УДК: 004.42 **DOI: 10.35595/2414-9179-2019-1-25-414-422**

Ж.Е. Темиргалиев[1]

ОСОБЕННОСТИ РАЗРАБОТКИ ГЕОЛОКАЦИОННЫХ ПРИЛОЖЕНИЙ ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ С ПОМОЩЬЮ ФРЕЙМВОРКА REACT NATIVE

АННОТАЦИЯ

Геоинформационное обеспечение различных практических и научных картографических исследований на данный момент является одним из обязательных требований. Бурное развитие информационных и мобильных технологий в последнее десятилетие приводят к тому, что смартфоны используются не только в качестве телефона или развлекательного устройства, но и для выполнения различных производственных работ, связанных с картографией.

В подавляющем большинстве современных мобильных устройств есть приёмник сигналов GPS, а в некоторых присутствует также приемник сигналов ГЛОНАСС. С помощью данных приёмников пользователи отслеживают своё местоположение, рассчитывают расстояние между объектами, могут построить маршрут до пункта назначения, либо найти нужный объект на карте. Но для того, чтобы в полной мере воспользоваться данными возможностями, необходимо соответствующее программное обеспечение, разработанное под конкретную мобильную платформу. Самыми популярными мобильными операционными системами (ОС) являются iOS от компании Apple и Android от компании Google, вместе они занимают 99,9 % рынка мобильных ОС[2].

Как iOS, так и Android обладают своим программным интерфейсом (API), с помощью которого разработчики получают доступ к датчикам и приёмникам мобильного устройства, в частности к GPS. Большинство мобильных устройств уже имеют предустановленное геоинформационное программное обеспечение (ПО), такое как Apple Maps в устройствах на базе iOS и Google Maps в устройствах Android. Но данные приложения обладают лишь базовым функционалом, поэтому для разработки специфических геоинформационных приложений необходимо привлекать программистов. Учитывая различия между этими двумя операционными системами, зачастую нужны отдельные специалисты для разработки под iOS и Android с высокой оплатой труда[3]. Это в свою очередь ведёт к повышению бюджета разработки.

В данной статье рассматривается возможность разработки геоинформационных приложений с использованием универсальной библиотеки (фреймворка) React Native. Фреймворк React Native использует язык программирования JavaScript и позволяет с небольшими доработками адаптировать приложения под обе платформы iOS и Android. Язык программирования JavaScript очень популярен среди веб-разработчиков; без труда можно найти специалистов по данному языку с приемлемым бюджетом на разработку[4]. Таким образом, используя фреймворк React Native, можно сократить трудозатраты при разработке геоинформационных приложения для мобильных устройств.

[1] Университет КИМЭП, ул. Масанчи, 56, 0500012, Алматы, Казахстан, *e-mail:* **tzhe@yandex.ru**

[2] IXBT. iOS и Android занимают уже 99,9% рынка мобильных ОС. Электронный ресурс: https://www.ixbt.com/news/2018/02/24/ios-android-99-9.html (дата обращения: 01.10.2018)

[3] *Гуляева А.* Зарплаты мобильных разработчиков 2017: деньги, платформы, стаж и регионы. Электронный ресурс: https://apptractor.ru/info/articles/zarplata-mobilnyih-razrabotchikov-2017-dengi-platformyi-stazh-i-regionyi.html (дата обращения 01.10.2018)

[4] Trud. Обзор статистики зарплат профессии Веб-разработчик в России. Электронный ресурс: https://russia.trud.com/salary/692/67314.html (дата обращения 01.10.2018)

КЛЮЧЕВЫЕ СЛОВА: картография, мобильные приложения, геоинформационные приложения, React Native

Zholdasbek E. Temirgaliyev[1]

FEATURES OF DEVELOPMENT OF GEOLOCATION APPLICATIONS FOR MOBILE DEVICES WITH THE HELP OF FRAMEWORK REACT NATIVE

ABSTRACT

Geoinformation applications of different practice and research are one of the mandatory requirements. A rapid development of mobile technologies leads to using a mobile device not as an entertainment tool but also as a production tool. One of the examples of application is geoinformation applications.

