5. ИНФОРМАЦИОННЫЕ СИСТЕМЫ. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И СИСТЕМЫ АВТОМАТИЗАЦИИ

УДК 004.75, 630*584

Н.П. Васильев

МОБИЛЬНЫЕ CORDOVA-ПРИЛОЖЕНИЯ СБОРА ДАННЫХ О СОСТОЯНИИ ЛЕСНЫХ ТЕРРИТОРИЙ С ПРИВЯЗКОЙ К ГЕОПОЗИЦИИ

Введение. С ростом популярности мобильных устройств становятся актуальными задачи их использования для сбора первичных данных о состоянии лесных территорий. Однако разработка даже простых приложений для мобильных платформ остаётся довольно затратной, несмотря на рост и обилие предложений на этом рынке услуг. Объясняется это отчасти тем, что так называемая native-разработка для различных мобильных платформ значительно различается используемыми языками, технологиями и средами программирования. Так для операционной системы iOS – это языки Objective-C или Swift и среда разработки Xcode. Для Android – это язык Java и среда разработки Android NDK (Native Development Kit). Кроме того, разработка под iOS возможна только на дорогостоящей технике от Apple. И языки и среды достаточно сложны, поэтому разработка приложений под каждую платформу является уникальной и требует специалистов высокой квалификации.

Альтернативой native-разработки является гибридная разработка на основе технологии Cordova [Васильев, 2017; Васильев, 2018; Васильев, Лушкин, 2019]. Эта технология обладает рядом преимуществ:

– универсальным для всех платформ языком программирования JavaScript;

– кроссплатформенным подходом (один и тот же код можно использовать для iOS, Android, Windows Phone и других мобильных платформ);

– сокращением времени и стоимости разработки за счёт универсальности кода и сокращения сроков обучения;

– мощной экосистемой разработки с множеством ресурсов (Apache Cordova – это платформа с открытым исходным кодом. Ядро платформы обеспечивает доступ к основным возможностям мобильной операционной системы и функциям устройства, а эко-сообществом разработан широкий спектр подключаемых модулей, позволяющих задействовать дополнительные функциональные возможности устройств).

Достигается такая универсальность за счёт того, что Cordova внедряет WEB-код (JavaScript, HTML, CSS) в компонент мобильной платформы webview и предусматривает доступ к собственным ресурсам мобильного устройства (к файловой системе, геолокации и другой аппаратуре) из кода JavaScript через базовые плагины. Дополнительные плагины, разработанные эко-сообществом, значительно расширяют возможности гибридных приложений.

За универсальность приходится расплачиваться следующими недостатками этой технологии:

– проблемами с производительностью, особенно для задач с быстрой графикой (у webview есть некоторые проблемы при обработке графики, типичной для игр или приложений с динамичным графическим интерфейсом);

– поскольку webview разные для разных платформ (даже для разных версий), то могут потребоваться дополнительные настройки и оптимизация кода, чтобы приложение работало, как ожидается, на всех устройствах.

Эти недостатки не имеют значения для задач централизованного сбора данных. Вместе с тем *обычные WEB-приложения, для решения подобных задач имеют ряд ограничений* [Заяц, Хабаров, 2018]:

– работа WEB-приложения требует постоянного наличия сети для связи с WEB-сервером (отсутствует на лесных территориях);

– WEB-приложения не позволяют использовать геолокацию, камеру или иные возможности мобильного устройства.

Методика и результаты исследования. В данной статье предложен макет гибридного мобильного приложения на основе технологии Cordova для централизованного сбора данных о состоянии лесных территорий с автоматической привязкой их к геопозиции (см. рис. 1).

Преимуществами приложения являются:

– привязка атрибутивных данных к данным о местоположении;

– способность накапливать данные в автономном режиме, без доступа к сети, следовательно, без связи с сервером, что особенно важно для лесных территорий, где связь зачастую отсутствует;

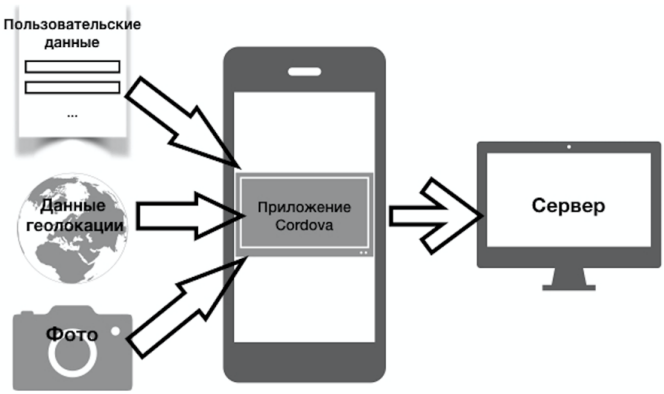

Рис. 1. Гибридное приложение Cordova сбора данных о состоянии лесных территорий

Fig. 1. Hybrid application Cordova for collecting data on the state of forest areas

– доступ к камере, что позволяет собирать фото и видеоматериалы;

– передача накопленных данных в централизованную базу данных при наличии связи с сервером;

– интеграция указанных возможностей в рамках единого приложения.

