



Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатики и систем управления»

КАФЕДРА «Информационные системы и телекоммуникации»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Портирование веб-сервиса GeofenceManager и
компонента пользовательского интерфейса системы
Трассар на OSGi сервис и портлет платформы Liferay с
сохранением протокола взаимодействия клиента с
сервером

Студент группы ИУЗ-71

(подпись, дата)

В.А. Ершов

Руководитель курсовой работы

(подпись, дата)

А.М. Иванов

2017 г.

Оглавление

1 Техническое задание	3
2 Теоретическая часть	3
2.1 Выявление заинтересованных сторон и их интересов	3
2.2 Traccar	4
2.3 Liferay	5
3 Конструкторская часть	6
3.1 Выбор технических решений, удовлетворяющих интересам заинтересованных сторон	6
3.2 Структура проекта	8
3.2.1 Core-api	9
3.2.2 Core-service	9
3.2.3 Web	10
3.3 Диаграмма компонентов	10
3.4 Диаграмма классов	11
4 Технологическая часть	13
4.1 Создание бинарной сборки системы	13
4.2 Анализ исходного кода с помощью метрик качества	14
4.3 Анализ зависимостей в коде системы	16
4.4 Тестирование на корректность работы	17
4.4.1 Корректный ввод данных и добавление новой геозоны	18
4.4.2 Проверка уникальности первичного ключа geozoneId	20
4.4.3 Корректный ввод удаляемых данных и удаление геозоны	21
4.4.4 Ввод несуществующего идентификатора геозоны при удалении	22
4.4.5 Некорректный ввод идентификатора geozoneId	22
4.4.6 Некорректный ввод в поля с числовым значением	23
5 Выводы	26

1 Техническое задание

Портирование веб-сервиса XXX и компонента пользовательского интерфейса системы Traccar на OSGi сервис и портлет платформы Liferay с сохранением протокола взаимодействия клиента с сервером.

Темы индивидуальных заданий:

- изучить соответствующий Manager и его графический интерфейс в Traccar;
- спроектировать интерфейс компонента;
- реализовать хранение данных в БД (функционал должен быть инкапсулирован);
- разделение модели данных и бизнес логики;
- провести тестирование;
- описать требования, конструкцию, особенности сборки и запуска в документации;
- реализовать визуализацию данных в GUI;
- обработка событий GUI и отправка команд;
- использование CSS стилей и шаблонов.

2 Теоретическая часть

2.1 Выявление заинтересованных сторон и их интересов

В таблице ниже представлены результаты выявления и начального анализа заинтересованных сторон (ЗС) и их интересов по отношению к системе.

Таблица 1. Заинтересованные стороны и их интересы по отношению к системе

Заинтересованные стороны	Интересы заинтересованных сторон
Пользователь	П1 Возможность добавления нужного пользователю приложения в свой персональный Liferay;

	П2 Настройка интерфейса взаимодействия с сервисом; П3 Дружественный интерфейс;
Владелец опенсорсного проекта (project owner)	В1 Быстрая и полная передача исходного кода, настроек, документов. В2 Возможность производить изменения портлетов с автоматической синхронизацией с сервером. В3 Возможность в дальнейшем усложнять реализацию системы. В4 Возможность загружать различные модули, которые будут подгружаться в системе Liferay только при взаимодействии пользователя с ним.

2.2 Traccar

Traccar - это бесплатная, современная, свободно распространяемая система GPS слежения [1]. С помощью одноименного приложения Traccar Client возможно отправлять данные о своем местоположении в систему, которая будет отображать зарегистрированного пользователя на карте.

Благодаря своей реализации Traccar работает на любой платформе и с любыми устройствами (рис. 1).



Server

Traccar software provides high performance and stability on Windows, Linux or any other platform. The server can be self-hosted in the cloud or on-premise. We also provide a number of hosted options with professional support.



Devices

Traccar supports more protocols and device models than any other GPS tracking system on the market. You can select GPS trackers from a variety of vendors from low cost Chinese models to high-end quality brands.



Interface

Traccar includes a modern fully-featured web interface with both desktop and mobile-friendly layouts. We also provide native mobile apps for Android and iOS platforms. In addition to that we have a set of apps enabling mobile devices to be used as GPS trackers.



Live Tracking

With Traccar you can view your GPS devices in real-time with no delay. We have various mapping options, including road maps and satellite imagery. The Server can handle a wide variety of sensors and additional information supplied by GPS units.



Alerts

Traccar software provides instant web notifications along with support for email and SMS. This allows for external alerting in cases of harsh driving behaviour like speeding, fuel and maintenance events, geo-fencing and many other types of alerts.



Reports

Traccar supports simple location history, trip, chart and summary reports. You can view data directly in the web or mobile app and also export and download an Excel file. History can also be projected on the map providing visual representation.

Рисунок 1 – Возможности Traccar

Трассар дает возможность просматривать свои GPS-координаты в режиме реального времени без задержки. Также, как и все существующие на данный момент GPS-трекеры, Трассар предоставляет возможность отображения в всех доступных версиях отображения карты.

Документация по Трассар доступна по ссылке [1].

2.3 Liferay

Liferay Portal — программный продукт, представляющий собой корпоративный портал, то есть решение, предназначенное для централизованного доступа к нескольким различным корпоративным приложениям в одном месте. Liferay иногда описывается как система управления содержимым (CMS) или платформу для веб-приложений. Написан на языке Java и распространяется под двумя видами лицензий, свободной и проприетарной, используя бизнес-модель двойного лицензирования. Liferay Portal позволяет пользователям настроить общий доступ к разным приложениям через один-единственный сайт (рис. 2). Это реализуется с помощью функциональных модулей, называемых портлеты. Liferay поддерживает разработку портлетов на нескольких языках программирования, включая Ruby и PHP. Хотя Liferay предлагает сложный программный интерфейс для разработчиков, но для его установки и базового администрирования навыки

программирования не требуются. Liferay Portal разработан на Java и работает на любой вычислительной платформе в среде Java Runtime Environment и сервере приложений. Liferay доступен в комплекте с сервером приложений Apache Tomcat.

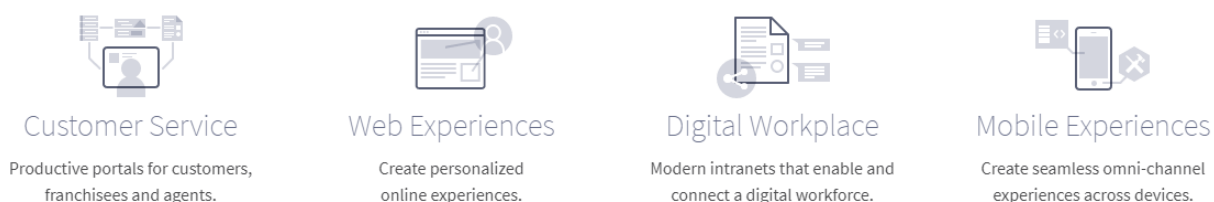


Рисунок 2 – Возможности Liferay

Сравнительно недавно Liferay был переписан с помощью новой технологии OSGi (Open Services Gateway Initiative), которая позволила убыстрить систему, а так же перевести ее в более удобно реализуемый формат. Теперь каждый отдельный модуль системы является так называемым бандлом (bundle) или же плагин (plug-in). Это позволило:

- Усовершенствовать разработку веб-контента;
- Шаблонизировать отображение приложений;
- Использовать декларативные сервисы Service Builder-ом, вместо Spring для инъекции зависимостей.

OSGi ввести в приложение термин “жизненного цикла” приложения, что позволило разбить создаваемый функционал и подгружать только те его части, которые необходимы для решения задачи в поставленный момент времени. Так же, есть возможность загружать плагины в Liferay и выгружать их из системы непосредственно без остановки работы системы [4].

Документация по Liferay доступна по ссылке [2].

3 Конструкторская часть

3.1 Выбор технических решений, удовлетворяющих интересам заинтересованных сторон

В таблице ниже представлены результаты выбора технических решений, позволяющие удовлетворить интересы заинтересованных сторон по отношению к системе.

Таблица 2. Заинтересованные стороны и их интересы по отношению к системе

Интересы заинтересованных сторон	Технические решения
<p>П1 Возможность добавления нужного пользователю приложения в свой персональный Liferay;</p> <p>П2 Настройка интерфейса взаимодействия с сервисом;</p> <p>П3 Дружественный интерфейс;</p>	<p>Возможно добавления портлета будет у каждого пользователя, зарегистрировавшегося на Liferay.</p> <p>Настройка интерфейса будет произведена с помощью возможности загружать на страницу нужный функционал и удалять не нужный.</p> <p>П3 С помощью встроенного интерфейса вывода ошибок пользователь будет получать сообщения о неверно введенных данных, также интерфейс будет интуитивно понятным – все строки вводимых данных будут иметь емкое название, отображающее суть данного поля.</p>
<p>В1 Быстрая и полная передача исходного кода, настроек, документов.</p> <p>В2 Возможность производить изменения портлетов с автоматической синхронизацией с сервером.</p>	<p>В1 Самодокументируемый код с использованием Google Code Style.</p> <p>В2 Опция авто-выгрузки на сервер реализована с помощью ServiceBuilder-а, что позволит создавать портлеты, которые автоматически будут отображать изменения на сайте Liferay вашего приложения.</p>

В3 Возможность в дальнейшем усложнять реализацию системы.	Реализация системы основана на принципах модульности, что позволяет разработчику сколь угодно расширять ее функционал.
В4 Возможность загружать различные модули, которые будут подгружаться в системе Liferay только при взаимодействии пользователя с ним, что позволяет сделать систему более легковесной и быстрой.	С помощью встроенной фабрики сервисов Liferay будет подгружать именно тот плагин, который непосредственно необходим для воздействия с пользователем.

