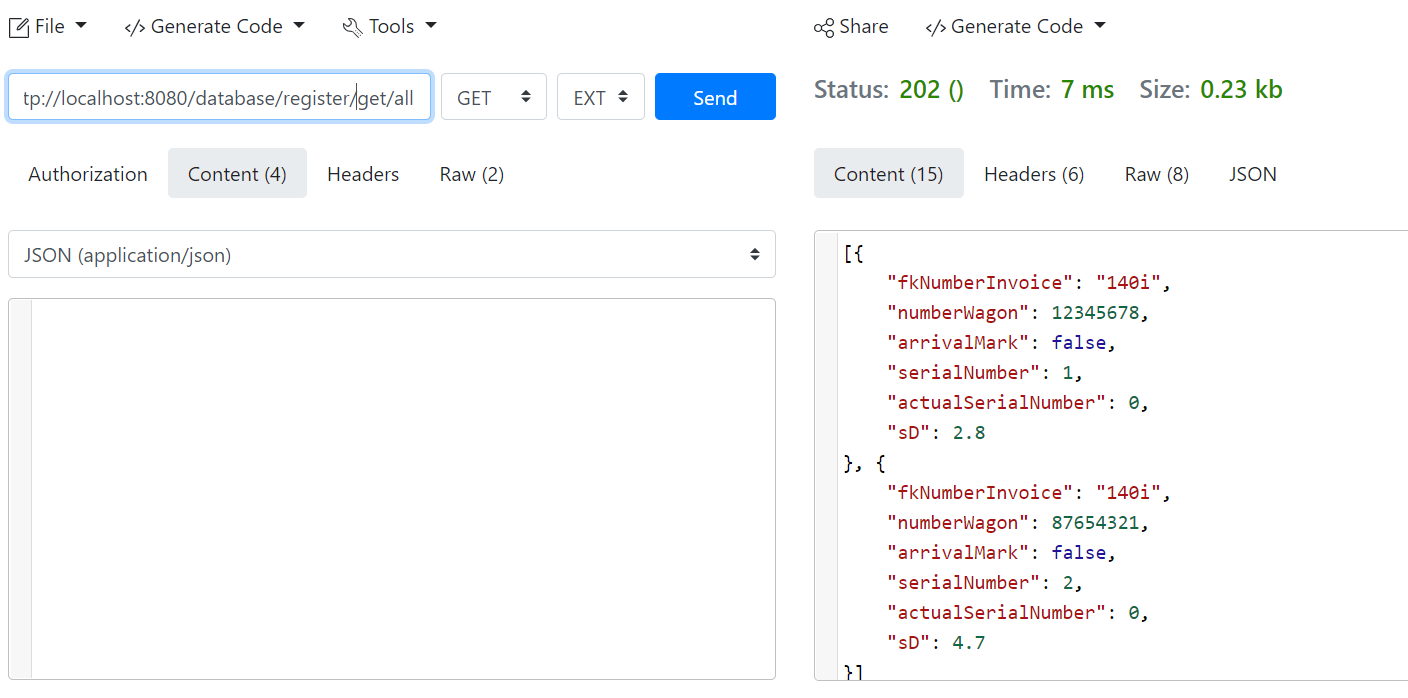


Рисунок X – Результат выполнения теста №13 из таблицы тестирования №X

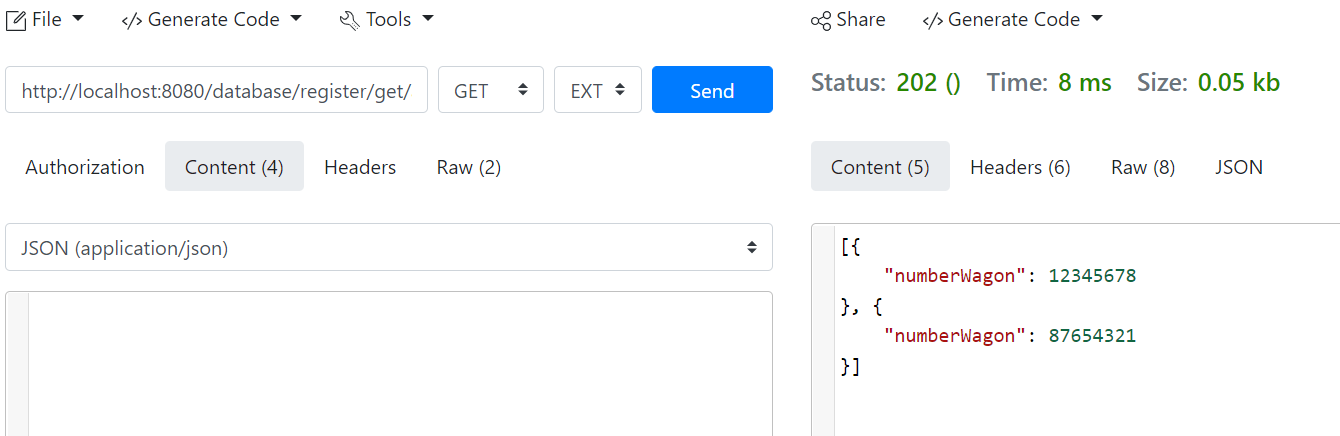


Рисунок X – Результат выполнения теста №14 из таблицы тестирования №X

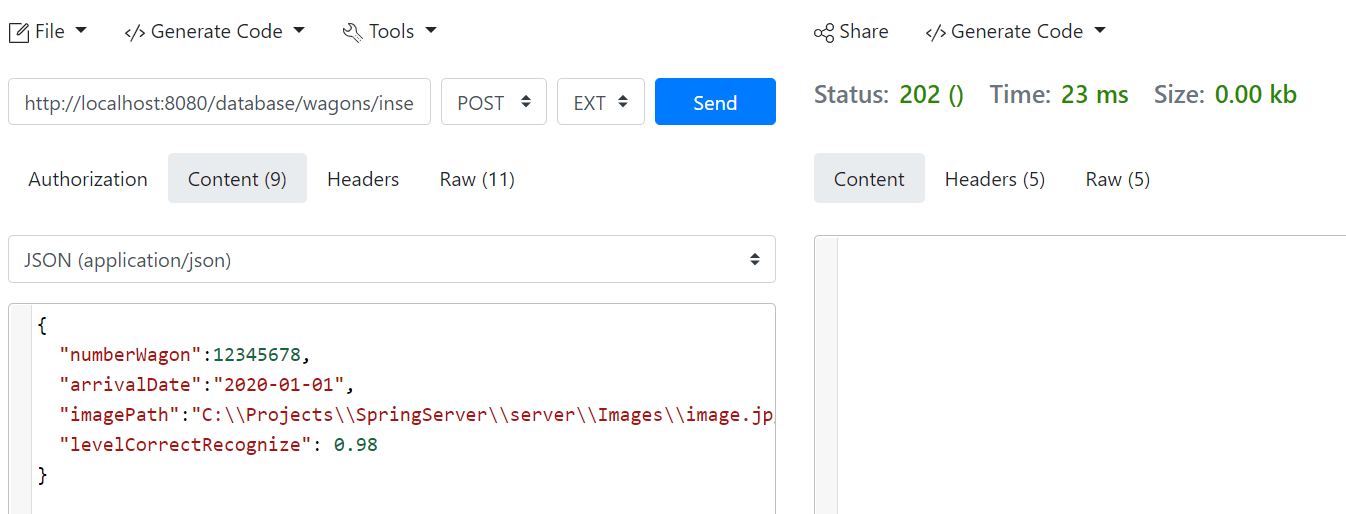


Рисунок X – Результат выполнения теста №15 из таблицы тестирования №X

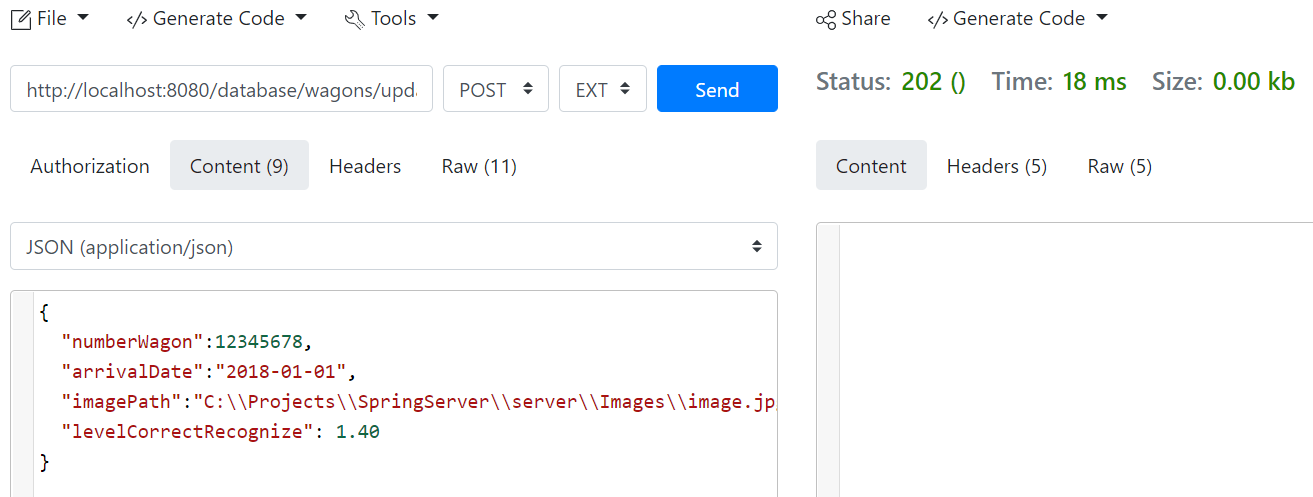