Для реализации представленных возможностей потребуются следующие плагины Cordova (https://cordova.apache.org/plugins/):

– cordova-plugin-geolocation «Geolocation» – для доступа к источникам информации о местоположении, включая GPS (Global Position System), сетевые сигналы (RFID (Radio Frequency IDentification), WiFi, идентификаторы сот GSM/CDMA);

– cordova-plugin-network-information «Network Information» – для получения информации о состоянии сотового или WiFi соединения и о доступе к сети Internet;

– cordova-plugin-camera «Camera» – этот плагин предоставляет API для фотографирования или выбора изображений из альбома изображений системы;

– cordova-plugin-file-transfer «File Transfer» – для передачи файлов на сервер (upload) или загрузки с сервера (download), а также для перемещения (или копирования) файлов в рамках файловой системы устройства.

Создание форм ввода данных с помощью HTML и CSS не представляет сложностей (здесь не обсуждаются современные фреймворки JavaScript,

где это не совсем тривиально). Более интересны для исследования возможности доступа к геолокации, камере и другим аппаратным сервисам мобильного устройства.

Исследование возможностей доступа к геолокации. Для работы с Cordova достаточно командной строки (cmd в Windows, терминал в macOS). Cordova – это модуль популярной платформы node.js [Заяц, Васильев, 2019], которая работает в любой операционной системе и её установка, обычно, не вызывает проблем (https://nodejs.org). Вместе с node будет установлен менеджер управления модулями npm. Выполняем в терминале с правами администратора команду установки модуля Cordova:

```
npm install –g cordova
```

Теперь можно стартовать проект Cordova:

```
cordova create app_path com.mycompany.myteam.myapp MyAppName
```

Здесь com.mycompany.myteam.myapp – это ID приложения (AppID в iOS), которое строится по таким же принципам, как и доменные имена серверов (DNS), только в обратном порядке и с учётом регистра. Следует тщательно продумывать этот идентификатор, особенно для iOS-приложений, которые реально планируется развивать вплоть до загрузки в App Store. Например, для приложения, демонстрирующего доступ к геолокации, AppID может выглядеть так: learn.CordovaTest.geolocation. Последний параметр – это название приложения, например CordovaGeolocationApp:

```
cordova create CordovaGeolocation learn.CordovaTest.geolocation
    CordovaGeolocationApp
```

В результате, будет создан каталог CordovaGeolocation с довольно сложной структурой, однако *разработка приложения ведётся исключительно в рамках подкаталога www.*

Далее потребуется добавить мобильную платформу (одну или несколько), например, iOS:

```
cordova platform add ios
```

А также плагин геолокации:

```
cordova plugin add cordova-plugin-geolocation
```

Все команды выполняются из каталога приложения – CordovaGeolocation.

В индексном файле созданного приложения (index.html в каталоге www) подгружается файл cordova.js и index.js (из подкаталога js). Первый файл без особой необходимости изменять не надо, впрочем, он появится только после сборки приложения в подкаталоге platforms/ios/www. Анали-

зируя второй файл, можно сделать вывод, что Cordova готова к работе после события deviceready, реакция на которое в сгенерированном приложении – сигнализировать мигающей эмблемой. Наша цель иная – начать работу с плагином геолокации:

```
var app = {
  watchID: null,

  initialize: function() {
    document.addEventListener('deviceready',
                    this.onDeviceReady.bind(this), false);
  },

  onDeviceReady: function() {
    var that = this;
    var options = {
      maximumAge: 3600000,
      timeout: 20000,
      enableHighAccuracy: true
    };
    that.watchID =
      navigator.geolocation.watchPosition(function() {
        that.onSuccess.apply(that, arguments);
      },
      function() {
        that.onError.apply(that, arguments);
      },
      options);
  },

  onSuccess: function(position) {
    for(key in position.coords)
     document.getElementById(key).innerText =
                                        position.coords[key];
     document.getElementById("timestamp").innerText =
 new Date(position.timestamp).toLocaleTimeString().split(" ")[0];
  },

  onError: function (error) {
    switch(error.code) {
     case error.PERMISSION_DENIED: alert('PERMISSION_DENIED');
break;
     case error.POSITION_UNAVAILABLE:
                        alert('POSITION_UNAVAILABLE'); break;
     case error.TIMEOUT: alert('TIMEOUT'); break;
     default:
      alert('code: ' + error.code + '\n' +
            'message: ' + error.message + '\n');
      break;
    }
  }
};
app.initialize();
```