3.2 Структура проекта

Проект состоит из 2-х модулей:

- com.bmstu.geofence.manager.core;
 - com.bmstu.geofence.manager.core-api
 - com.bmstu.geofence.manager.core-service
- com.bmstu.geofence.manager.web.

Core-api и Core-service создавались с помощью Service-Builder - это инструмент генерации кода, созданный Liferay, который позволяет разработчикам определять пользовательские модели, называемые объектами.

Service Builder создает сервисный уровень с помощью технологии объектно-реляционного сопоставления (ORM), которая обеспечивает чистое разделение между вашей объектной моделью и кодом для базы данных. Это освобождает от добавления необходимой бизнес-логики для приложения. Service Builder берет XML-файл в качестве входных данных и генерирует необходимую модель, уровни обслуживания для приложения. Service Builder генерирует большую часть общего кода, необходимого для реализации операций создания, чтения, обновления,

удаления и поиска в базе данных, что позволяет сосредоточиться на аспектах дизайна более высокого уровня.

Service Builder создает проект по инструкции, написанной в service.xml. В данном файле указываются поля заданной модели. Необходимые поля взаимодействия Трассар и сервера указаны на сайте описания API [5]. Таким образом было необходимо внести поля, отображенные на рисунке 3.

```
<!-- Status fields -->
<column name="status" type="int" />
<column name="statusByUserId" type="long" />
<column name="statusByUserName" type="String" />
<column name="statusDate" type="Date" />

<!-- Other fields -->
<column name="name" type="String" />
<column name="area" type="String" />
<column name="calendarId" type="long" />
<column name="description" type="String" />
<column name="geozoneAttributes" type="String" />
```

Рисунок 3 – Добавляемые поля файла service.xml

3.2.1 Core-api

Модуль содержит API для проекта GeofenceManager. Все классы и интерфейсы в core-api модуле упакованы в .jar файл, GeofenceManager-api.jar, указанный в build/libs папке модуля. Этот .jar файл генерируется всякий раз, когда компилируется и разворачивается сервисный модуль. При разворачивании этого JAR в Liferay доступны необходимые интерфейсы для определения API сервиса.

3.2.2 Core-service

Модуль содержит реализацию интерфейсов, определенных в модуле GeofenceManager-api. Эти интерфейсы предоставляют службы OSGi для экземпляра портала, на котором разворачивается приложение. Service Builder генерирует классы и интерфейсы, относящиеся к уровню сохранения, уровню обслуживания и уровню модели.

Ключевым классом данного модуля является GeofenceLocalServiceImpl.java, который обрабатывает полученный от портлета запрос и заносит новые данные в базу данных. Данный класс наследуется от GeofenceLocalServiceBaseImpl.java и переопределяет ранее определенные методы наследуемого класса. Это позволило вносить нужные нам изменения в интерфейс сайта, который далее будут

передаваться в качестве данных в базу данных. Одним из таких переписанных методов является метод `addGeozone`, который позволяет добавлять геозону в базу данных в формате отображения на сайте, тогда как стандартный метод добавления геозоны предполагал уже добавление сформированного объекта типа `Geozone`.

3.2.3 Web

Модуль содержит реализацию MVC-портлета. Портлеты являются одним из основных базовых блоков `Liferay Portal`. `Liferay Portal` предоставляет платформу, которая содержит общие функции, необходимые для современных приложений, включая управление пользователями, безопасность, пользовательские интерфейсы, службы и многое другое.

В классе `GeofenceManagerPortlet.java` описываются методы обработки запроса на добавление и удаление группы `addGeofence` и `deleteGeofence`. Эти методы посредством запроса с сайта (`response`) преобразуют вводимые пользователем данные в удобный формат для взаимодействия с ними. Так удаление геозоны производится с помощью уникального идентификатора “`geofenceId`”, который вводится пользователем на основе отображаемых данных портлета.

Два файла `view.jsp` и `edit_geofence.jsp` описывают 2 состояния портлета – основное и состояние добавления новой зоны. Так же существует файл `delete_geofence.jsp`, который является третьим отображением системы – удаление геозоны.

3.3 Диаграмма компонентов

Предполагается, что данный проект может использоваться как объяснение взаимодействия `Traccar Geofence` и сервера с данными, поэтому в данном проекте эмулируется поведение, а также хранение данных `Traccar Geofence Manager`-а с отображением данных в виде портлета `Liferay`. Данная структура проекта отображена на рисунке 4.

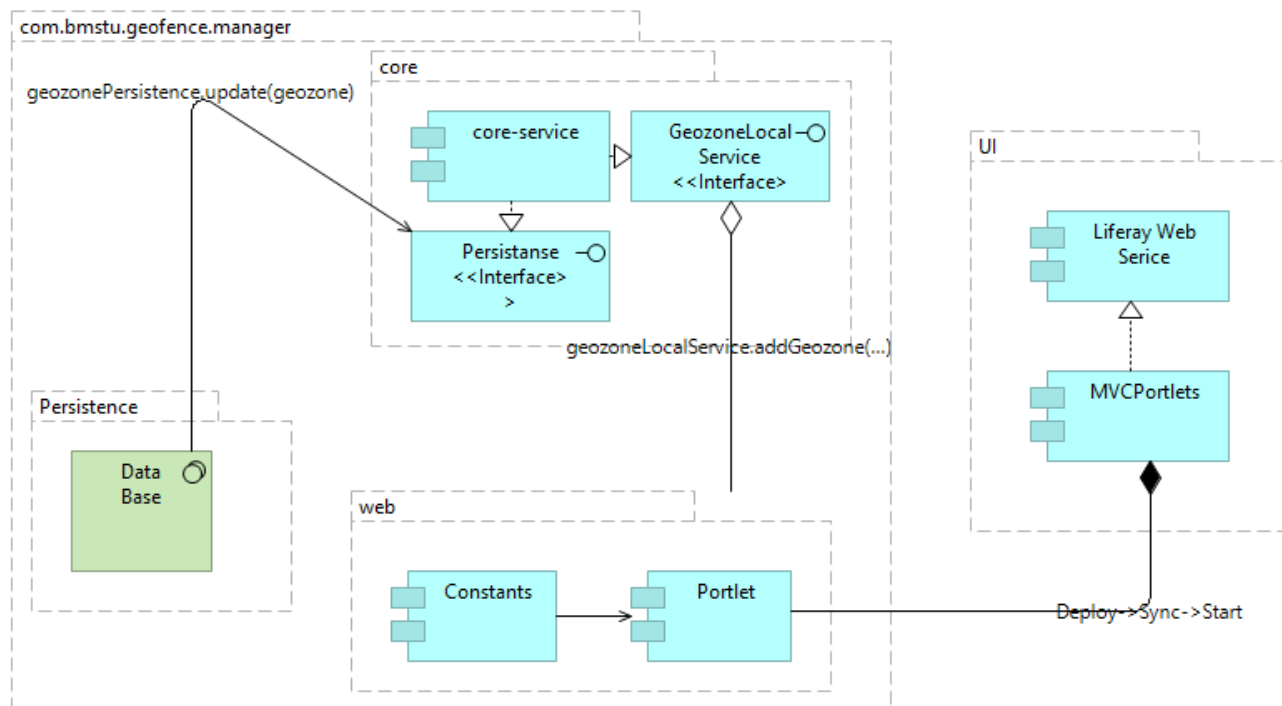


Рисунок 4 – Диаграмма компонентов

Данная диаграмма (рис. 4) отображает упрощенный функционал данной системы. Основным запускающим действием данного проекта является взаимодействие пользователя с UI – user interface. Его взаимодействие будет направлено на web-страницу, которая реализована с помощью MVCPortlet. Данный портлет является результатом бинарной сборки с помощью фреймворка Gradle (подробнее п. 4.1). Функционал портлета, который отображается на web-страничке реализован в пакете с названием *.web, который содержит методы добавления (addGeofence) и (deleteGeofence), реализуемые на основе запроса и ответа типа ActionRequest и ActionResponse, позволяющие получать данные с web-странички. Основной метод добавления геозоны реализован в GeozoneLocalServiceImpl, который был переписан с теми конфигурируемыми полями, которые получают с сайта Liferay. Данный класс реализует формирование геозоны типа Geozone и добавление ее в базу данных с помощью экземпляра и метода geozonePersistence.update(geozone).

3.4 Диаграмма классов

На рисунке 5 представлена диаграмма классов. Так как данная диаграмма является общей и отражает взаимодействие модулей *.core и *.web, то детальное рассмотрение будет основано на рисунках 6 и 7.