Almost in every smartphone, there is a GPS receiver and somewhere a GLONASS receiver. With the help of data receivers, users track their position, calculate a distance between points, plot a route or find a required object on a map. In order to gain the most from these technologies, there should be appropriate software for certain mobile platform. The most popular mobile platforms are iOS from Apple and Android from Google. Together they occupy 99 % of the mobile OS market[2].

Both iOS and Android mobile OS have their application-programming interface (API). Using these API developers access different sensors and receivers of a mobile device, GPS receiver in particular. The most smartphones have preinstalled geoinformation software like Apple Maps on iOS devices and Google Maps on Android devices. But these applications have only basic functionality, so to implement specific software we need to hire developers. Given the differences between the two operating systems, often need individual specialists for development for iOS and Android with high pay[3]. And this causes a big budget for development.

In this article, we consider using React Native framework for mobile development. This framework uses JavaScript programming language and can provide application development for both iOS and Android platforms. JavaScript programming language is very popular among web-developers, and there are not any barriers to hiring a developer with a suitable salary[4]. So, using React Native we can reduce the budget for mobile geoinformation application development.

KEYWORDS: cartography, mobile applications, geoinformation applications, React Native

ВВЕДЕНИЕ

Развитие мобильных устройств, а именно появление в них приёмника GPS для геопозиционирования, существенно расширило возможности разработки новых геоинформационных приложений, которые могут отображать текущее местоположение устройства, различные маршруты, пройденное расстояние и так далее.

Для разработчика мобильных приложений открываются большие возможности для создания уникальных программ, использующих данные GPS-приёмника мобильного устройства. Но сложность разработки под различные мобильные ОС значительно повышает

[1] KIMEP University, Masanchi str., 56, 0500012, Almaty, Kazakhstan, *e-mail:* **tzhe@yandex.ru**

[2] IXBT. iOS and Android already occupy 99.9% of the mobile OS market. Web resource: https://www.ixbt.com/news/2018/02/24/ios-android-99-9.html (accessed 01.10.2018) (in Russian)

[3] *Gulyayeva A*. Salaries of mobile developers 2017: money, platforms, experience and regions. Web resource: https://apptractor.ru/info/articles/zarplata-mobilnyih-razrabotchikov-2017-dengi-platformyi-stazh-i-regionyi.html (accessed 01.10.2018) (in Russian)

[4] Work. Overview of statistics on the salaries of the profession Web developer in Russia. Web resource: https://russia.trud.com/salary/692/67314.html (accessed 01.10.2018) (in Russian)

бюджет поставленной задачи. Поэтому остро стоит задача поиска универсального инструмента для разработки, и желательно, чтобы данный инструмент позволял без значительных доработок создавать приложения одновременно для двух платформ: iOS и Android.

Важно, чтобы используемая технология не имела существенных ограничений при разработке в сравнении с технологиями, предназначенными для мобильных ОС. Например, компания Google рекомендует использовать язык программирования Java для разработки под Android, тогда как для платформы iOS используется язык программирования Swift, созданный компанией Apple.

Таким образом, остро встаёт вопрос, под какую систему разрабатывать приложение. В идеале, конечно, это нужно делать для обеих платформ. Но разные инструменты разработки повышают трудозатраты и бюджет. Однако в последнее время появились универсальные библиотеки и среды разработки, которые позволяют разрабатывать приложения одновременно для обеих платформ. Одним из таких инструментов является фреймворк React Native.

Под фреймворком (англицизм, неологизм от framework «остов, каркас, структура») понимается набор заготовок, шаблонов для программной платформы, определяющий структуру программной системы, а также программное обеспечение, облегчающее разработку и объединение разных модулей программного проекта. Фреймворк React Native использует язык программирования JavaScript и позволяет с небольшими доработками адаптировать приложения под обе платформы: iOS и Android. В последнее время он набирает большую популярность[1]. На рис. 1 представлен восходящий поисковый тренд по ключевому запросу React Native в поисковой системе Google. Данный график показывает, что фреймворк опережает по популярности классические методы разработки мобильных приложений.

