В итоге было выделено 10 зон, в которые вошли: шахматная аллея, детская площадка, аттракционы, веревочный городок, дома ремесел, зона общественного питания, концертная площадка, летний амфитеатр, спортивные площадки, дома отдыха.

3. Детальная прорисовка каждой зоны. Нанесение на план отдельных элементов инфраструктуры с указанием их предполагаемых масштабных размеров.

4. Создание итогового альбома «Дизайн-концепция развития мультифункционального парка «Мазурино».

По итогам проделанной работы был создан альбом «Дизайн-концепция развития мультифункционального парка «Мазурино» и передан в качестве идеи преобразования парка главе Витебского областного исполнительного комитета.

В альбоме представлена схема-план развития парка, который отвечает всем современным требованиям и представляет собой мультифункциональное место в общем городском ландшафте г. Витебска, где жители могут отдохнуть от ежедневной суеты, заняться активными видами спорта и с пользой провести время. Занятие для себя здесь сможет найти каждый, т. к. дизайн-концепция предусматривает активное вовлечение всех возрастных групп населения.

Список использованных источников

1. Булгакова, К. Каким должен быть современный городской парк [Электронный ресурс]. Режим доступа: https://j.etagi.com/stati/interesnoe/modern_park/. – Дата доступа: 27.10.2020.

2. Гулина, М. Главное место в городе [Электронныйресурс]. – Режим доступа: https://34travel.me/post/urbanparks. – Дата доступа: 27.10.2020.

3. Комплекс культурно-массового отдыха имени советской армии [Электронный ресурс]. – Режим доступа: http://www.gck.by/2009/10/1038. – Дата доступа: 27.10.2020.

УДК 004.413

ВЗАИМОДЕЙСТВИЕ С ВНЕШНИМИ API НА ПРИМЕРЕ ПРИЛОЖЕНИЯ ВИРТУАЛЬНОГО РОБОТА ДЛЯ СЕРВИСА TELEGRAM

Давидовская М. И., Шиловский С. В.
Белорусский государственный университет
e-mail: fpm.shilovski@bsu.by

Summary. *The use of external APIs is studied by the example of the implementation of a virtual robot for the Telegram service. The example shows the integration of external APIs into application and connection to it, as well as simplicity and ease of use.*

Реализация архитектуры проектов зависит от многих факторов, как внешних, так и внутренних [1, с. 64]. При разработке надо учитывать результирующую эффективность: масштабируемость, тестирование, гибкость и другие параметры. Каждый проект уникален и реализуется на основе определенных требований. Например, это может быть большой монолитный проект, структуру и сервисы которого сложно исследовать из-за объема кодовой базы и непрозрачной архитектуры, либо это может быть проект, разбитый на множество небольших микросервисов, которые могут взаимодействовать друг с другом.

Для поставленной задачи рассмотрим микросервисную архитектуру. Основной целью данной архитектуры является декомпозиция задач [1, с. 63]. В зависимости от проекта и его реализации идею микросервисной архитектуры можно интерпретировать по-разному. Например, можно выделить и вынести отдельно различные сервисы, которые выполняют определенные задачи. Как следствие, получаем взаимодействие с каждой отдельно вынесенной задачей, не меняя других функциональных участков нашего проекта. Для взаимодействия с различными сервисами, не обязательно реализованными нами, существуют технологии для связи с

ними. Одной из основных является интерфейс прикладного программирования (Application Programming Interface – API).

В общем случае API – это интерфейс, написанный разработчиками, чтобы их продукт могли использовать сторонние разработчики. REST API – технология взаимодействия через HTTP-запросы [1, с. 301]. На сервер отправляется запрос определенного типа, после этого запрос обрабатывается на его стороне, а затем сервер формирует и возвращает ответ на запрос. Применяются следующие запросы для взаимодействия сервисов: создание (POST – запрос), чтение (GET – запрос), обновление (PUT и PATCH – запросы), удаление (DELETE – запрос).

Микросервисная архитектура в сочетании с REST API позволяет разделить одно большое монолитное приложение на множество сервисов, которые могут общаться между собой, используя API запросы. Существуют другие технологии взаимодействия, но в контексте данной статьи не рассматриваются. Широко распространенные программные системы предоставляют общедоступные API, чтобы разработчики могли внедрять в свои проекты уже готовый функционал и использовать его. Любое качественное приложение сопровождается детальной документацией. Данное утверждение действительно и для интерфейса API. Как для внешнего, так и внутреннего API формируется документация с детализацией запросов, ответов и их примерами.