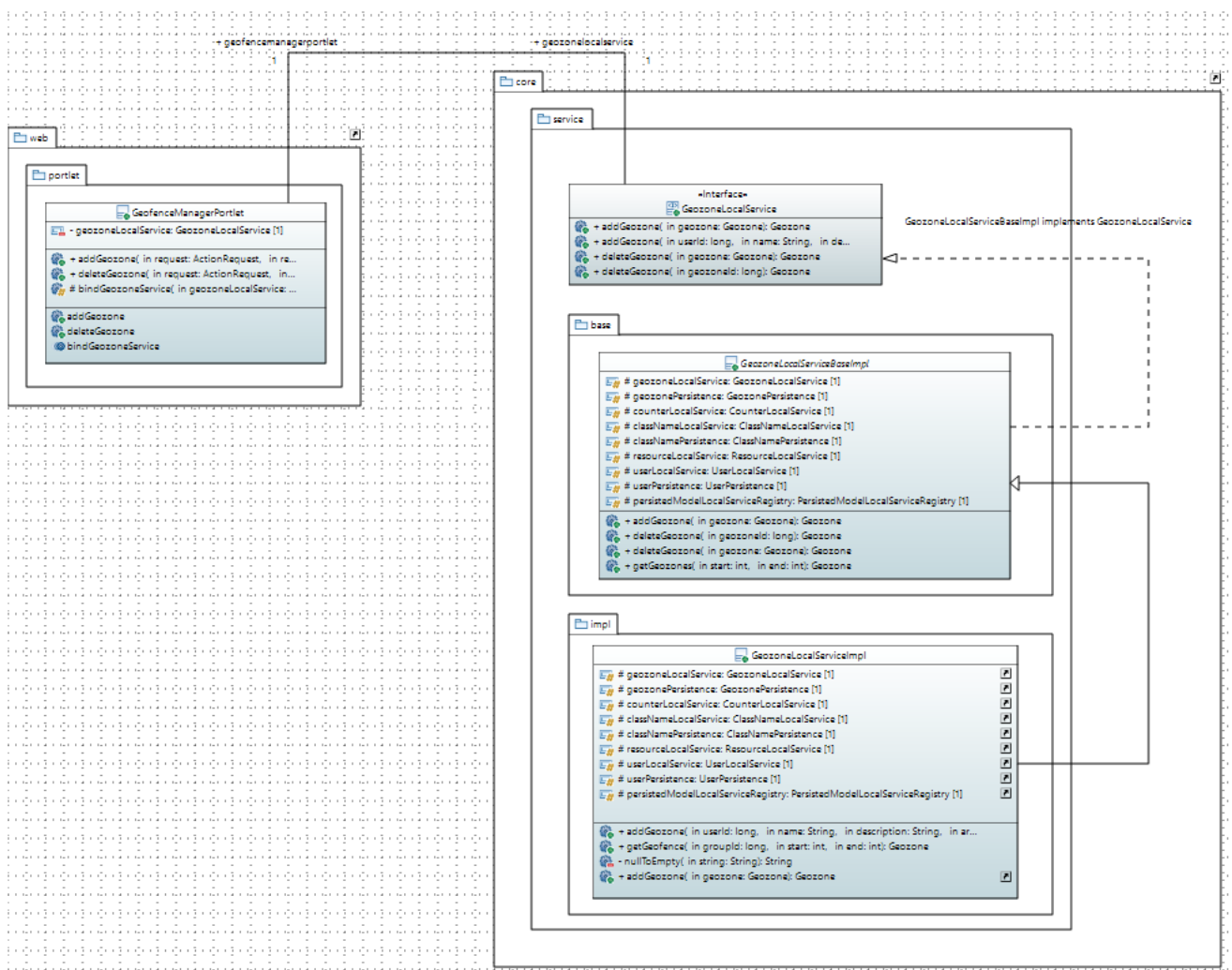


Рисунок 5 – Общая диаграмма классов

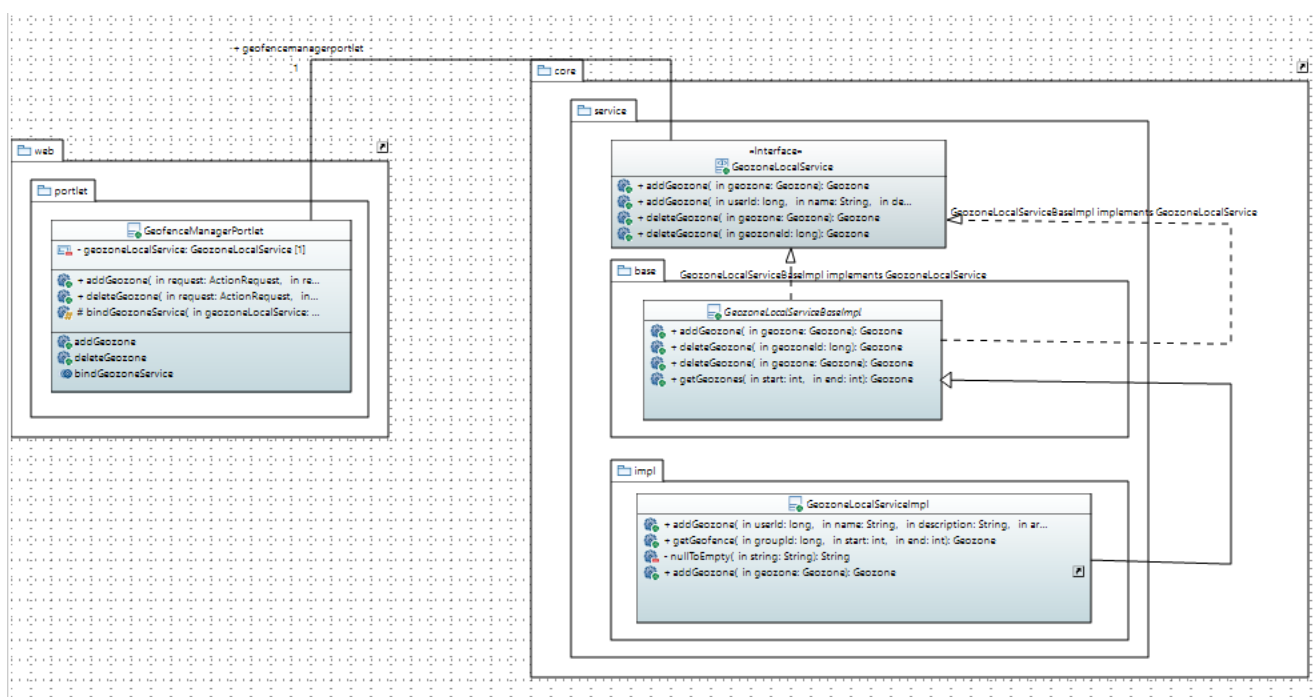


Рисунок 6 – Упрощенная диаграмма классов

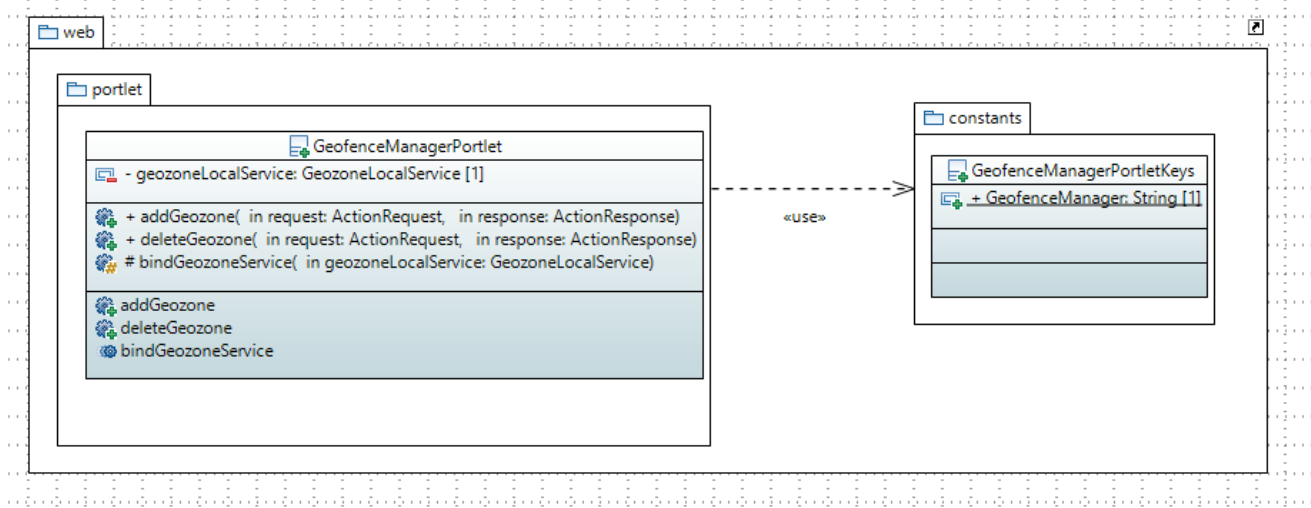


Рисунок 7 – Диаграмма классов *.web

На диаграмме классов описана общая структура проекта, которая соответствует диаграмме компонентов. Вынесено два модуля:

1. Взаимодействие *.web и *.core (рис. 5-6);
2. Взаимодействие внутри *.web.

Как и было описано в п. 3.3, портлет `GeofenceManagerPortlet` взаимодействует через интерфейс `GeozoneLocalService` с классом, который реализует добавление информации в базу данных, поля и методы взаимодействия которой сгенерированы `service.xml`.