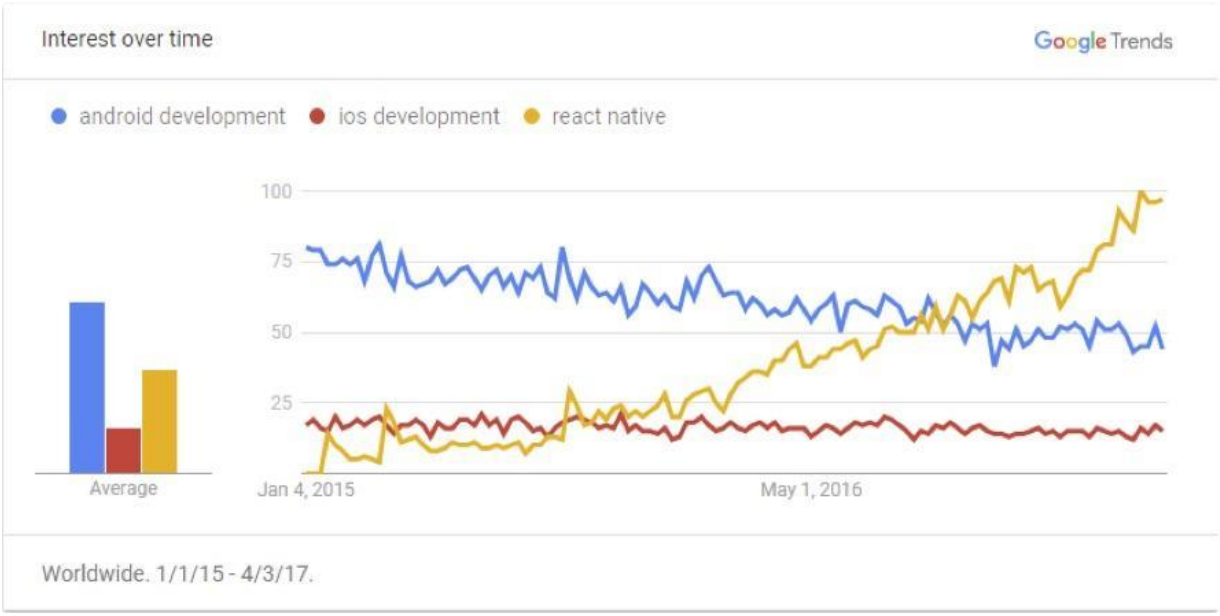

Рис. 1. Частота поисковых запросов по теме мобильной разработки
Fig. 1. The frequency of search queries on mobile development

[1] Gauri Pawar. Why is React Native the Future of Mobile App Development? Электронный ресурс: https://eluminoustechnologies.com/blog/2017/05/08/react-native-app-development/ (дата обращения 01.10.2018)

Фреймворк React Native использует язык программирования JavaScript. Данный язык программирования в настоящее время является самым популярным среди разработчиков[1]. Он с большим отрывом опережает классические языки программирования для мобильной разработки, такие как Java для Android и Swift для платформы iOS (рис. 2).

В данной статье приведён пример разработки геолокационного приложения для мобильных устройств. Для разработки используется фреймворк React Native.

