Физический браузер: концепция и обзор существующих решений API

А.К. Слабоузова, Д.Е. Намиот

***Аннотация*— Физический браузер представляет собой контекстно-зависимый браузер, в котором веб-приложение имеет доступ к информации об окружающей среде или контексту. Контекст обозначает местоположение, идентичность близлежащих людей и объектов, а также изменения этих объектов. Контекстной информацией для обычного смартфона может послужить информация о доступных Wi-Fi сетях и Bluetooth устройствах. В работе предполагается, что браузер представляет собой мобильное приложение, основанное на концепции сетевой близости. В рамках этой модели вычисление координат пользователя в системах, использующих информацию о местоположении заменяется информацией о доступности узлов беспроводных сетей. Именно близость к существующим или специально созданным сетевым узлам беспроводных сетей заменяет гео-вычисления в таком подходе. Такая модель также полностью исключает передачу информации о местоположении стороннему провайдеру.**

В работе рассматривается концепция физического браузера, а также текущие реализации API, которые могут быть использованы веб-приложением для получения необходимого взаимодействия с системным API устройства.

***Ключевые слова*— Physical Web Browser, Network proximity, Web API, Bluetooth, Wi-Fi.**

I. Введение

Физический браузер представляет собой контекстно-зависимый браузер, в котором веб-приложение имеет доступ к информации об окружающей среде. Согласно работе [1] подход к разработке веб-приложений может основываться как минимум на двух следующих моделях. В первой модели для каждого предполагаемого контекста разрабатывается и используется отдельное приложение. Во второй модели приложение одно, но оно адаптируется и подстраивается под окружающую среду или контекст.

A. ОПРЕДЕЛЕНИЕ КОНТЕКСТА

В оригинальной статье, в которой был введен термин «контекстно-зависимый» [2], контекст обозначает местоположение, идентичность близлежащих людей и объектов, а также изменения этих объектов. Другими словами, контекст - это любая информация, которая может быть использована для характеристики сущности, где сущность - это человек, место или объект, который считается релевантным для взаимодействия между пользователем и приложением. С точки зрения конечных пользователей контекстно-зависимое приложение собирает информацию, предоставляемую мобильным устройством, обрабатывает и предоставляет результат, локализованные данные или интерфейс, включая таргетированную рекламу, наиболее подходящий для данного места и времени (рис. 1) [3].

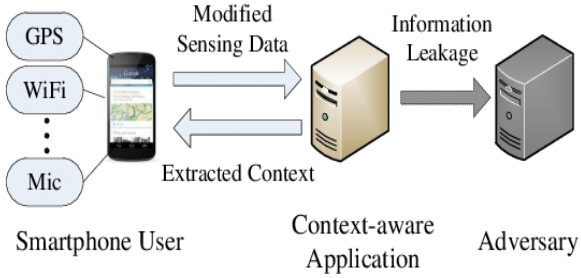

Рис. 1. Схема, иллюстрирующая работу контекстно-зависимого приложения

Контекстной информацией для обычного смартфона может послужить информация о доступных Wi-Fi сетях или Bluetooth устройствах, уровень шума, уровень освещенности, данные с микрофона и камеры, текущая скорость устройства, ориентация в пространстве, GPS данные и другие. Часто в основе контекстно-зависимого приложения лежит именно концепция сетевой близости [22].

B. СЕТЕВАЯ БЛИЗОСТЬ

Сетевая близость - это свойство пространственной близости, которая определяется видимостью одного сетевого устройства другим. Радиус действия сетевого устройства ограничен, поэтому его видимость другими устройствами является некоторой характеристикой местоположения этих устройств (рис. 2). В зависимости от этого местоположения пользователю предоставляется контекстно-зависимая информация и дополнительная функциональность.

Разработка сетевых технологий ближнего действия имеет относительно низкую стоимость, что увеличивает их распространенность. Такие технологии, как Bluetooth и Wi-Fi, на сегодняшний день являются повсеместными, и их распространенность в свою очередь снижает общую стоимость систем, так как требуется меньшее количество дополнительных затрат на оборудование -

Статья получена 12 июля 2021.
А.К. Слабоузова - МГУ имени М.В. Ломоносова (email: slabouzova@mail.ru)
Д.Е. Намиот - МГУ имени М.В. Ломоносова (email: dnamiot@gmail.com)

люди сами приобретают мобильные устройства, оснащенные Bluetooth и Wi-Fi [4].