Взаимодействие внутри пакета *.web ограничивается заимствованием константы типа `String` из `GeofenceManagerPortletKeys`. Класс `GeofenceManagerPortlet` наследуется от класса `MVCPortlet`, который является базовым классом портлета в Liferay.

4 Технологическая часть

4.1 Создание бинарной сборки системы

Исходный код проекта доступен в репозитории GitHub [3].

Для сборки проекта используется фреймворк для автоматизации сборки Gradle. Для запуска проект требуется сервер Liferay версии 7 и выше. Чтобы создать сборку системы, нужно выполнить Build-task фреймворка Gradle, далее Deploy, для создания исполняемых файлов, которые будут загружены на сервер.

4.2 Анализ исходного кода с помощью метрик качества

На рисунке 8 показано соотношение классов проекта по их размеру. Видно, что самый большой размер имеют классы пакета *.core.

Далее на рисунке 9 отображен список всех метрик по разделам. Всего имеется четыре раздела:

- метрики количества (Count);
- метрики сложности (Complexity);
- метрики Роберта Мартина (Robert C. Martin);
- метрики Чидамбера-Кемерера (Chidamber & Kemerer).

Первый раздел с метриками количества (Count) содержит следующие метрики:

- количество классов верхнего уровня (Unit);
- среднее число внутренних классов на класс (Classes / Class);
- среднее число методов в классе (Methods / Class);
- среднее число полей в классе (Fields / Class);
- число строчек кода (ELOC);
- число строчек кода на модуль (ELOC / Unit).

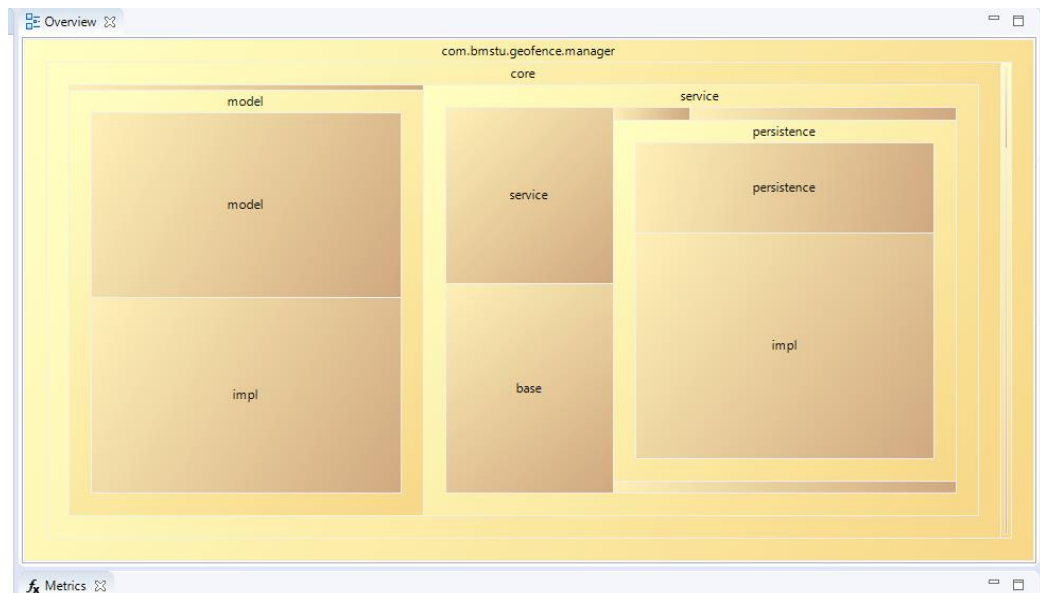


Рисунок 4 – Соотношение классов по размеру

Второй раздел с метриками сложности (Complexity) содержит всего три различных метрики:

- средняя циклическая сложность (CC);
- метрика Fat (Fat);
- средняя зависимость компонентов между модулями (ACD - Unit).

Третий раздел с метриками Роберта Мартина содержит следующие метрики:

- нормализованное расстояние от основной последовательности (D);
- абстрактность (A);
- нестабильность (I);
- число афферентных соединений (Ca);
- число эфферентных соединений (Ce).

Category/Metric	Value
▼ Count	
Libraries	3
Packages	12
Units	27
Units / Package	2.25
Classes / Class	0
Methods / Class	22.84
Fields / Class	5.1
ELOC	4325
ELOC / Unit	160.19
▼ Complexity	
CC	1.27
Fat - Libraries	2
Fat - Packages	18
Fat - Units	53
Tangled - Libraries	0.00%
ACD - Library	33.33%
ACD - Package	17.42%
ACD - Unit	13.96%
▼ Robert C. Martin	
D	-0.19
D	0.33
▼ Chidamber & Kemerer	
WMC	29
DIT	1.19
NOC	0.16
CBO	1.11
RFC	34.39
LCOM	227.26

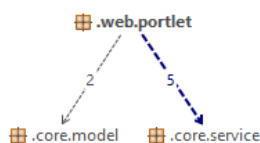
Рисунок 9 – Значение метрик

Последний раздел с метриками Чидамбера-Кемерера содержит следующие метрики:

- средняя длина метода на класс (WMC);
- средняя глубина наследования (DIT);
- среднее количество классов-наследников (NOC);
- среднее число соединений класса (CBO);
- среднее число методов, которые потенциально могут быть выполнены в ответ на сообщение, полученное объектом этого класса (RFC);
- отсутствие единства методов (LCOM).

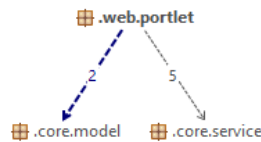
4.3 Анализ зависимостей в коде системы

Рисунки 10-11 характеризуют зависимости между пакетами проекта и внешними пакетами.



Violations Dependencies Properties Query Map		
Source	-->	Target
.web.portlet.GeofenceManagerPortlet	contains	.core.service.GeozoneLocalService
.web.portlet.GeofenceManagerPortlet.addGeozone(ActionRequest, ActionR...	calls	.core.service.GeozoneLocalService.addGeozone(long, String, String, String, ...)
.web.portlet.GeofenceManagerPortlet.bindGeozoneService(GeozoneLocalSe...	has param	.core.service.GeozoneLocalService
.web.portlet.GeofenceManagerPortlet.deleteGeozone(ActionRequest, Action...	calls	.core.service.GeozoneLocalService.deleteGeozone(long)
.web.portlet.GeofenceManagerPortlet.geozoneLocalService	is of type	.core.service.GeozoneLocalService

Рисунок 10 – Зависимости *.web от *.core.service



Source	-->	Target
• .web.portlet.GeofenceManagerPortlet.addGeozone(ActionRequest, ActionR...	references	• .core.model.Geozone
• .web.portlet.GeofenceManagerPortlet.deleteGeozone(ActionRequest, Action...	references	• .core.model.Geozone

Рисунок 11 – Зависимости *.web от *.core.model

Главные зависимости, а так же предоставляемые методы и интерфейсы общения модулей показаны на рисунке 12. По рисунку можно убедиться, что диаграмма компонентов (рис. 4) и диаграмма классов (рис. 5) были составлены верно и отображают суть взаимодействия модулей проекта GeofenceManager.