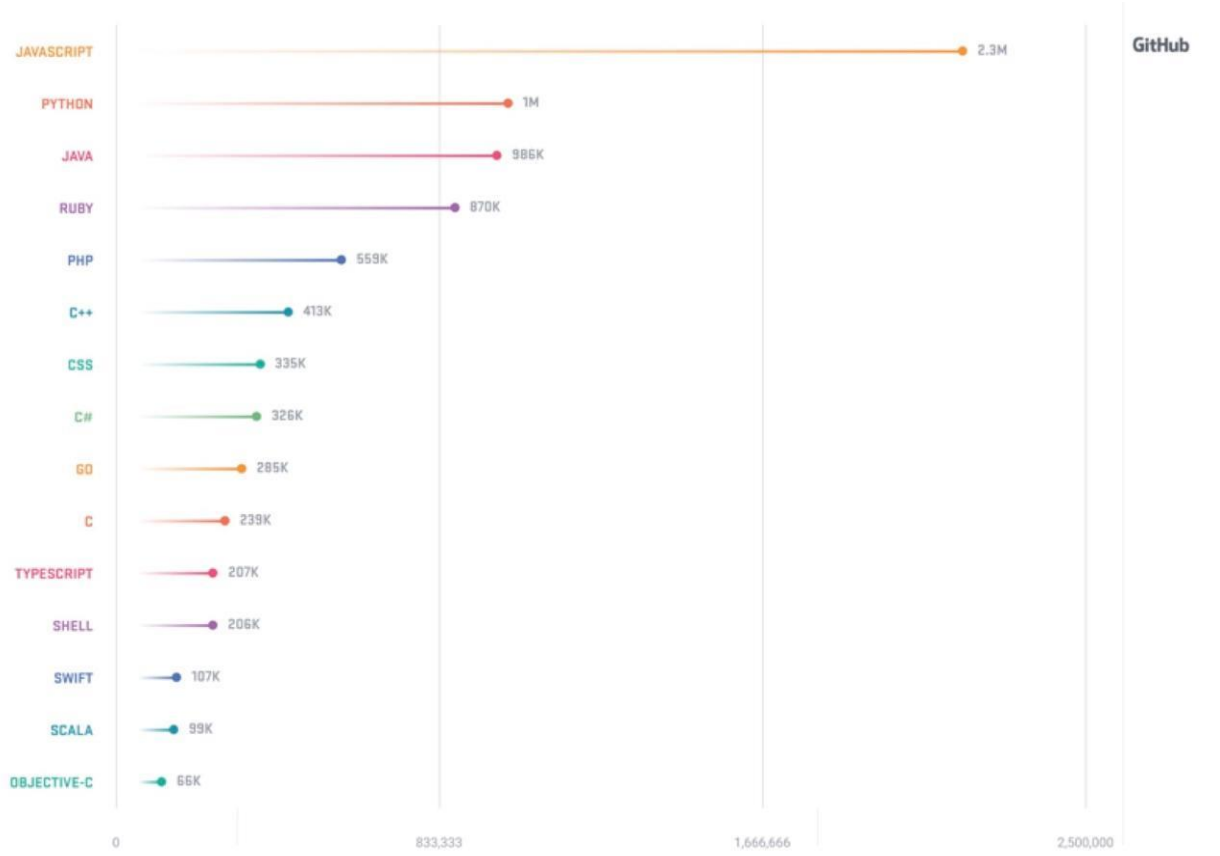

Рис. 2. Количество проектов на ресурсе GitHub по языкам программирования
Fig. 2. Number of GitHub projects on programming languages

МАТЕРИАЛЫ И МЕТОДЫ ИССЛЕДОВАНИЯ

Установка необходимых компонент

Целью работы является разработка геолокационного приложения для ознакомления с технологией React Native. Приложение будет отображать пройденный маршрут на карте. На первый взгляд может показаться, что это достаточно простое приложение, но для ознакомительных целей и апробации технологии React Native для разработки геоинформационных приложений этого вполне достаточно.

Для начала нужно установить библиотеку для работы с картами следующей командой:

```
npm install react-native-maps –save
```

[1] Octoverse. The State of the Octoverse 2017. Электронный ресурс: https://octoverse.github.com (дата обращения 01.10.2018)

Получение обновлённых координат

После установки компонента карт перейдем к программированию приложения. Для начала создадим конструктор, который будет хранить первоначальное состояние приложения:

```
constructor(props) {
  super(props);
  this.state = {

    latitude: LATITUDE,
    longitude: LONGITUDE,
    routeCoordinates: [],
    distanceTravelled: 0,
    prevLatLng: {},
    coordinate: new AnimatedRegion({
     latitude: LATITUDE,
     longitude: LONGITUDE
    })
  };
      }
```

Каждый раз, когда пользователь перемещается, необходимо обновлять координаты местоположения:

```
componentDidMount() {
  this.watchID = navigator.geolocation.watchPosition(
    position => {
      const { coordinate, routeCoordinates, distanceTravelled } =
this.state;
      const { latitude, longitude } = position.coords;

      const newCoordinate = {
        latitude,
        longitude
      };
      if (Platform.OS === "android") {
        if (this.marker) {
          this.marker._component.animateMarkerToCoordinate(
            newCoordinate,
            500
          );
        }
       } else {
        coordinate.timing(newCoordinate).start();
       }
       this.setState({
        latitude,
        longitude,
        routeCoordinates: routeCoordinates.concat([newCoordinate]),
        distanceTravelled:
         distanceTravelled + this.calcDistance(newCoordinate),

         prevLatLng: newCoordinate
       });
     },
     error => console.log(error),
     { enableHighAccuracy: true, timeout: 20000, maximumAge: 1000 }
  );
}
```