Из представленного примера видно, что доступ к данным геолокации осуществляется через объект navigator.geolocation. Метод watchPosition() принимает три параметра: реакцию на успешное получение данных геолокации (обязательный), реакцию на ошибку, наконец, параметр, смысл которого – это время устаревания кэшированных данных (maximumAge),

максимальное время ожидания ответа (timeout), а также режим максимальной точности (enableHighAccuracy). Если не удаётся получить ответ за отведённое время (timeout), то допускается использование уже полученных (кэшированных) значений, если их возраст меньше заданного (maximumAge), иначе – ошибка error.TIMEOUT. Обычно эта ошибка возникает, если включён режим enableHighAccuracy, требующий использования GPS (а не сетевых источников), и указано недостаточное время maximumAge. Временные значения указываются в миллисекундах.

Результатом успешного запроса является объект position.coords. Свойства этого объекта: latitude – широта, longitude – долгота, altitude – высота, accuracy – точность измерений координат в метрах, altitudeAccuracy – точность измерения высоты в метрах, heading – направление движения в градусах, speed – скорость движения в метрах в секунду, timestamp – время измерения. В индексной странице предусмотрены соответствующие элементы для отображения значений этих свойств. Сборку приложения выполняем с помощью команды:

```
cordova build ios
```

В результате, в каталоге platforms/ios будет создан файл проекта с расширением xcodeproj, который можно загрузить в Xcode и выполнить окончательную сборку приложения. Результат его работы на эмуляторе представлен на рис. 2.

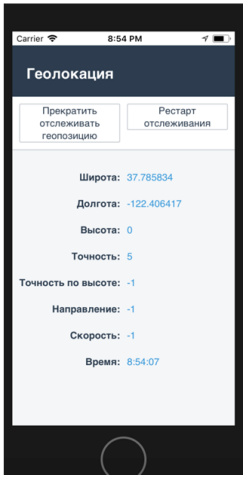

Рис. 2. Результат работы приложения на эмуляторе iPhone5
Fig. 2. The result of the application on the emulator iPhone5

Кроме watchPosition() интерфейс плагина предусматривает прекращение процесса наблюдений:

```
navigator.geolocation.clearWatch(this.watchID);
```

Или разовое получение данных геолокации:

```
navigator.geolocation.getCurrentPosition(onSuccess, [onError],
                                                    [options]);
```

Исследование возможностей использования камеры мобильного устройства. Плагин camera предоставляет полное имя графического файла, в котором содержится только что сделанный снимок или уже готовый снимок из альбома устройства (в зависимости от настроек). Этот файл находится в кэше устройства, т. е. в памяти, где его сохранность не гарантируется, поэтому желательно скопировать его в безопасное место – память, выделенную для приложения. Во вторую очередь, конечно, следует каким-то образом представить изображение. В Sencha Ext JS наиболее подходящим для этих целей является компонент Ext.Img, который располагает свойством src и соответствующим методом setSrc(), позволяющим установить путь к файлу с изображением.

Таким образом, логика работы с камерой выглядит следующим образом. Обращаемся к плагину (navigator.camera):

```
navigator.camera.getPicture(success, fail, options);
```

Если работа с камерой прошла успешно (отсняли изображение или выбрали его в альбоме), то управление передаётся функции обратного вызова – success, в которой копируем это изображение из кэша в память приложения, например в файл с именем 1.jpg:

```
function success(image_uri) {
    copyPhotoToFile('1.jpg', image_uri);
}
```

Функция copyPhotoToFile копирует файл image_uri, предоставленный плагином камеры, в файл 1.jpg, который уже находится в полном распоряжении приложения. Эта функция может быть реализована с помощью плагина cordova-plugin-file-transfer.

Если работа с камерой прошла не успешно: например, отказались от какого-либо выбора фото – отснятого или из альбома, или произошла ошибка, то управление передаётся функции fail:

```
function fail(message) {
    alert("Failed: " + message);
}
```

Здесь показываем пользователю формулировку ошибки, предоставленную плагином в переменной message.

В переменной options (она фигурирует выше в функции getPicture) задаются настроечные параметры. Вот некоторые из них:

```
var options = {
    quality: 70, // качество фото
    ,destinationType: // где разместить результат
           navigator.camera.DestinationType.FILE_URI // в файле
    ,sourceType: // камера или альбом
           navigator.camera.PictureSourceType.CAMERA
                                  //( или PHOTOLIBRARY)
    // формат изображения
    ,encodingType: navigator.camera.EncodingType.JPEG
    ,correctOrientation: true
};
```

Выводы. Предложен вариант организации централизованного сбора первичных данных о состоянии лесных территорий, включая фото, с привязкой их к местоположению с помощью гибридных приложений Cordova. Сбор возможен в автономном режиме, что важно при отсутствии связи на удалённых территориях. Разработаны шаблоны программных кодов для доступа к геолокации и камере мобильных устройств.

