Инженерный вестник Дона, №4 (2018)
ivdon.ru/ru/magazine/archive/n4y2018/5436

Веб - сервис для интеграции в урбанистическую среду

О.Д. Глод, В.В. Сетраков

Южный федеральный университет, Ростов-на-Дону

Аннотация: Для упрощения процесса интеграции пользователя в новую для него урбанистическую среду оптимально использовать веб-сервис, позволяющий на основе местоположения, определить ближайшие точки, носящие какой-либо информационный характер, или же сгруппированные списки, связанных между собой точек, образующих маршруты. Веб-сервис соответствует REST (Representational State Transfer) архитектурному стилю и разработан на платформе языка программирования Java 8. В качестве методологии проектирования использован ООАП (объектно-ориентированный анализ и проектирование).
Ключевые слова: веб-сервис, веб-приложение, пользователь, туризм, туристический маршрут, достопримечательности, координаты, локация.

В последнее время сфера туризма активно развивается. Требуются решения множества сопутствующих задач, например развитие транспортной системы, развитие системы общественного питания, улучшение имиджа, развитие информационной инфраструктуры. Необходимы инструменты, упрощающие процесс интеграции пользователя в новую для него урбанистическую среду. Для написания этих средств требуется централизованный источник данных с легким и доступным API (Application Programming Interface) [1], который позволяет свободно интегрироваться с различными сервисами и платформами. Разрабатываемый веб-сервис предоставляет API для доступа к информации по различным достопримечательностям основываясь на переданных данных.

В данный момент практически нет централизованных информационных ресурсов, предоставляющих возможность постройки маршрутов на основе различных карточных сервисов.

Конечным продуктом будет являться веб-сервис, соответствующий REST (Representational State Transfer) архитектурному стилю, предоставляющий следующую функциональную базу:

1. Возможность получения списка достопримечательностей на основе местоположения пользователя;
2. Возможность получения заранее предопределенных списков достопримечательностей, организованных в маршруты на основе местоположения пользователя;
3. Возможность получения списка достопримечательностей, имеющихся на конкретной улице;
4. Возможность получения списка достопримечательностей, имеющихся в конкретном районе;
5. Возможность получения списка достопримечательностей, имеющихся в конкретном городе;
6. Возможность получения списка достопримечательностей, имеющихся в конкретной стране;
7. Возможность получения маршрутов, имеющихся на конкретной улице;
8. Возможность получения маршрутов, имеющихся в конкретной районе;
9. Возможность получения маршрутов, имеющихся в конкретной городе;
10. Возможность получения маршрутов, имеющихся в конкретной стране;
11. Возможность создания маршрута;
12. Возможность создания достопримечательности.

Обязательные свойства веб-сервиса:

1. Возможность интеграции с любыми электронными картами;
2. Легкость в эксплуатации;
3. Результатами запросов являются данные в формате Json (JavaScript Object Notation).

В качестве ответа на запрос к веб-сервису, клиенту должен быть возвращен Json объект, представляющий запрошенный объекты, если же объектов, удовлетворяющих критериям поиска, не найдено, то необходимо вернуть пустой Json объект.

В качестве аналогичных веб-сервисов, схожих по функционалу можно привести веб-сервис компании Google — Place Details. Данный сервис предоставляет более обширную информацию по определенному месту, в том числе пользовательский рейтинг и обзоры.

Google накладывает ограничения по количеству запросов при бесплатном использовании своих сервисов. При бесплатном использовании сервисов вам будут доступны только 1000 запросов сутки. Этот порог можно увеличить, предоставив свои личные данные и платежную карту, в таком случае количество запросов увеличится до 150 000 в сутки. Также есть возможность купить лицензию [2].

Для разработки веб-сервиса необходимо определиться с такими инструментами как:

1. Язык программирования
2. База данных
3. Сервер приложений
4. Интегрированная среда разработки
5. Средство сборки проекта
6. Библиотеки для разработки

Так как разработка веб-сервиса не подразумевает разработки графического интерфейса, то здесь рассмотрим лишь выбор инструментария для разработки серверной части.

В качестве языка программирования для написания информационной системы выбран язык программирования Java версии 8.

Обоснования для выбора данного языка:

1. Независимость от платформы, на которой будет запущено приложение, написанное на языке Java.

Так как программы, написанные на данном языке, вначале компилируются в байт-код, а после запускаются на выполнение виртуальной java машиной, то эти приложения могут работать независимо от системы. Виртуальная java машина представляет из себя интерпретатор байт-кода, то есть способна запускать скомпилированные java программы

2. Объектно-ориентированный подход.

Язык программирования Java полностью соответствует объектно-ориентированному подходу. Данная методология представляет программу, как совокупность объектов, где каждому объекту отведена своя функциональность и свои свойства. Наследование позволяет решить проблему дублирования кода, вынося все общие элементы в родительский класс. Полиморфизм позволяет решить проблему замены компонентов, выделив весь необходимый функционал в отдельный интерфейс. Инкапсуляция позволяет по возможности скрыть реализацию различных методов, а также различные свойства, с которыми не должен оперировать пользователь данного класса.