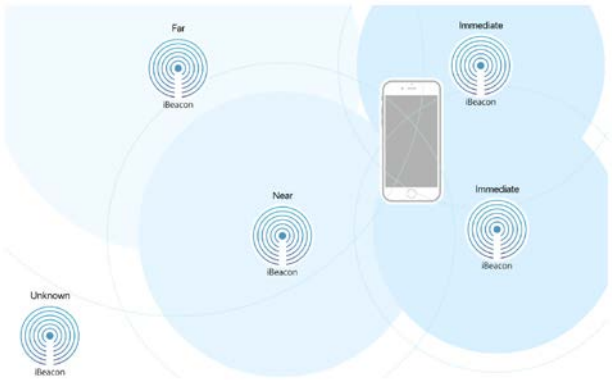

Рис. 2. Иллюстрация видимости мобильным устройств сетевых устройств в зависимости от их местоположения и радиуса действий

C. *МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ*

В качестве примера приложений, которые обращаются к системному API устройства можно привести работу [5]. Приложение с помощью Bluetooth получает идентификаторы окружающих людей, у которых также установлено данное приложение, и с помощью полученных идентификаторов находит страницы людей в известных социальных сетях. В другой работе [6] посетители умного музея используют Android-приложение, работающее в фоновом режиме. Приложение оценивает местоположение посетителя с помощью получения значений индикатора мощности принятого сигнала (RSSI) ближайших маяков BLE. В результате, когда посетители приближаются к экспонату, они получают уведомление, которое предлагает им ознакомится с экспонатом, рядом с которым они находятся.

D. *ВЕБ-ПРИЛОЖЕНИЯ*

Подобный функционал обращения к датчикам устройства интересен не только мобильным приложениям, но и веб-приложениям. Обращение к веб-сайтам через мобильные веб-браузеры становится все более и более популярным по сравнению с десктопными веб-браузерами: согласно статистике сайта StatCounter использование мобильных веб-браузеров на момент написания работы составляет 54.61%, использование десктопных веб-браузеров 42.87% [7]. Также разработка веб-приложения обычно является менее дорогим вариантом по сравнению с разработкой мобильных приложений под разные операционные системы.

Для доступа к информации о сетевой близости разработчики приложений напрямую используют системные библиотеки соответствующей операционной системы мобильного устройства. При этом для обращения к системному API мобильного устройства веб-приложению необходимы некоторое унифицированное API (обычно JavaScript API), и физический браузер, который будет поддерживать данное API.

В работе предлагается рассмотреть существующие веб-API, предоставляющие возможность веб-приложениям получения контекстной информации о ближайших точках Wi-Fi и ближайших Bluetooth устройствах, модель применения веб-приложений, использующих подобные API, а также пример реализации Android приложения, представляющего собой действующий прототип физического браузера.

E. *СТРУКТУРА РАБОТЫ*

Работа организована следующим образом. В разделе 2 будут представлены наиболее повсеместные беспроводные сетевые технологии - Wi-Fi и Bluetooth. В разделе 3 представлены модель применения веб-приложений, основанных на концепции сетевой близости. В разделе 4 - основные известные API, которые могут быть использованы при разработке физического браузера, имеющего возможность взаимодействовать с Wi-Fi и Bluetooth модулями и получать необходимую информацию о доступных Wi-Fi сетях и Bluetooth устройствах. В разделе 5 - пример реализации Android приложения физического браузера на основе компонента стандартной библиотеки Android - WebView. В разделе 6 представлены заключение и дальнейшее направление работы.

II. СЕТЕВЫЕ ТЕХНОЛОГИИ

A. *Bluetooth*

Bluetooth [8] представляет собой открытый стандарт IEEE для реализации беспроводных персональных сетей (WPAN). Bluetooth работает в нелицензируемом во всем мире радиочастотном спектре 2402-2480 МГц, предназначенном для устройств связи малого радиуса действия, подходящих для замены кабелей для принтеров, компьютерных мышей, клавиатур и т. д. [9]. Диапазон работы до 100 м.

Спецификация Bluetooth предусматривает два основных протокола: Basic Rate / Enhanced Data Rate (BR/EDR, так называемый «классический» Bluetooth) и Bluetooth Low Energy (BLE, ранее известный как Bluetooth Smart). Протокол BR/EDR построен на 79 каналах, с индексами от 0 до 78, и требует постоянного установленного соединения. Используется, в первую очередь, для передачи файлов, аудио и видеопотоков [5]. Протокол BLE построен на 40 каналах, с индексами от 0 до 39, причем индексы 37, 38 и 39 – это каналы адвертайзинга. BLE не требует обязательного установления соединения и используется для различных датчиков, маячков и прочих небольших устройств, ограниченных по энергопотреблению [5].