Метод `watchPosition` предоставляет обновлённые координаты в том случае, если пользователь перемещается. Переменные `latitude` и `longitude` содержат данные широты и долготы. Создадим переменную `newCoordinate` для хранения обновлённых координат; эти координаты будем использовать для маркера на карте. Платформы Anroid и iOS отличаются в том, как получать обновлённые координаты, поэтому используем условный оператор:

```
if (Platform.OS === "android") {
  if (this.marker) {
  this.marker._component.animateMarkerToCoordinate(

    newCoordinate,
    500
   );
  }
} else {
  coordinate.timing(newCoordinate).start();
}
```

После этого обновим состояние нашего приложения:

```
this.setState({
  latitude,
  longitude,
  routeCoordinates: routeCoordinates.concat([newCoordinate]),
  distanceTravelled: distanceTravelled + this.calcDistance(newCoordinate),
  prevLatLng: newCoordinate
});
```

Расчёт пройденного расстояния

Для хранения пройденного расстояния будем использовать переменную `distanceTravelled`. Чтобы её рассчитать, создадим функцию `calcDistance`, которая будет принимать параметр `newLatLng`, хранящий новые координаты, и `prevLatLng`, хранящий предыдущие координаты:

```
calcDistance = newLatLng => {
  const { prevLatLng } = this.state;
  return haversine(prevLatLng, newLatLng) || 0;
};
```

Из-за кривизны Земли для расчёта расстояния между двумя точками мы воспользовались формулой гаверсинуса [Korn, Korn, 2000]. Чтобы данная формула была доступна при разработке, нужно установить необходимую библиотеку:

```
npm install haversine
```

Отображение пройденного маршрута на карте

Для построения области карты, содержащей пройденный маршрут, создадим функцию `getMapRegion`:

```
getMapRegion = () => ({
  latitude: this.state.latitude,
  longitude: this.state.longitude,
  latitudeDelta: LATITUDE_DELTA,
  longitudeDelta: LONGITUDE_DELTA
});
```

Далее отобразим маршрут на карте:

```
<MapView
  style={styles.map}
  showUserLocation
  followUserLocation
  loadingEnabled
  region={this.getMapRegion()}
>
  <Polyline coordinates={this.state.routeCoordinates} strokeWidth={5} />
  <Marker.Animated

    ref={marker => {
      this.marker = marker;
    }}
    coordinate={this.state.coordinate}
  />
</MapView>
```

В данном случае мы использовали функцию `Polyline` из библиотеки Google Maps, чтобы отобразить маршрут на карте. Эта функция имеет свойства `coordinate`, в которых хранятся координаты. С помощью свойства `strokeWidth` мы можем задать толщину линии маршрута. Для отображения маркера текущего местоположения используется компонент `Marker.Animated`.