3. Современность.

Хотя язык Java был создан в 1995 году, его продолжают развивать в соответствии с современными потребностями, методологиями и подходами к программированию.

4. Сильная статическая типизация.

Сильная статическая типизация позволяет обезопасить свой код от ошибок во время выполнения из-за несоответствия типов, так как уже на стадии компиляции известен тип каждого объекта. Также это оптимизирует работу программы, так как нет частой необходимости проверять тип объекта.

5. Богатый API.

Java EE (Enterprise Edition) в своем комплекте разработки SDK (Software Development Kit) имеет множество полезных библиотек, реализующих тот или иной необходимый функционал, в том числе для создания и работы с веб-сервисами, и взаимодействия с базами данных при помощи технологии программирования ORM (Object-Relational Mapping) [3].

В качестве среды разработки выбран NetBeans. NetBeans – это полностью бесплатная среда разработки со множеством встроенных и внешних плагинов. Она предоставляет удобный инструментарий для разработки на различных языках программирования и создание проектов на основе различных шаблонов [4].

В качестве сервера приложений выбран JBoss Enterprise Application Platform 7. Это обусловлено возможностью масштабирования до более миллиона подключений; поддержкой последних стандартов и технологий, в частности, поддержкой новейших стандартов доступа к данным на основе REST, включая сервисы JAX-RS 2 и JSON-P; модульность.

В качестве базы данных выбран SQL (structured query language) Server, так как SQL Server обладает всеми качествами, необходимыми для реализации ключевых требований к СУБД, а именно – производительностью, стабильностью и возможностью масштабирования; имеет бесплатный выпуск - SQL Server Express для разработчиков; гарантирует повышенную безопасность и производительность средств разработки; доступна на практически на любых операционных системах [5]. Основной используемый язык запросов — Transact-SQL, является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Используемые библиотеки – Hibernate и RestEasy. Hibernate - это самая популярная реализация спецификации JPA (Java Persistence API), предназначенная для решения задач объектно-реляционного отображения. JPA - спецификация API_Java EE, предоставляет удобный API для работы с реляционными базами данных в контексте объектно-ориентированного подхода [6,7]. RestEasy - это проект JBoss, являющийся фреймворком для разработки RESTful Web Service и RESTful Java application. Это полностью сертифицированная и портативная реализация JAX-RS 2.1 спецификации, которая предоставляет API для RESTful Web Services по HTTP протоколу [8,9].

Рассмотрим функции предлагаемого приложения на примере получения списка достопримечательностей на основе местоположения пользователя.

Данная функция позволяет на основе переданных координат, которые определяют местоположение пользователя вернуть список достопримечательностей. Параметрами для фильтрации списка мест являются долгота, широта, где долгота и широта определяют улицу поиска. В ответ на данный запрос пользователь получит список достопримечательностей, определенных параметрами запроса. Результат будет представлен в формате JSON. Алгоритм выполнения данной функции заключается в том, что пользователь делает GET запрос на веб-сервер, передавая такие параметры, как долгота и широта. Далее, система, приняв запрос на уровне представления, передаст параметры этого запроса уровню бизнес-логики. На уровне бизнес-логики система запросит у уровня источника данных список доступных стран, получив данный список, система определит вначале город, в котором находится пользователь, далее район и после улицу, после чего, запросив список достопримечательностей на этой

улице, передаст полученные данные на презентационный уровень, откуда они будут конвертирован в JSON формат и переданы пользователю.

Для доступа к функциональной базе веб-сервиса могут использоваться различные клиентские программы, функционирование веб-сервиса не зависит от них. Пользователь формирует запрос к веб-сервису на получение необходимых данных с помощью адреса URL (Uniform Resource Locator), при необходимости, указывая параметры. Клиентская программа устанавливает соединение с требуемым веб-сервисом, используя HTTP протокол передачи данных. Веб-сервер получает запрос, перенаправляет его серверу приложений, который, в свою очередь, запускает на выполнение Java приложение в коде которого идет обращение к базе данных для получения запрошенной информации, после чего, на основе полученных данных, формируется Json объект, который передается клиенту.

В силу используемого объектно-ориентированного подхода к написанию приложения, при конструировании моделей данных применялся также объектно-ориентированный подход, что подразумевает использование наследования и агрегации сложных типов данных объектами.

При конструировании модели общие свойства выносились в родительские класс для избегания ненужного дублирования кода и логики. Каждый из уровней иерархии содержит только свои атрибуты, которые, при необходимости, наследуются другими классами, расширяя их функциональность и информативность. Данный подход позволяет не размазывать схему данных, определяя конкретно одно место для каждого свойства, не дублировать одни и те же атрибуты в разных сущностях, позволяет использовать полиморфизм, что дает нам возможность манипулировать по факту различными сущностями через их общие родительские классы.

Отражение объектного способа представления данных на реляционные базы данных берет на себя библиотека Hibernate. На основе определенных аннотаций, применяемых к классам и их атрибутам, данная библиотека строит таблицы, отражающие все необходимые сущности, со всеми связями и ограничениями.