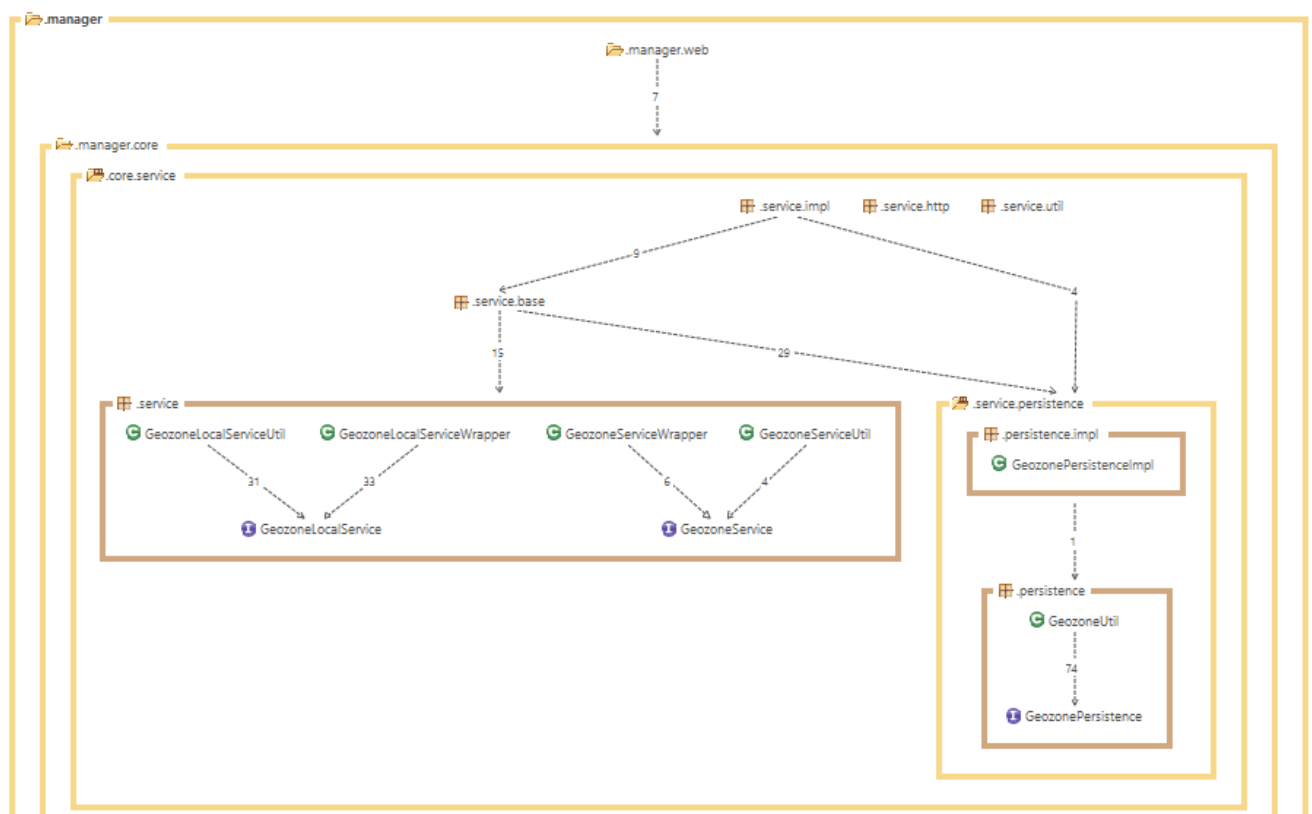


Рисунок 12 – общие зависимости модулей *.web и подмодулей *.core

4.4 Тестирование на корректность работы

В качестве тестирования было принято применить метод ручного тестирования. Для этого будет проверяться выполнение всех требований

технического задания (в упрощенном варианте, объединяя схожие компоненты в одну графу), а именно:

- корректный ввод данных и добавление новой геозоны;
- проверка уникальности первичного ключа `geozoneId`;
- корректный ввод удаляемых данных и удаление геозоны;
- ввод несуществующего идентификатора геозоны при удалении;
- некорректный ввод идентификатора `geozoneId`;
- некорректный ввод в поля с числовым значением;

4.4.1 Корректный ввод данных и добавление новой геозоны

Изначально было сконфигурирован `view.jsp` файл, который должен выводить добавленные в базу данных данные в виде таблицы с параметрами, указанными на рисунке 13.

```
<liferay-ui:search-container-row  
  className="com.bmstu.geofence.manager.core.model.Geozone"  
  modelVar="geozone">  
  
  <liferay-ui:search-container-column-text property="name" />  
  
  <liferay-ui:search-container-column-text property="description" />  
  
  <liferay-ui:search-container-column-text property="area" />  
  
  <liferay-ui:search-container-column-text property="calendarId" />  
  
  <liferay-ui:search-container-column-text property="geozoneAttributes" />  
  
  <liferay-ui:search-container-column-text property="geozoneId" />  
  
</liferay-ui:search-container-row>
```

Рисунок 13 – Поля выводимой на web-страницу таблицы файла `view.jsp`

Таким образом, на этапе добавления портлета на web-страницу для отображения будут доступны только кнопки Add Geozone и Delete Geozone, так как в базе данных портлета еще нет записей (рис. 14).

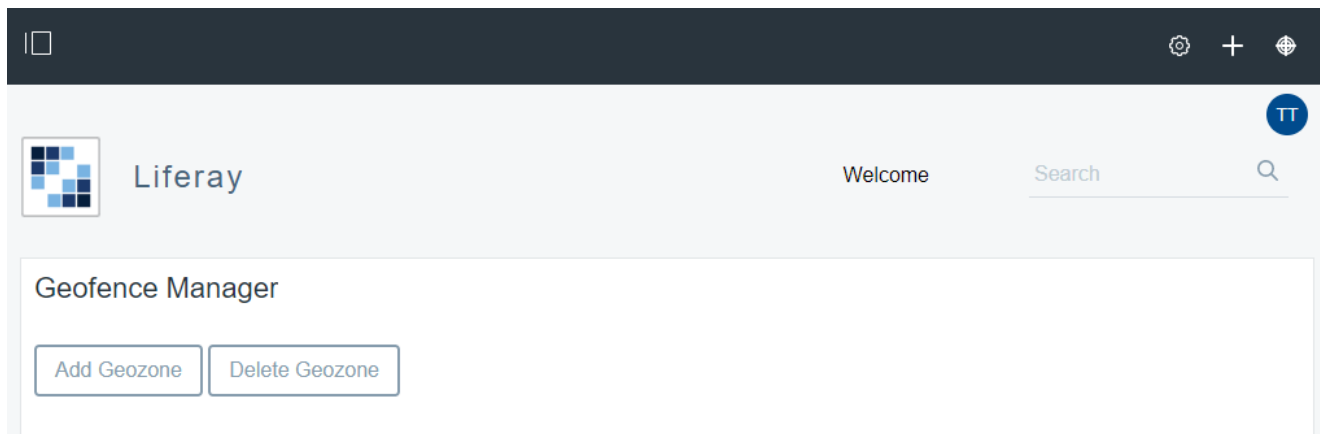


Рисунок 14 – Добавление созданного портлета в Liferay

Попробуем создать зону. При нажатии кнопки “Add Geozone” должна появиться новая web-страница со следующими заполняемыми формами (после знака равно значения, которыми мы заполним первую запись в данной таблице) (рис. 15):

- name = testN;
- description = testD;
- area = testA;
- calendarId = 0 (default value);
- geozoneAttributes = testAt
- geozoneId недоступно для редактирования, потому что является

автоматически генерируемым значением.

Рисунок 15 – Форма, появившаяся после нажатия на кнопку “Add Geozone”.

Заполнение формы значениями

Должна обновиться таблица и добавиться новые данные, что и произошло (рис 16), значит функционал добавления работает верно.

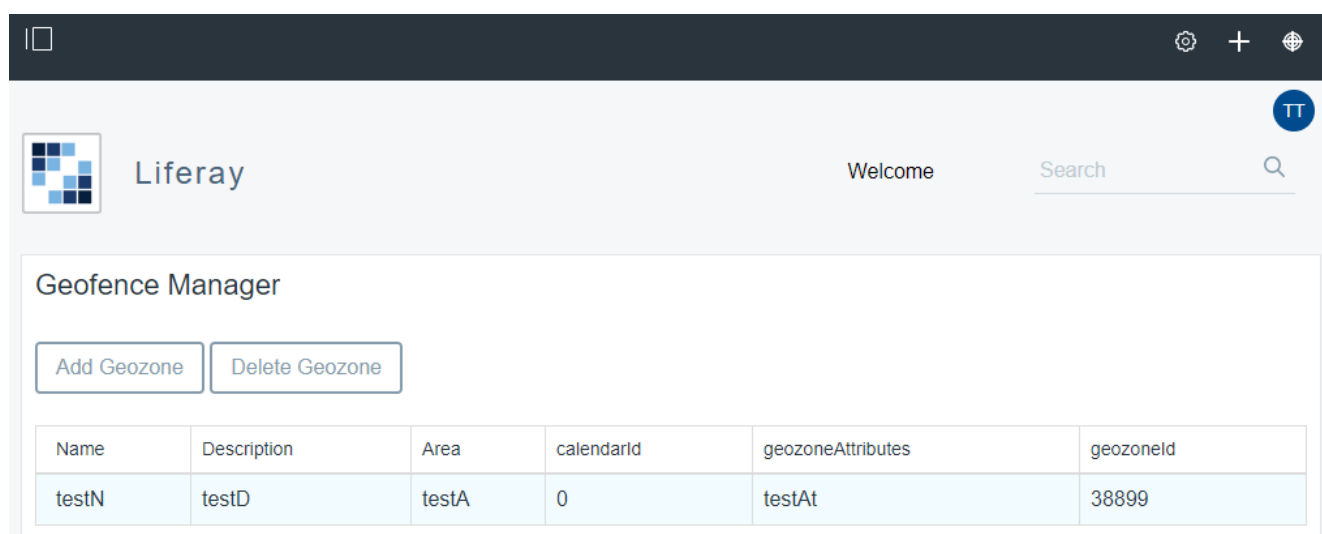


Рисунок 16 – Обновленный вид web-страницы портлета с добавленной записью в базу данных

4.4.2 Проверка уникальности первичного ключа geozoneId

Для проверки уникальности генерируемого ключа создадим абсолютно такую же строку данных, как и в п.4.4.1. Ключ должен быть уникальным, несмотря на полное совпадение всех полей.