Создание протокола BLE было, прежде всего, ориентировано на устройства IoT, то есть устройства, которые долгое время способны работать на батарее или аккумуляторе, и которые по беспроводной технологии передают небольшой объем данных на довольно низкой скорости. Одной из наиболее перспективных технологий BLE являются маячки BLE. Маячки BLE представляют собой небольшие недорогие беспроводные передатчики с батарейным питанием [10]. Маяк периодически, с

интервалом от долей секунды до нескольких секунд, транслирует адвертайзинг пакеты стандарта BLE, не устанавливая само соединение. Используются для предоставления сервисов, основывающихся на локализации и близости [11].

B. Wi-Fi

Wi-Fi — технология беспроводной локальной сети на основе стандартов IEEE 802.11. Данный стандарт работает на частотах 2,4 и 5 ГГц из диапазона частот ISM. Каждая точка доступа Wi-Fi использует определенный канал шириной по меньшей мере 20 МГц для передачи своего идентификатора сети (SSID). Расстояние, на которое действует сигнал Wi-Fi, зависит от версии протокола, частоты, мощности передатчика, антенны, наличия препятствий. В среднем, бытовые роутеры обеспечивают покрытие не более 20-50 м внутри помещений и до 100-250 м на открытом пространстве. Специальные станции могут распространять сигнал на расстояния до нескольких сотен километров. В 2010 году появилась технология Wi-Fi Direct, позволяющая устройствам связываться напрямую, без маршрутизатора [12]. Таким образом, смартфон может стать точкой доступа Wi-Fi. Примерно 38% современных смартфонов поддерживают эту технологию.

В отличие от Bluetooth, Wi-Fi работает быстрее и на большие расстояния, но потребляет больше энергии. Для передачи данных без установления соединения больше подходит Bluetooth, который может транслировать до 254 байт, тогда как SSID имеет длину максимум 32 байт [5].

III. МОДЕЛЬ ПРИМЕНЕНИЯ

Основная модель применения физического браузера следующая. Пользователь заходит на сайт веб-приложения магазина, находясь на территории конкретной точки магазина или в непосредственной близости.

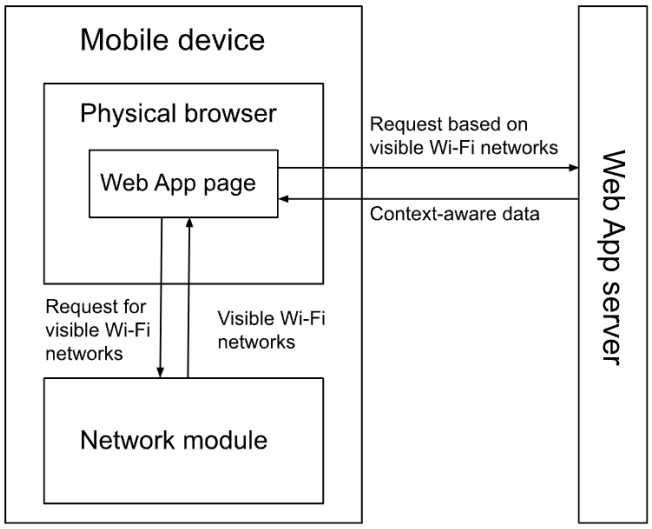

Рис. 3. Схема взаимодействия между веб-приложением и сетевым модулем на примере видимых Wi-Fi сетей

Веб-приложение, обращается с помощью, например, JavaScript API к сетевому модулю и фиксирует, есть ли поблизости знакомая приложению Wi-Fi сеть или знакомый идентификатор маяка Bluetooth (рис. 3).

При этом не происходит отслеживание мобильного абонента. Только в момент, когда пользователь заходит на сайт веб-приложения, происходит фиксация факта, находится ли пользователь на территории магазина и в какой из точек, если точек несколько.