У нас имеется основной тип для наследования всеми объектами, которые будут представлены в базе данных, этот тип содержит атрибут, который будет представлять первичный ключ, это избавляет нас от необходимости определять данное свойство в каждом класс, который мы будем хранить в базе данных. Также, в контексте тематики приложения, мы будем оперировать именованными объектами, поэтому создадим отдельный родительский класс, который будет содержать в себе информацию о имени объекта, по той же логике создадим классы, отвечающие за представление объектов, которые могут содержать контент, местоположение, границы. Так как контент у нас может быть разный, создадим класс, который будет обобщать различные типы контента, данный класс будет содержать ссылку на своего владельца, владельцем может выступать любой класс, который наследует тип владельца контента, это точки маршрутов, маршруты, страны, города, районы, достопримечательности. Также нам потребуется объект представляющий адрес, данный объект будет содержать ссылки на объекты страны, города, района, улицы, достопримечательности, данный объект нужен для связи объектов точек маршрута с достопримечательностями. Также будут созданы объекты, олицетворяющие точки маршрута и маршруты. Объекты маршрутов будут содержать списки точек маршрута, который их составляет, точки маршрута будут содержать адрес, связывающий их с достопримечательностью на которую указывает адрес. Данная декомпозиция сущностей позволяет более гибко взаимодействовать с ними. Например, так как объекты страны содержат список городов, города —

список районов, районы — список улиц, а улицы — список достопримечательностей, то выделение связи точек маршрутов с локациями через отдельный объект адреса, позволяет нам не содержать отчасти дублированные ссылки на одни и те же точки, также позволяя простой поиск точек по локациям.

Поток данных определяет информацию, передаваемую через некоторое соединение от источника к приемнику. Схема поток данных веб-сервиса, в случае запроса от пользователя на получение каких-либо данных, состоит из приемника, которым выступает пользователь, и источника, которым выступает сам веб-сервис, соединением является канал передачи данных. В случае, когда пользователь хочет что-то разместить на веб-сервисе, схема является обратной, то есть пользователь выступает источником данных, а веб-сервис является приемником, соединение не менялось при этом.

Потоки данных непосредственно внутри приложения представляют из себя передачу данных с уровня на уровень. Вначале данные для запроса передаются от пользователя на уровень веб-интерфейса, после чего система передает их на уровень бизнес логики, где формируется запрос к базе данных на основе переданных параметров от пользователя, далее данные с запросом передаются на уровень источника данных, где выполняется запрос на выборку нужной информации, после чего, выбранная информация поднимается вверх по уровням и передается обратно пользователю, который их запросил.

Таким образом, веб-сервис упрощает создание инструментов для интеграции пользователей в новую для них урбанистическую среду, посредствам предоставления общей базы с данными и простого интерфейса, для взаимодействия с сервисом и независимости от способов представления

результатов запросов. Это позволяет, при необходимости, интегрировать данный веб-сервис с другими [10].

Литература

1. Янишевская А.Г., Чурсин М.А. Использование сторонних интерфейсов программирования приложений на примере интерфейсов прикладного программирования социальных сетей Facebook и Twitter // Инженерный вестник Дона, 2015, №2 (часть 2) URL: ivdon.ru/ru/magazine/archive/n2p2y2015/2978.

2. Google Developers URL: developers.google.com.

3. Oracle URL: oracle.com/javase/8.

4. NetBeans URL: netbeans.org.

5. Microsoft URL: microsoft.com/ru-ru/sql-server/sql-server-2016.

6. WildFly URL: wildfly.org.

7. Hibernate URL: hibernate.org.

8. RestEasy URL: resteasy.github.io.

9. Виды диаграмм UML // Интуит URL: intuit.ru/studies/courses/1007/229/lecture/5954.

10. Файзрахманов Р.Р. Новый подход к обеспечению веб-доступности для незрячих пользователей на основе улучшения навигационных характеристик веб-страниц // Инженерный вестник Дона, 2012, №4 (часть 2) URL: ivdon.ru/ru/magazine/archive/n4p2y2012/1442.

References

1. Yanishevskaya A.G. Inženernyj vestnik Dona (Rus), 2015, №2 (part 2) URL: ivdon.ru/ru/magazine/archive/n2p2y2015/2978.

2. Google Developers URL: developers.google.com

3. Oracle URL: docs.oracle.com/javase/8.

4. NetBeans URL: netbeans.org.

5. Microsoft URL: microsoft.com/ru-ru/sql-server/sql-server-2016.

6. WildFly URL: wildfly.org.

7. Hibernate URL: hibernate.org.

8. RestEasy URL: resteasy.github.io.

9. Intuit URL: intuit.ru/studies/courses/1007/229/lecture/5954.

10. Faizrahmanov R.R. Inženernyj vestnik Dona (Rus), 2012, №4 (part 2) URL: ivdon.ru/ru/magazine/archive/n4p2y2012/1442.