Библиографический список

Васильев Н.П. Гибридные технологии разработки приложений для мобильных платформ // Информационные системы и технологии: теория и практика: сб. научн. тр. Вып. 9. СПб.:СПбГЛТУ, 2017. С. 12–21.

Васильев Н.П. Универсальные технологии разработки мобильных приложений // Информационные системы и технологии: теория и практика: сб. научн. тр. Вып. 10. Ч. 1. СПб.: СПбГЛТУ, 2018. С. 23–30.

Васильев Н.П., *Лушкин Н.В.* Интеграция гибридных приложений Cordova с WEB-серверной обработкой графики // Информационные системы и технологии: теория и практика: сб. науч. тр. Вып. 11. СПб.: СПбГЛТУ, 2019. С. 10–24.

Заяц А.М., Хабаров С.П. Организация доступа к беспроводным AD HOC сетям информационных систем мониторинга лесных территорий из среды Windows 10 // Известия Санкт-Петербургской лесотехнической академии. СПб, 2018. Вып. 223. С. 285–299.

Заяц А.М., Васильев Н.П. Проектирование и разработка WEB-приложений. Введение в frontend и backend разработку на JavaScript и node.js: учеб. пособие. СПб.: Лань, 2019. 120 с.

References

Vasilev N.P. Gybridnye technologii rasrabotki prilogenij dlja mobilnyh platform. *Informacionnye sistemy i tehnologii: teorija i praktika: sb. nauchn. tr.,* 2017, is. 9, pp. 12–21. (In Russ.)

Vasilev N.P. Universal'nye tehnologii rasrabotki mobil'nyh prilogenij. *Informacionnye sistemy i tehnologii: teorija i praktika: sb. nauchn. tr.*, 2018, is. 10, pp. 23–30. (In Russ.)

Vasilev N.P., Lushkin N.V. Integracija gybridnyh prilogenij Cordova s WEB-servernoj obrabotkoj grafiki. *Informacionnye sistemy i tehnologii: teorija i praktika: sb. nauchn. tr.,* 2019, is. 11, pp. 10–24. (In Russ.)

Zajac A.M., Habarov S.P. Organizacija dostupa k besprovodnym AD HOC setjam informacionnyh sistem monitoringa lesnyh territorij iz sredy Windows 10. *Izvestija Sankt-Peterburgskoj lesotehnicheskoj akademii*, 2018, is. 223, pp. 285–299. (In Russ.)

Zajac A.M., Vasilev N.P. Proektirovanie i rasrabotka WEB-prilogenij. Vvedenie v frontend i backend razrabotku na JavaScript i node.js: Uchebnoe posobie. SPb.: Lan', 2019. 120 p. (In Russ.)

Материал поступил в редакцию 01.10.2019

Васильев Н.П. Мобильные Cordova-приложения сбора данных о состоянии лесных территорий с привязкой к геопозиции // Известия Санкт-Петербургской лесотехнической академии. 2020. Вып. 230. С. 265–274. DOI: 10.21266/2079-4304.2020.230.265-274

Предложен вариант организации централизованного сбора первичных данных о состоянии лесных территорий, включая фото, с привязкой их к местоположению с помощью гибридных приложений Cordova. Сбор возможен в автономном режиме, что важно при отсутствии связи на удалённых территориях. Разработаны шаблоны программных кодов для доступа к геолокации и камере мобильных устройств.

Ключевые слова: гибридные приложения Cordova, программирование для мобильных устройств.

Vasilev N.P. Mobile Cordova-applications for collecting data on the state of forest territories with reference to geolocation. *Izvestia Sankt-Peterburgskoj Lesotehniceskoj Akademii*, 2020, is. 230, pp. 265–274 (in Russian with English summary). DOI: 10.21266/2079-4304.2020.230.265-274

A variant of organizing a centralized collection of primary data on the state of forest territories, including photos, with reference to their location using the Cordova hybrid applications is proposed. Collection is possible offline, which is important in the absence of communication in remote territories. Software code templates have been developed for access to geolocation and the camera of mobile devices.

Keywords: Cordova hybrid applications, programming for mobile devices.

ВАСИЛЬЕВ Николай Павлович – доцент кафедры информационных систем и технологий Санкт-Петербургского государственного лесотехнического университета имени С.М. Кирова, кандидат технических наук.

194021, Институтский пер., д. 5, Санкт-Петербург, Россия. E-mail: nikpv@mail.ru

VASILEV Nikolai P. – PhD (Technical), Associate Professor of department of Information Systems and Technology of St.Petersburg State Forest Technical University.

194021. Institute per. 5. St. Petersburg. Russia. E-mail: nikpv@mail.ru