В случае, если пользователь находится на территории магазина, ему предоставляется более удобно настроенные ассортимент магазина, различные специальные предложения и таргетированная реклама. Последнее особенно важно. Естественно, что с точки зрения бизнес-приложений, есть разница между пользователями, которые зашли на сайт торговой организации с мобильного устройства, находясь на территории этой организации и пользователями, которые зашли на этот же сайт, например, со своего десктоп-компьютера из дома или офиса. Физический браузер позволяет кастомизировать (настроить) выдачу веб-приложения. Отметим также, что физический браузер позволяет предоставлять всю информацию пользователям в рамках веб-приложений. Сейчас, для реальных посетителей торговые предприятия предлагают мобильные приложения. Соответственно, им необходимо разрабатывать и поддерживать как обычные сайты, так и мобильные приложения. Физический браузер позволяет разрабатывать только веб-приложения.

Физический браузер также удобен тем, что его идею можно продолжать развивать различными способами. Например, при предоставлении пользователем веб-приложению своего идентификатора в социальных сетях, можно отображать на странице, находятся ли другие его знакомые на территории магазина. Это работает как аналог отметки о местоположении (check-in), но информация об этом не будет оставляться в социальной сети (сохранение приватности пользователя). Эту информацию увидят только члены социального круга пользователя, которые именно в это время находятся в том же самом месте.

Физический браузер представляет собой средство разработки для приложений, использующих информацию о местоположении, но основанных на модели сетевой пространственной близости [23]. Такие приложения не используют информацию GPS и могут применяться, например, в помещениях или на подвижных объектах, где использование GPS затруднено. Например, определение нахождения мобильного пользователя в транспортном средстве [24, 25]. Принципы сетевой пространственной близости могут использоваться в IoT и M2M приложениях [26], а также в так-называемых супер-приложениях, как наиболее простой способ кастомизировать выдачу информации в зависимости от местоположения пользователя. Узлы беспроводных сетей обеспечивают физическую разметку местности (помещения). И при этом обеспечивается единая форма представления данных, что позволяет избежать их фрагментации [27].

IV. WEB API

Продвижением интерфейсов для работы с сетевыми устройствами на языке JavaScript занимаются две группы разработки - Mozilla Developer и группа разработки Chromium. Международной стандартизацией новых возможностей для веб-разработки занимается Консорциум Всемирной паутины W3C [4].

Отметим, что для получения почти всех данных с сенсоров используется в основном асинхронное системное API устройства. Это означает, что и API, предназначенное для получения веб страницей дополнительных контекстно-зависимых данных будет асинхронным. Асинхронные методы либо принимают функции обратного вызова (callback функции), либо возвращают *Promise* [13] - удобный объект JavaScript, предназначенный для того, чтобы асинхронно выполнять функции, а также строить асинхронные последовательные цепочки вызовов.

A. MOZILLA DEVELOPER

Mozilla Developer представляет следующие интересные в рамках работы API-интерфейсы: WiFi Information API, Web Bluetooth API, а также Network Information API. API не являются стандартизированными и поддерживаются не всеми браузерами. Некоторые API являются экспериментальными, что означает, они в любое время могут быть существенно изменены.

1) WiFi Information API

WiFi Information API [14] на данный момент поддерживается только платформой Firefox OS. Однако является хорошим примером API, который может быть встроен в физический браузер.

WiFi Information API интерфейс предназначен для поиска доступных сетей Wi-Fi и управления подключением и отключением устройства от этих сетей.

Интерфейс WifiManager, являющийся частью API, предоставляет информацию о том, включен ли Wi-Fi, позволяет подписаться на событие включения/отключения Wi-Fi. Также предоставляет информацию о текущем соединении и позволяет обнаружить доступные сети и информацию о них (SSID, силу сигнала в абсолютном значении и относительном значении и др.)

Пример получения SSID WiFi сети с наиболее сильным сигналом:

```
  var wifi = navigator.mozWifiManager;
  function sortNetworksByStrength(a, b) {
  return a.signalStrength>b.signalStrength ?
-1 : 1;
  }

  var request = wifi.getNetworks();
  request.onsuccess = function() {
  console.log('Network with best signal: ’);
  var networks = this.result;
  networks.sort(sortNetworksByStrength);
  console.log(network[0].ssid);
  }
```

2) Web Bluetooth API

Web Bluetooth API [15] представляет собой экспериментальное API, поддерживается в известных мобильных браузерах Chrome Android и Opera Android. Не поддерживается WebView, мобильным браузером Firefox и мобильным браузером Safari. API обеспечивает возможность подключения и взаимодействия с периферийными устройствами Bluetooth Low Energy. Рассмотрим два основных метода интерфейса Bluetooth, являющегося частью API.

