

- geo-zones. Данной функцией сервиса PushWoosh можно воспользоваться на устройствах под управлением операционной системы IOS, Android или Windows Phone. Она позволяет создавать области на карте, с точностью до 50 метров, чтобы отсылать уведомления пользователям в конкретно заданном месте;
- timezone Sensitive. Сервис PushWoosh предоставляет возможность отсылать уведомления пользователю в соответствии с его часовым поясом. Это позволяет сделать ваши push-сообщения своевременными для каждого конкретного пользователя и, как результат, более привлекательными [2].

Сервис PushWoosh отлично себя зарекомендовал и является лучшим решением для работы с push-уведомлениями. По этой причине, при создании мобильного приложения «Музыканты России», было принято решение об использовании именно данного сервиса. Далее будет рассмотрен пример использования данного сервиса в мобильном приложении, написанном на языке программирования Java. Процесс интеграции данного сервиса достаточно прост, на официальном сайте можно найти подробную инструкцию о том, как использовать PushWoosh на любой платформе. Важную роль в приложении «Музыканты России» играют Push-уведомления. Ведь именно они позволяют вовремя узнать о скидках и акциях в музыкальных магазинах, об изменениях в расписании посещения репетиционных баз и музыкальных лейблов, о приближающихся важных событиях в мире музыки. В основе работы системы уведомлений лежит сервис PushWoosh. Далее будет приведен пример, как именно можно использовать данный сервис в своем приложении.

В первую очередь необходимо наследовать MainActivity от FragmentActivity class и реализовать интерфейс PushEventListener (см. листинг 1).

Листинг 1 – Реализация интерфейса PushEventListener

```
public class MainActivity extends FragmentActivity implements PushEventListener
```

В методе onCreate вызываем метод PushFragment.init(this); (см. листинг 2).

Листинг 2 – Инициализация PushFragment

```
@Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    PushFragment.init(this);
}
```

Далее создаем метод onNewIntent и вызываем функцию PushFragment.onNewIntent (this, intent); (см. листинг 3).

Листинг 3. – Создание метода onNewIntent

```
@Override public void onNewIntent(Intent intent) {
    super.onNewIntent(intent); // check if we have a new intent with push notification
    PushFragment.onNewIntent(this, intent);
}
```

После этого необходимо реализовать методы PushEventListener интерфейса. Самым важным для нас будет метод doOnMessageReceive (String message), который срабатывает, когда мы получаем новое уведомление от сервиса PushWoosh. В данном методе реализована обработка и вывод всплывающего уведомления на экран мобильного устройства (см. листинг 4).

Листинг 4 – Обработка и вывод уведомления на экран пользователя

```
@Override public void doOnMessageReceive(String message) {
    String show = parsePushNotification(message); // сообщение приходит в формате JSON, поэтому
    его необходимо распарсить
    NotificationCompat.Builder builder = new NotificationCompat.Builder(getApplicationContext()) //
    создаем уведомление
    .setSmallIcon(R.drawable.info) // устанавливаем небольшую иконку для нашего сообщения
    .setContentTitle("Музыканты России") // задаем заголовок
    .setContentText(show) // помещаем текст уведомления
    .setVibrate(new long[]{10, 1000}) // добавляем вибрацию к нашему оповещению
    .setColor(Color.argb(255,45,192,233))
```

```
.setAutoCancel(true); // сообщение автоматически исчезает из меню уведомлений, когда пользо-
ватель к нему прикасается Intent
intent = new Intent(getApplicationContext(), MainActivity.class); // создаем новый экземпляр Intent
PendingIntent pendingIntent = PendingIntent.getActivity(getApplicationContext(), 0, intent, 0); // предо-
ставляем право сервису PushWoosh для выполнения фрагмента кода
Notification notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION_
SERVICE); // Notification Manager – системный сервис Ан-
дроид, который управляет всеми уведомлениями builder.setContentIntent(pendingIntent); // предо-
ставляем ответ, который будет получен при нажатии на уведомление
Notification notification = builder.build(); // объединяем все свойства, которые были заданы для
уведомления и возвращаем экземпляр класса Notification
notificationManager.notify(1, notification); // объект notification передается в систему путем вы-
зова метода notificationManager.notify()
}
```

Как можно легко заметить, процесс обработки push-уведомлений в мобильном приложении «Музыканты России» достаточно прост и не требует каких-либо дополнительных знаний. Все что вам необходимо, это внимательно ознакомиться с инструкцией, которая представлена на официальном сайте, и выполнить все необходимые действия шаг за шагом. Такой простой способ интеграции push-уведомлений заметно выделяет сервис PushWoosh, который является несомненно одним из лучших современных решений для реализации push-уведомлений в своем приложении.

Заключение. В данной статье был представлен современный способ работы с всплывающими уведомлениями на примере мобильного приложения «Музыканты России». Были представлены основные преимущества и функциональные возможности сервиса уведомлений PushWoosh, а также разобран пример интеграции данного сервиса в мобильное приложение.

ЛИТЕРАТУРА

1. Блог App Global [Электронный ресурс] / Push-уведомления: взрывная статистика. – Режим доступа: <http://app-global.ru/blog/push-uedomleniya-vzrivnaya-statistika>. – Дата доступа: 24.09.2020.
2. PushWoosh [Электронный ресурс] / Features. – Режим доступа: [https:// www.pushwoosh.com/features](https://www.pushwoosh.com/features). – Дата доступа: 24.09.2020.