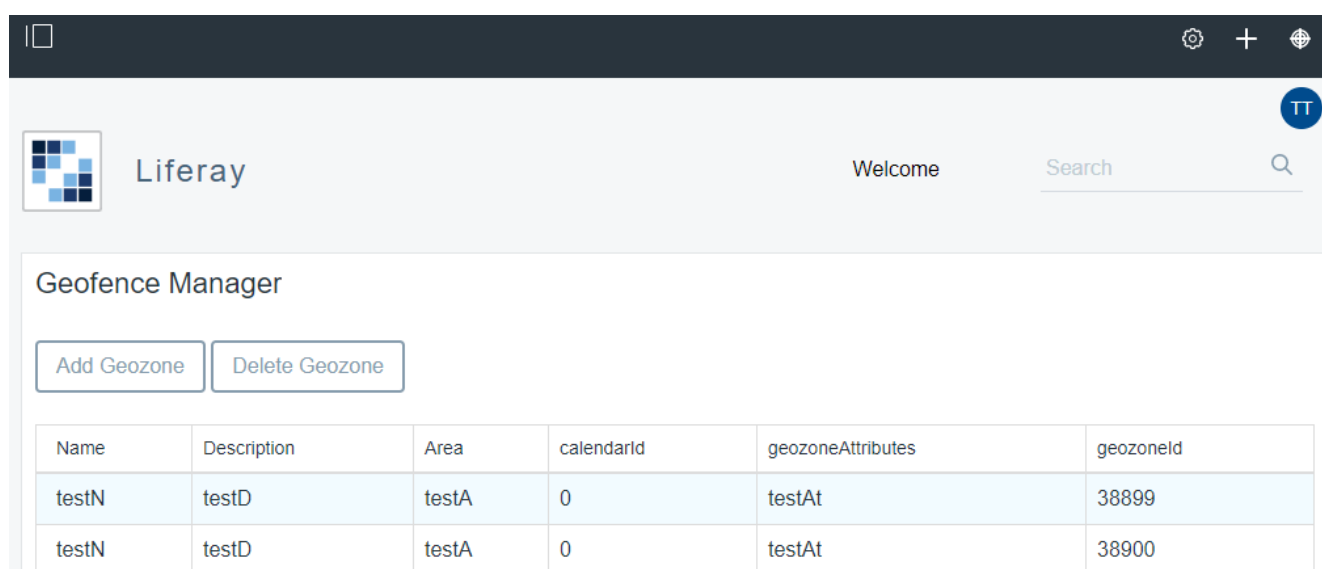


Рисунок 17 - Обновленный вид web-страницы портлета с добавлением строки идентичной первой записи в базу данных

Как видно из рисунка 17, данные добавились с новым идентификационным ключом, значит уникальный ключ генерируется корректно.

4.4.3 Корректный ввод удаляемых данных и удаление геозоны

При нажатии на кнопку “Delete Geozone” должна открываться новая web-страница с единственным полем ввода данных – уникальным ключом `geozoneId`. Так как реализованный метод удаления `deleteGeozone(long geozoneId)` является стандартным сгенерированным методом класса `GeozoneLocalServiceImpl`, то удаление записи должно производиться по уникальному ключу записи. Проверим это, введя в поле `geozoneId` номер 38900, соответствующий последней записи.

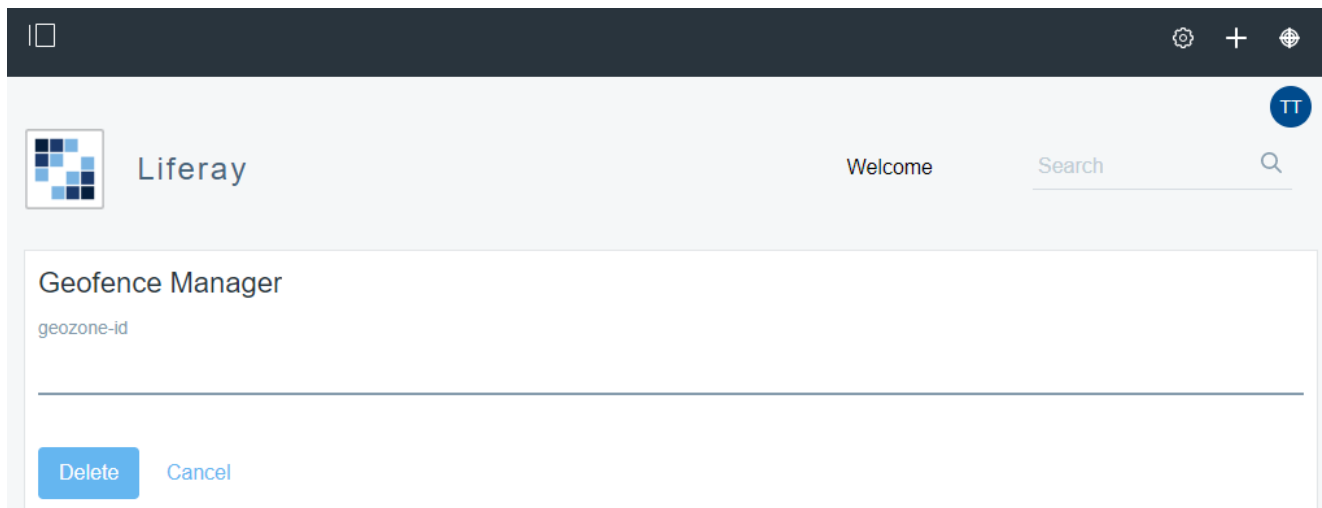


Рисунок 18 – Корректное открытие web-страницы с удалением после нажатия кнопки “Delete Geozone”

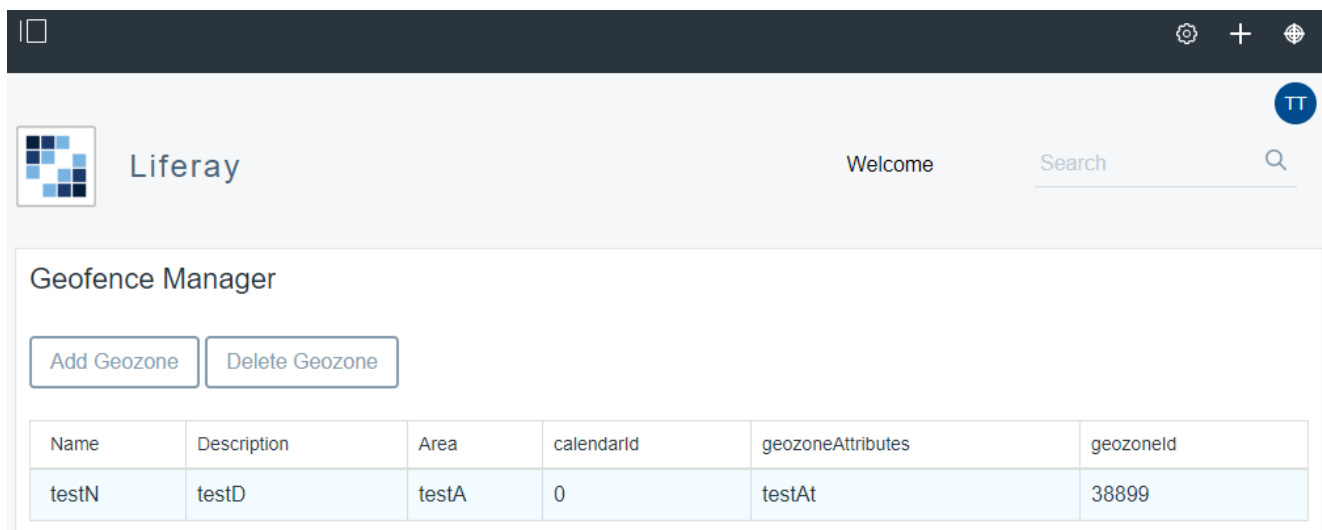


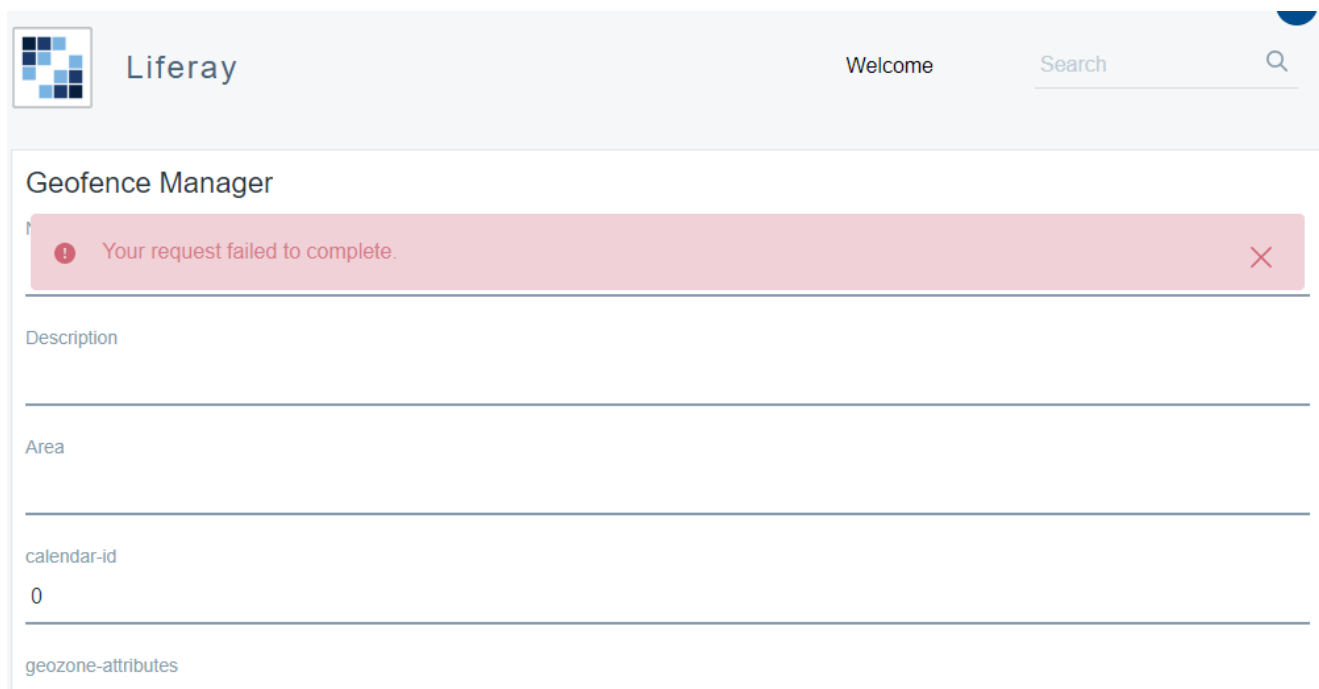
Рисунок 20 – Корректное удаление записи с уникальным ключом `geozoneId` равной 38900

Как видно из рисунка 19, web страница с полем ввода уникального идентификационного номера открылась корректно, а после ввода данных и

нажатии кнопки “Delete” произведено удаление записи с уникальным ключом geozoneId равным 38900. Таким образом, функция удаления геозоны работает.