Метод requestDevice() возвращает Promise<BluetoothDevice> с указанными при вызове метода параметрами. Параметрами могут быть сервисы, реализованные через Bluetooth устройства, конкретные названия соединений или префикс к названиям:

```
let options = {
  filters: [
    {services: ['heart_rate']},
    {name: 'ExampleName'},
    {namePrefix: 'Prefix'}
  ],
  optionalServices: ['battery_service']
}
navigator.bluetooth.requestDevice(options)
.then(function(device) {
  console.log('Name: ' + device.name);
})
.catch(function(error) {
  console.log('Error: ' + error);
});
```

Метод *getDevices()* возвращает *Promise<sequence<BluetoothDevice>>* устройств, с которыми до этого уже было установлено соединение.

3) Network Information API

Network Information API [16] также представляет собой экспериментальное API. API поддерживается WebView, мобильными браузерами Chrome, Firefox и Opera. Не поддерживается мобильным браузером Safari. API позволяет веб-приложениям получать доступ к информации о сетевом соединении, используемом устройством в данный момент времени, а также позволяет подписаться на событие переключения на другую сеть.

С помощью таких свойств как, например, *downlink* и *type* можно получить информацию о текущей скорости соединения и типу подключения ("bluetooth", "cellular", "wifi" и др.).

Свойство *onchange* представляет собой *callback* функцию, которая будет запущена при изменении основного объекта NetworkInformation.

API позволяет разработчикам вносить динамические изменения в свой пользовательский интерфейс, чтобы информировать пользователя об изменении типа сетевого подключения. Это также позволяет приложениям, которые откладывали передачу больших объемов данных, автоматически запускаться при обнаружении сети с высокой пропускной способностью. Или при переключении на тип соединения "cellular" предупреждать пользователя, что оператор сотовой связи может взимать плату за использовании сети.

Пример контролирования использования сети для загрузки данных большого размера в зависимости от скорости соединения:

```
  //текущая скорость соединения
  var rate = navigator.connection.downlink;
  function changeHandler(event) {
    var lastRate = navigator.connection.downlink;
    if (lastRate > rate) {
      //запустить процесс загрузки данных
    } else {
      //приостановить процесс загрузки данных
    }
    rate = lastRate;
  }
  navigator.connection.onchange = changeHandler;
```

B. CHROMIUM

Chrome предлагает два API-интерфейса, предназначенные для работы с Bluetooth устройствами: Chrome.bluetoothLowEnergy и chrome.bluetooth. Представленные API являются устаревшими (deprecated) и поддерживаются только устройствами с операционной системой Chrome OS. На момент написания работы Chrome не предоставляет API-интерфейса для работы с Wi-Fi сетями.

Chrome.bluetooth API [17] используется для поиска Bluetooth устройств и получения информации о текущем Bluetooth адаптере устройства: MAC адрес устройства, доступно ли, находится ли в процессе обнаружения других устройств и т.п. Также можно подписаться на события изменения состояния текущего адаптера устройства и добавления/изменения/отсоединения Bluetooth устройств.

```
  chrome.bluetooth.getDevices(function(devices) {
    for (var i = 0; i < devices.length; i++) {
      console.log('Name: ' + devices[i].name);
    }
  });
  chrome.bluetooth.startDiscovery(function() {
    setTimeout(function() {
      chrome.bluetooth.stopDiscovery(function() {});
    }
  }, 3000);
```

Chrome.bluetoothLowEnergy API [18] используется для работы с BLE устройствами. API предоставляет методы для соединения/отсоединения с устройством с конкретным MAC адресом, для получения и создания GATT сервисов, дескрипторов и характеристик, запуск/отмену процесса адвертайзинга, подписку на события изменения значений характеристик, дескрипторов и некоторые другие.

C.W3C GEO

Один из известных веб-стандартов - это W3C Geolocation API [19]. API поддерживается всеми известными мобильными и десктопными браузерами. Предназначен для использования в веб-приложениях (через JavaScript) и позволяет веб-приложению, запущенному на удаленном сервере, получать доступ к данным о местоположении пользователя. Для этого необходимо, чтобы либо устройство, на котором запущен браузер, предоставляло браузеру один или несколько механизмов определения местоположения (например, GPS), либо механизм предоставляется через какое-либо другое стороннее приложение.

API предоставляет информацию о местоположении выраженную парой координат (долготой и широтой). Также предоставляет точность полученных данных, высоту над уровнем моря, скорость, поворот относительно севера. Позволяет приложению при формировании запроса на местоположение указывать, нужен ли высокий уровень точности, максимальное количество времени на получение запроса и возраст записи, если данные берутся из кеша. Предоставляет подписку на получение обновление данных.