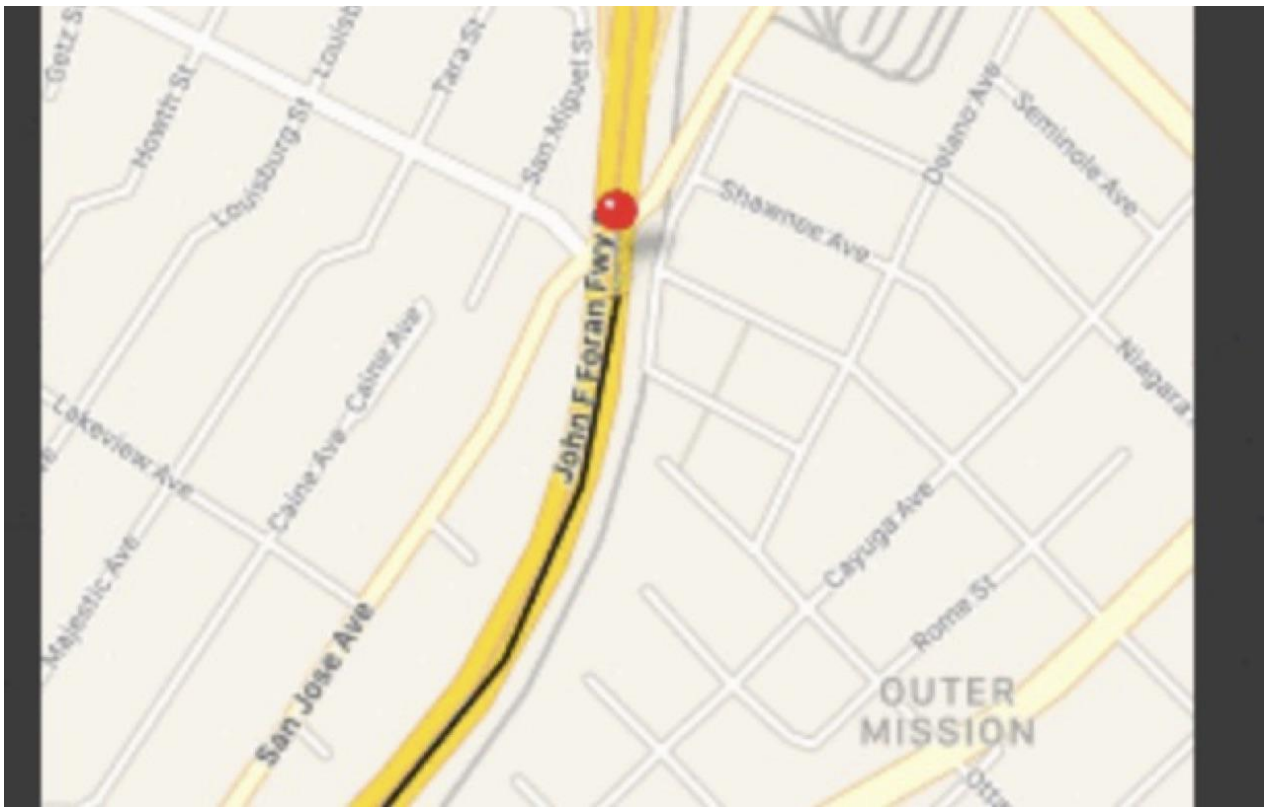

Рис. 3. Маркер текущего положения и пройденный маршрут
Fig. 3. Current position marker and travelled route

Отображение пройденного расстояния

Для того, чтобы показать пройденное пользователем расстояние, создадим элемент View с соответствующими стилями:

```
<View style={styles.buttonContainer}>
  <TouchableOpacity style={[styles.bubble, styles.button]}>
    <Text style={styles.bottomBarContent}>
      {parseFloat(this.state.distanceTravelled).toFixed(2)} km
    </Text>
  </TouchableOpacity>
</View>
```

Данный элемент представляет собой визуальный индикатор с числовым отображением пройденного расстояния.

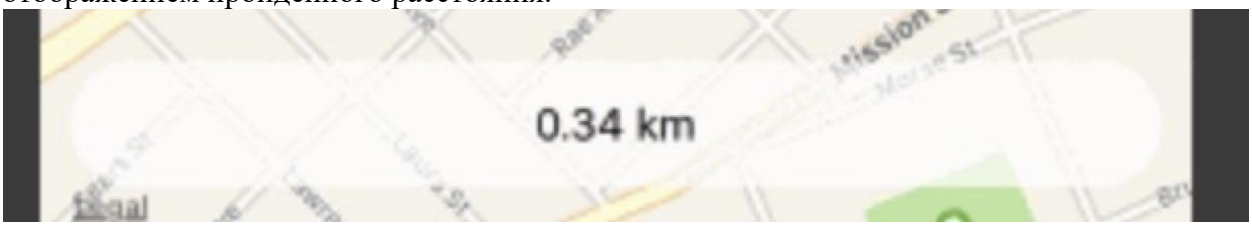

Рис. 4. Визуальный элемент отображения пройденного расстояния
Fig. 4. Visual element displaying the traveled distance