Рисунок X – Результат выполнения теста №16 из таблицы тестирования №X

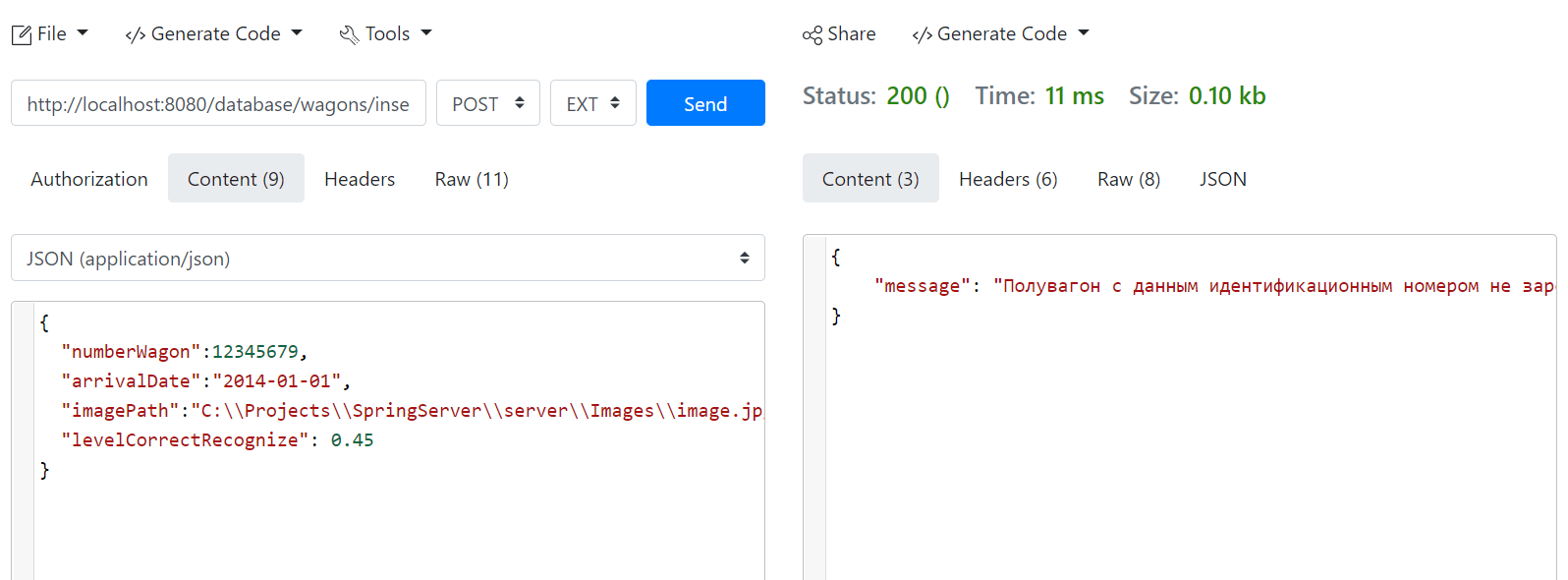


Рисунок X – Результат выполнения теста №17 из таблицы тестирования №X

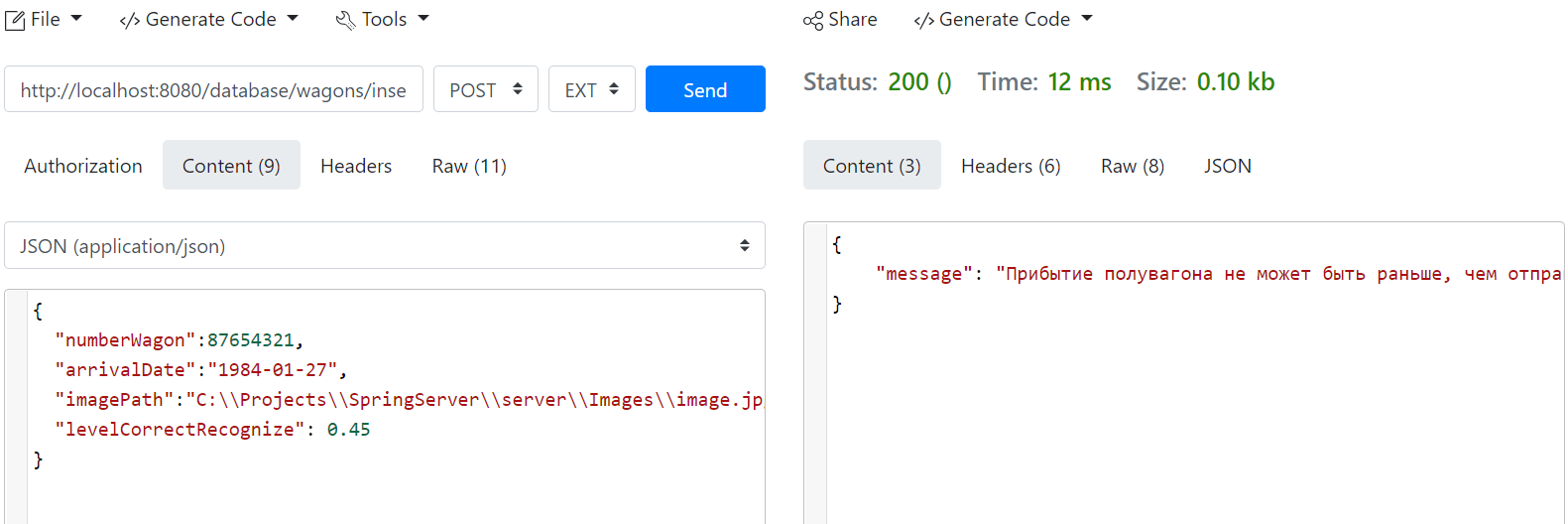


Рисунок X – Результат выполнения теста №18 из таблицы тестирования №X

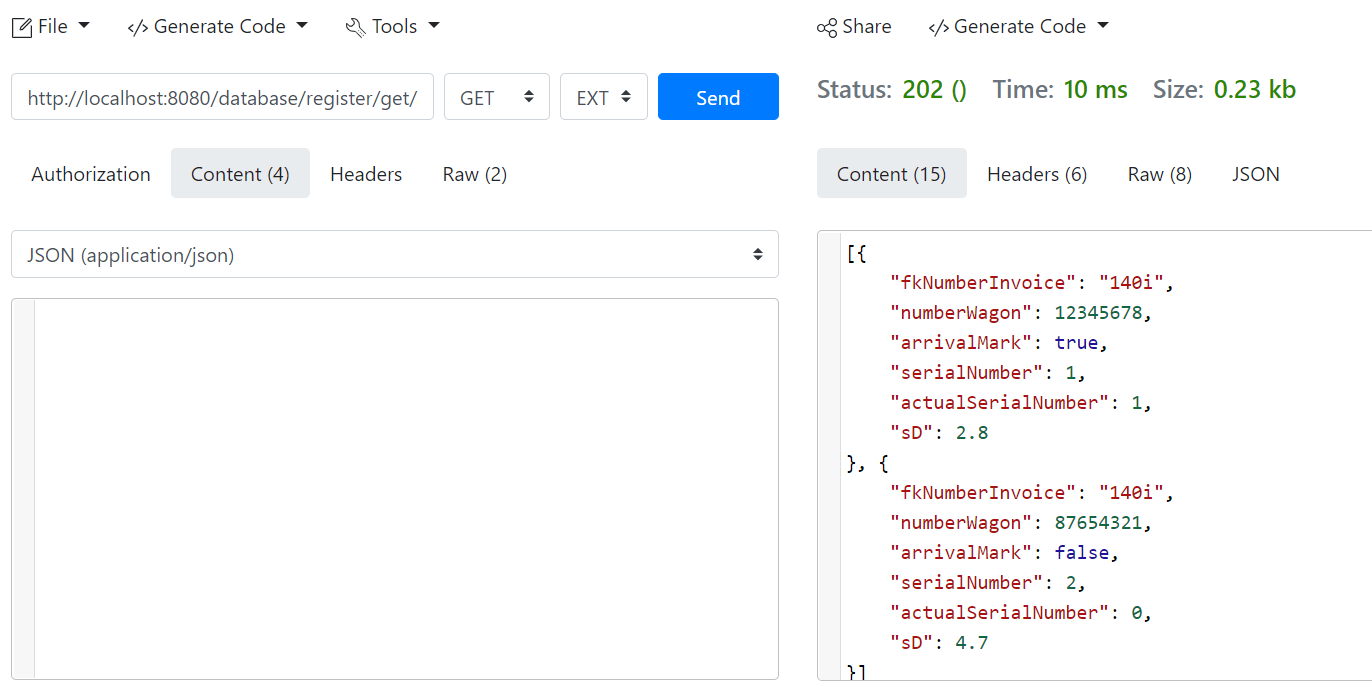


Рисунок X – Результат выполнения теста №19 из таблицы тестирования №X