4.4.4 Ввод несуществующего идентификатора геозоны при удалении

При заполнении структуры рисунка 18 несуществующим идентификационным номером пользователь должен увидеть соответствующее сообщение и должна отобразиться web-страница добавления новой геозоны. Для проверки данного условия введем уже удаленный номер 38900 в поле geozoneId.



The screenshot shows the Liferay user interface. At the top, there is a header with the Liferay logo, the word "Welcome", and a search bar. Below the header, the main content area is titled "Geofence Manager". A red error message banner is displayed, stating "Your request failed to complete." with a close button (X). Below the error message, there are several input fields: "Description", "Area", "calendar-id" (with the value "0"), and "geozone-attributes".

Рисунок 21 – удаление несуществующей записи

Как видно из рисунка 21, пользователь видит ошибку о том, что его запрос не может быть распознан (введены не верные данные) и пользователь видит перед собой web-страницу добавления новой записи. Таким образом, данный функционал работает корректно.

4.4.5 Некорректный ввод идентификатора geozoneId

Как и в п.4.4.4 попробуем ввести неправильное значение идентификатора geozoneId. На этот раз попробуем ввести туда строковое значение, которое будет сгенерировано случайным образом нажатиями любых клавиш на клавиатуре – как буквенных, так и числовых. При обработке этих данных должна возникнуть ситуация абсолютно похожая на ситуацию п.4.4.4 – ошибка о некорректности данных и перевод на web-страницу с добавлением.

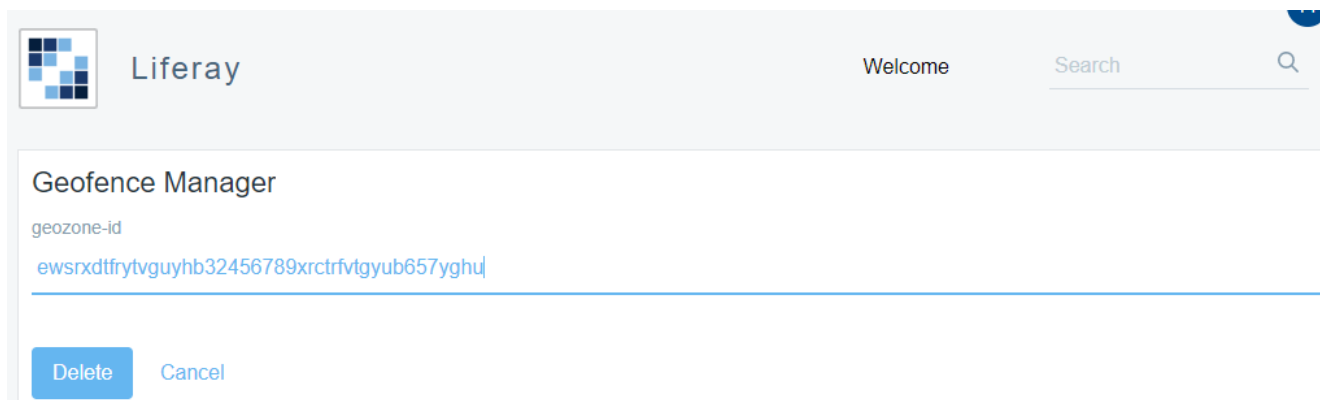


Рисунок 22 – В поле geozoneId введены некорректный набор символов

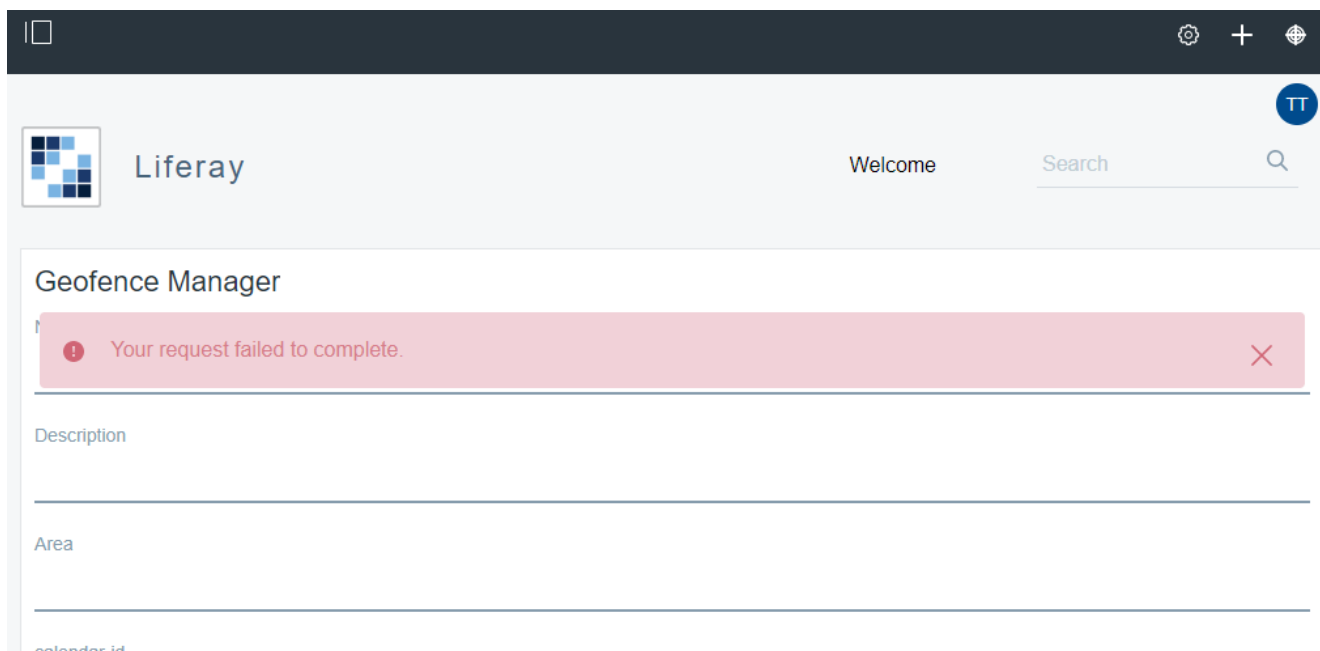
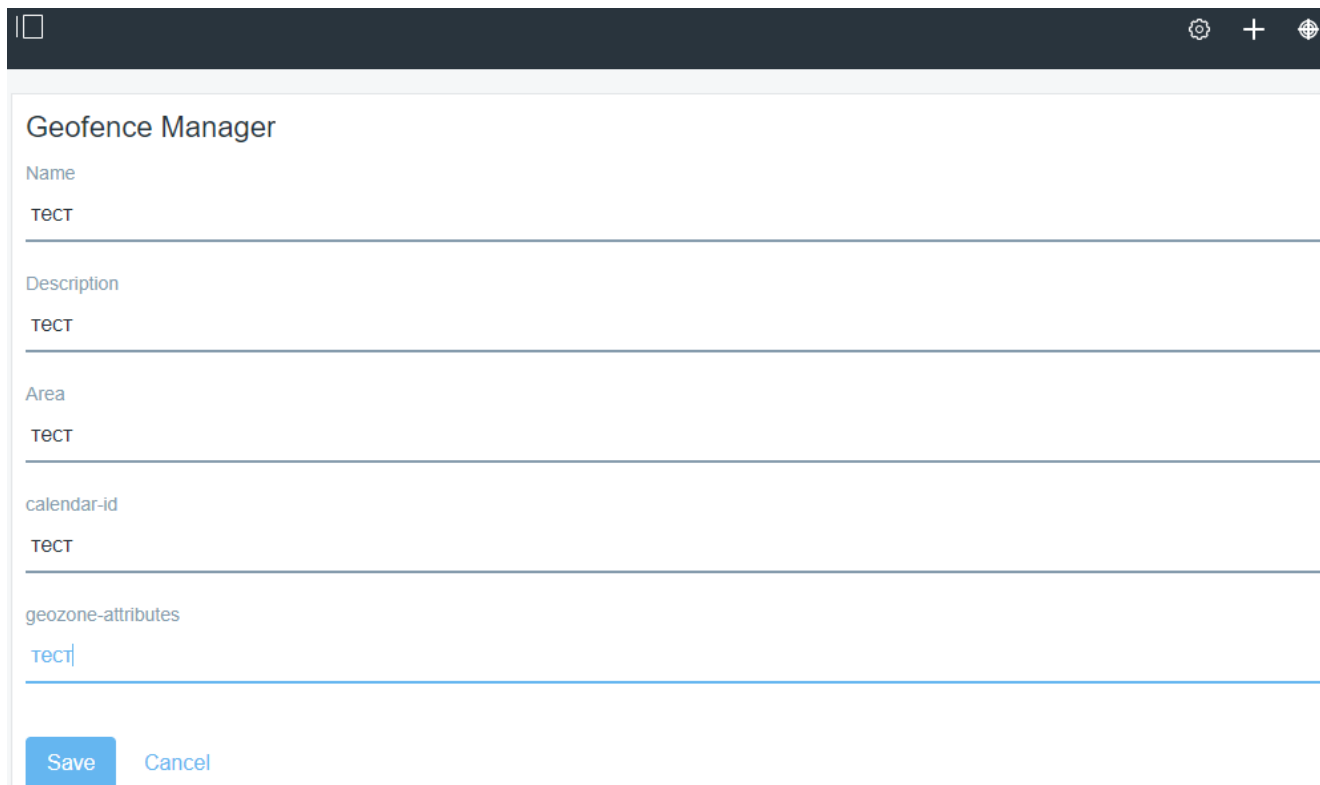


Рисунок 23 – Реакция системы на ввод некорректных данных

Как видно на рисунке 22, данные введены не корректно и система отреагировала на них с помощью ошибки и перевела пользователя на web-страницу с добавлением новой записи (рис. 23). Таким образом, данный функционал работает корректно.