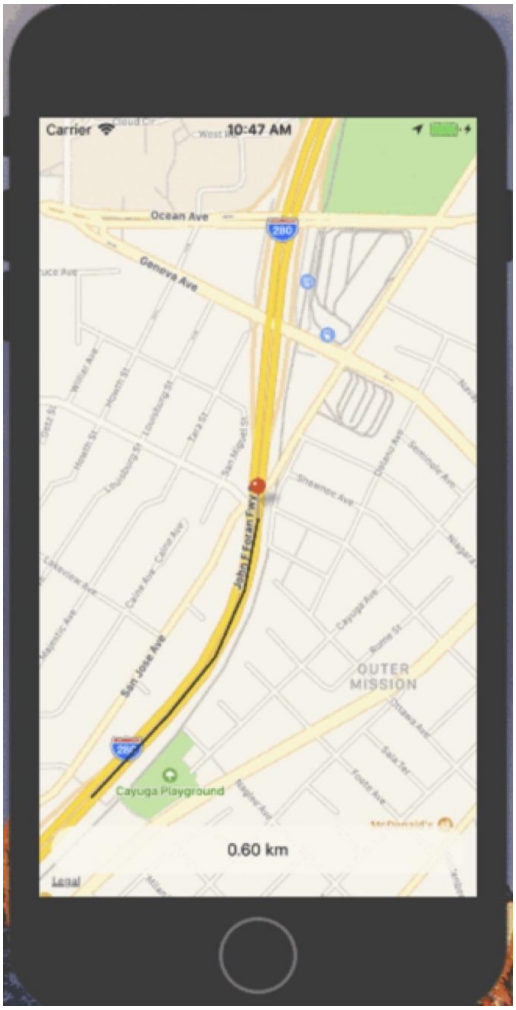

Рис. 5. Интерфейс приложения
Fig. 5. Application interface

Для того, чтобы протестировать приложение, можно воспользоваться симулятором iOS, который входит в набор разработчика Xcode, распространяемого бесплатно компанией Apple. Для этого в настройках симулятора надо войти в меню Debug > Location > Freeway Drive. Таким образом, можно смоделировать реальное устройство с включённой функцией GPS и проверить работу нашего приложения без необходимости установки приложения на реальное устройство.

В случае платформы Android функционал имитации движения устройства не предусмотрен в симуляторе, то есть нужно тестировать приложение на реальном устройстве.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ И ИХ ОБСУЖДЕНИЕ

Разработано геолокационное приложение, которое показывает пройденный путь и текущее местоположение. При этом важно, что данное приложение может быть установление как на платформу iOS, так и на платформу Android. Для разработки использовался фреймворк React Native. На рис. 5 представлен интерфейс разработанного приложения.

ВЫВОДЫ

В ходе исследования показано, что универсальный фреймворк React Native позволяет разрабатывать геоинформационные мобильные приложения для обеих платформ: iOS и Android. При этом интерфейс приложений практически не отличим от интерфейса, который бы получился в том случае, если бы разработка велась на нативных инструментах [Byung, Min, 2014]. Разработка по отдельности для платформ iOS и Android потребовала бы задействовать специалистов в этих областях, что повлекло бы за собой временные затраты на поиск специалистов и финансовые затраты на оплату их работы. Но, если ресурсы ограничены и есть жёсткие временные рамки для выпуска приложения, то фреймворк React Native предоставляет все возможности мобильных платформ для реализации тех или иных задач и использует для этого популярный язык программирования JavaScript. Причем данный язык программирования отличается лёгкостью для изучения и позволяет разработчику сделать своё приложение в разумные сроки.

Следовательно, целесообразно создавать приложения, используя фреймворк React Native для экономии производственных и временных ресурсов программистов.

СПИСОК ЛИТЕРАТУРЫ

1. *Byung W., Min A.* Usability improvement of mobile interface design optimized for smart phone and tablet PC. Environment, 2014. V. 9. № 22. P. 14549–14560.
2. *Korn G.A., Korn T.M.* Appendix B: B9. Plane and spherical trigonometry: formulas expressed in terms of the haversine function. Mathematical handbook for scientists and engineers: Definitions, theorems, and formulas for reference and review (3 ed.). Mineola, New York, USA: Dover Publications Inc, 2000. P. 892–893.

REFERENCES

1. *Byung W., Min A.* Usability improvement of mobile interface design optimized for smart phone and tablet PC. Environment, 2014. V. 9. No 22. P. 14549–14560.
2. *Korn G.A., Korn T.M.* Appendix B: B9. Plane and spherical trigonometry: formulas expressed in terms of the haversine function. Mathematical handbook for scientists and engineers: Definitions, theorems, and formulas for reference and review (3 ed.). Mineola, New York, USA: Dover Publications Inc, 2000. P. 892–893.