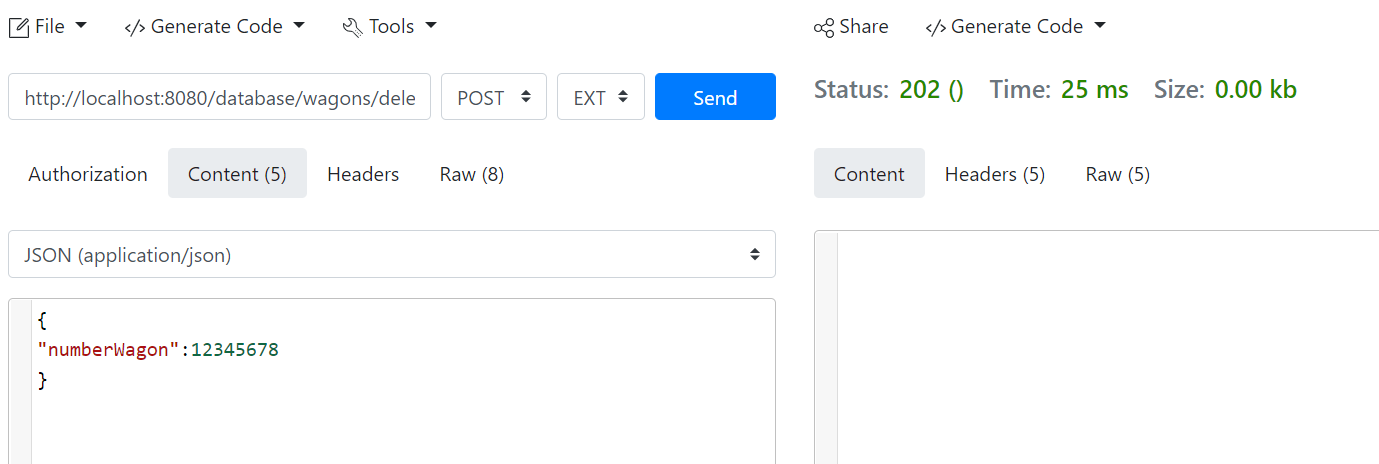


Рисунок X – Результат выполнения теста №20 из таблицы тестирования №X

ссылки

1. REQBIN. – URL: https://reqbin.com/ (дата обращения: 12.05.2021).

2. SDK распознования номеров вагонов - Intlab Wagon. – URL: https://www.intlab.com/products/intlab-wagon (дата обращения: 17.03.2021).

3. Автоматизация и роботизация производства. – URL: https://top3dshop.ru/blog/industry-automatization-with-robots.html (дата обращения: 17.03.2021).

4. Анна Немировская, Системы распознавания номеров на практике. – URL: https://habr.com/ru/company/croc/blog/158719/ (дата обращения: 17.03.2021).

5. В июле 2020 года общая численность грузовых вагонов на сети выросла на 3,8% в годовой динамике. – URL: https://finance.rambler.ru/other/44626876-v-iyule-2020-goda-obschaya-chislennost-gruzovyh-vagonov-na-seti-vyrosla-na-3-8-v-godovoy-dinamike/ (дата обращения: 18.04.2021).

6. Светлана Соболева, Как работает и где используется технология компьютерного зрения. – URL: https://blog.onlime.ru/2019/07/04/kak\_rabotaet\_tehnologia\_komputernogo\_zrenia/ (дата обращения: 18.03.2021).

7. Система распознавания номеров железнодорожных вагонов. – URL: https://iss.ru/products/securos-transit (дата обращения: 17.03.2021).