4.4.6 Некорректный ввод в поля с числовым значением

Для проверки данного функционала в поле calendarId (единственное изменяемое поле с числовым значением) строковые данные. Система должна записать новую строку и в поле calendarId поставить значение по умолчанию – значение 0 (ноль).



Geofence Manager

Name
тест

Description
тест

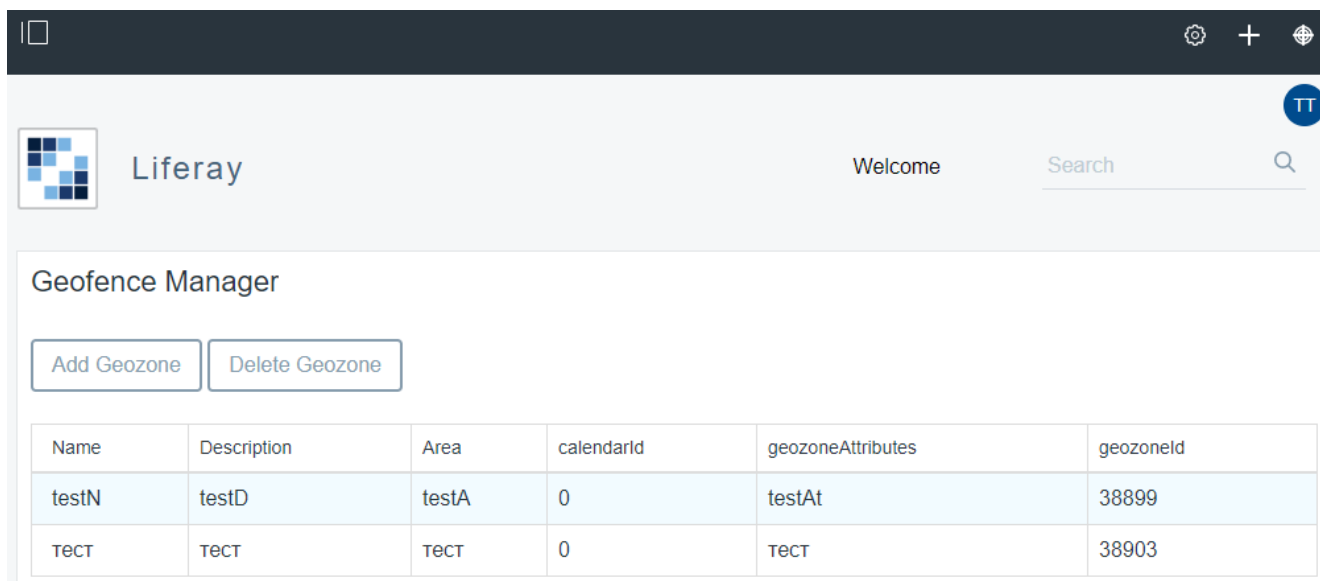
Area
тест

calendar-id
тест

geozone-attributes
тест

Save Cancel

Рисунок 24 – Вводимые значения теста (поле calendarId имеет значение “тест”)



Geofence Manager

Add Geozone Delete Geozone

Name	Description	Area	calendarId	geozoneAttributes	geozoneId
testN	testD	testA	0	testAt	38899
тест	тест	тест	0	тест	38903

Рисунок 25 – Результат теста.

Как видно из рисунка 25, данные введенные на рисунке 24 были приняты системой, а значение поля calendarId заменено на значение по умолчанию. Чтобы проверить, введем такие же данные, но в поле calendarId поставим значение 1 (единица). Система должна добавить идентичное поле, но со значением поля calendarId равным единице, потому что значение изменяемого поля было задано корректно – в форме числа.

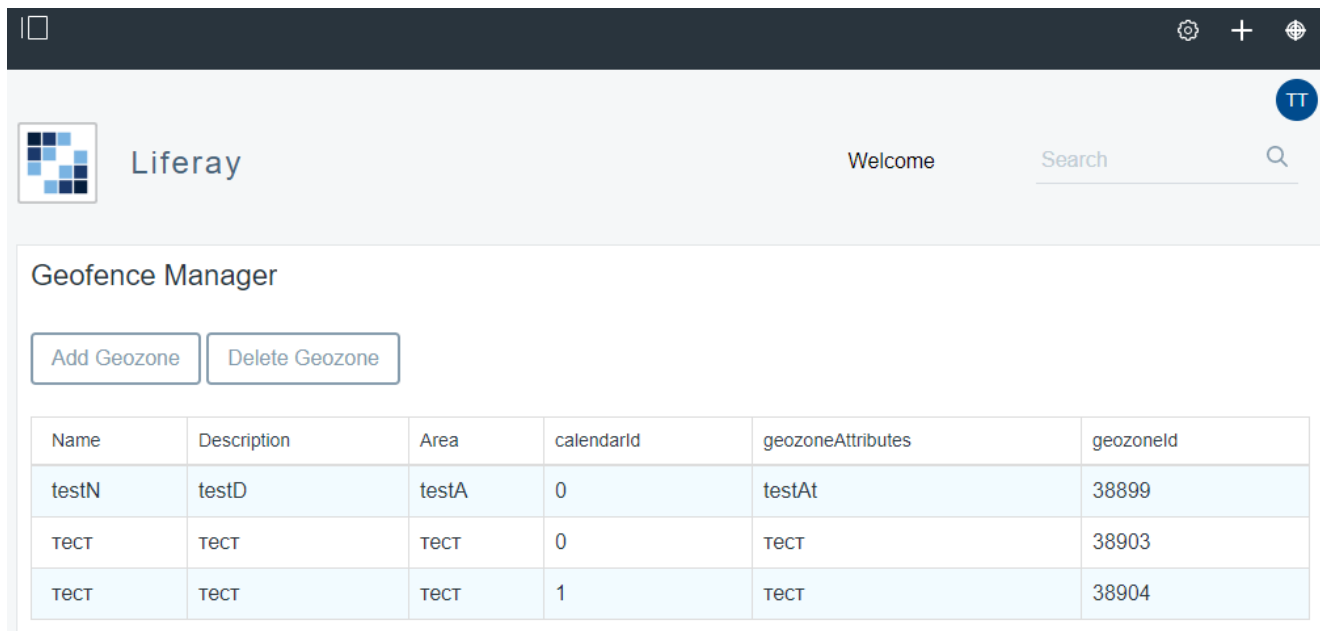


Рисунок 26 – Изменение поля calendarId путем ввода корректных (числовых) данных

Как видно из рисунка 26, была добавлена строка с такими же параметрами, как и на рисунке 25, за исключением поля `calendarId`, которое было изменено вводом корректным (числовым) значением. Таким образом, данный функционал работает корректно.

5 Выводы

В ходе данной курсовой работы был:

- изучен GeofenceManager и его графический интерфейс в Traccar;
- спроектирован интерфейс компонента;
- реализовано хранение данных в БД;
- изучен метод ручного тестирования;
- реализована визуализация данных в GUI;
- изучена обработка событий GUI и отправка команд;
- использование CSS стилей и шаблонов.

Также в ходе данной работы удалось поработать с реальными проектами, такими как Traccar, Liferay. Удалось создать свой ServiceBuilder и поработать с OSGi.

Результатом выполнения данной курсовой работы является написанный на языке Java портлет, который запускается в системе Liferay и реализует менеджер Geofence приложения Traccar по GPS трекингу.

Список литературы

- [1] Документация по Traccar: [Электронный ресурс]. Режим доступа: <https://www.traccar.org/documentation/>.
- [2] Документация по LifeRay: [Электронный ресурс]. Режим доступа: <https://dev.liferay.com/develop/tutorials/>.
- [3] GitHub – GroupsManager [Электронный ресурс]. Режим доступа: https://github.com/vldrshv/RPO_course_work
- [4] Liferay 7 Features [Электронный ресурс]. Режим доступа: <http://www.liferay savvy.com/2016/03/liferay-7-features.html>
- [5] Traccar API-reference [Электронный ресурс]. Режим доступа: <https://www.traccar.org/api-reference/>