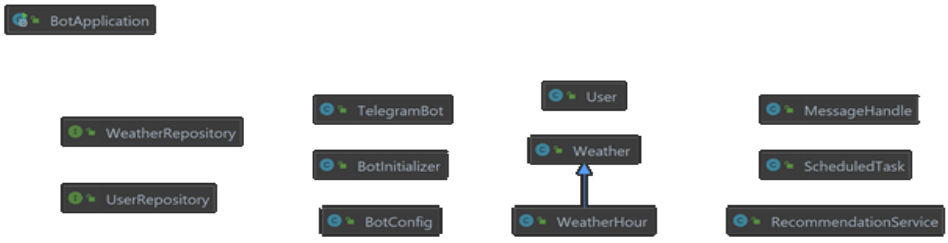

Рисунок 1 – Структура проекта

Для практического рассмотрения использования внешних API был разработан виртуальный робот для сервиса Телеграмм. Основной его функционал – получение информации о погоде в определенной точке мира с использованием внешних сервисов. Для разработки использовалась среда разработки программного обеспечения IntelliJ IDEA, сборщик проекта Maven, интерфейс Telegram Bot API [2], WeatherAPI[3], библиотека Hibernate JPA для работы с базами данных, база данных PostgreSQL, библиотека Lombok для упрощенного написания классов. Структура проекта представлена в виде UML диаграммы (см. рис. 1).

Первый этап разработки – создание Maven проекта, добавление необходимых зависимостей используемых библиотек в конфигурационном файле и подключение к базе данных. Второй этап разработки – реализация класса для подключения виртуального робота. Третий этап разработки – создание классов-сущностей, а именно образ пользователя и образ погоды, описание полей для хранения в базе данных (см. рис. 2) и настройка библиотеки Hibernate JPA. Далее разрабатывалась система уведомлений пользователей о предстоящих погодных условиях.

```
@Data
@Entity
@Table(name = "BotUsers")
public class User {
    @Id
    String Id;

    private String name;

    @ManyToOne
    private Weather weather;

    @Column(columnDefinition = "boolean default false")
    private boolean weather_subscription;
}
```

Рисунок 2 – Класс пользователя

Четвертый этап разработки – интеграция Weather API[2] в приложение. Пользователь приложения отправляет сформированный HTTP запрос (см. рис. 3) к сервису прогноза погоды. Далее используемый сервис отправляет объект согласно запросу в формате JSON (см. рис. 4). Объект обрабатывается и предоставляется пользователю в удобном для восприятия виде.

```
String urlReq = new String( original: "http://api.weatherapi.com/v1/forecast.json?key="
+ key + "&q="+city+"&days=1&aqi=yes&alerts=no&lang=ru");
```

Рисунок 3 – Формирование HTTP запроса

```
{
    "id": 1,
    "name": "Vladimir",
    "age": 18,
}
```

Рисунок 4 – Пример объект JSON

Последний этап разработки – развертывание виртуального робота на облачной платформе Heroku, которая основана на управляемой контейнерной системе. Heroku предоставляет определенные функции для публикации и размещения приложений. Также платформа обеспечивает отдельные сервисы для интеграции и использовании в проектах, например управляемая база данных как услуга Heroku Postgres, которая была использована при реализации виртуального робота.

Таким образом, результатом исследования является готовое приложение с микросервисной архитектурой, интегрирующее внешние приложения по API и опубликованное на облачном хостинге Heroku. Идея разделения приложения на сервисы позволяет оптимизировать его за счет связывания сервисов и их изоляции друг от друга, что помогает сделать приложение более безопасным.

Список использованных источников и литературы

1. Ричардсон, К. Микросервисы. Паттерны разработки и рефакторинга.: учебн. пособие / К. Ричардсон – СПб: Питер, 2019.

2. Telegram Bot API: электронн. ресурс. URL: https://core.telegram.org/bots/api

3.Weather API Doc: электронн. ресурс. URL: https://www.weatherapi.com/docs

УДК 339

АДАПТАЦИЯ ЗАПАДНЫХ КОМПАНИЙ В КИТАЕ

Джумалиева А. Д.
Полесский государственный университет
e-mail: annadzumalieva@gmail.com

Summary. *In the context of large-scale internationalization, the advertising message is borrowed from a different sociocultural environment. In this case its perception is complicated due to the national and cultural peculiarities of non-verbal information by people from different countries. For this reason, there is a need to adapt the company's products to the peculiarities of consumers of foreign-language advertising.*

Китай занимает второе место по объемам мирового импорта, что делает его внутренний рынок крайне привлекательным для экспортеров. Однако завоевать сердце китайского потребителя не так уж и просто. На примерах трех компаний (IKEA, KFC и Carrefour) мы попытаемся понять, как важны знания культурных особенностях в маркетинговой сфере.