При этом API не зависит от какого-либо конкретного метода определения местоположения и может работать с любым механизмом, доступным браузеру: GPS, IP-адрес, RFID, WiFi, MAC-адрес Bluetooth, и др. Однако использование методов, отличных от GPS, может давать значительно меньшую точность.

Пример получения текущих координат с ограничением времени в 10 секунд:

```
  function successCallback(position) {
    console.log('Request finished in under 10 seconds');
    latitude = position.coords.latitude;
    longitude = position.coords.longitude;
    console.log('Latitude: ' + latitude);
    console.log('Longitude: ' + longitude);
  }

  function errorCallback(error) {
    console.log('Request failed or timeouted: ' + error.code);
  }

  navigator.geolocation.getCurrentPosition(
    successCallback,
    errorCallback,
    {timeout: 10000}
  );
```

D.ANDROID WEBVIEW

Компонент *WebView* [20] стандартной библиотеки Android позволяет отображать веб-приложение как часть другого приложения, возможно и не выполняющего функцию браузера. *WebView* полезен, когда нужен усиленный контроль над пользовательским интерфейсом и расширенные параметры конфигурации, которые позволяют встраивать веб-страницы в специально разработанную среду для мобильного приложения.

Компонент *WebView* содержит метод *addJavascriptInterface()*, принимающий произвольный объект, методы которого можно будет будет вызывать из веб-приложения с помощью Javascript кода напрямую. Сам объект также может вызывать известные ему методы Javascript кода веб-приложения, что

позволяет Android приложению передавать данные о физическом окружении напрямую веб-приложению. Необходимое взаимодействие с Wi-Fi модулем и Bluetooth модулем, включая получение информации о доступных сетях или устройствах, осуществляется с помощью классов *android.bluetooth.BluetoothAdapter и android.net.wifi.WiFiManager*.

В следующем разделе будет показан пример использования интерфейса *JavascriptInterface*.

V. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

В качестве прототипа было реализовано тестовое Android приложение [21], имитирующее работу физического браузера. Приложение основано на компоненте WebView. Приложение с помощью WebView отображает HTML страницу, которая с помощью JavaScript кода может взаимодействовать с физическим окружением Android устройства, а именно отображать текущую Wi-Fi сеть, к которой подключено устройство и список доступных Wi-Fi сетей, которые видны устройству.

В приложении реализован класс JavascriptInterfaceImpl - наследник интерфейса JavascriptInterface. У компонента WebView вызывается метод addJavascriptInterface(), позволяющий связать JavascriptInterfaceImpl с объектом androidInterface JavaScript кода.

webView.addJavascriptInterface(new JavascriptInterfaceImpl(this, webView), "androidInterface");

Теперь JavaScript код страницы сможет вызывать методы объекта JavascriptInterfaceImpl. Методы вызываются через обращение к androidInterface. Выглядит в JavaScript коде следующим образом:

```
var ssid = androidInterface.getWiFiSSID();
```

И наоборот, через *WebView* можно вызвать JavaScript методы. Так как сканирование доступных Wi-Fi сетей – долгий процесс, требующий вызова Android намерения SCAN_RESULTS_AVAILABLE_ACTION, то необходимо чтобы *JavascriptInterfaceImpl* дождался ответа системы и передал результат JavaScript коду.

webView.loadUrl("javascript:showAvailableWifis('" + ssids + "');")

В результате процесс передачи данными между веб-страницей и Android приложением налаживается. В HTML коде созданы две кнопки: “Get current Wi-Fi name” и “Scan available Wi-Fis”. По нажатию первой можно получить название текущей Wi-Fi сети (название показывается с помощью объявления Toast), по нажатию второй выводятся все доступные Wi-Fi сети.

VI. ЗАКЛЮЧЕНИЕ

Существует достаточно большое количество мобильных приложений, основанных на концепции сетевой близости. Вполне возможно разработку подобных приложений можно обратить в разработку веб-приложений, что более удобно, так как минимизирует количество установленных приложений. Однако для таких приложений необходим физический браузер, который позволит веб-приложениям обращаться к сенсорам смартфонов.

На данный момент единого стандартизированного решения API не существует. Mozilla Developer и Chromium предлагают приблизительно равные по возможностям API для работы с Bluetooth устройствами. При этом только Mozilla Developer предлагает какое-либо решение для работы с Wi-Fi сетями, однако решение поддерживается только на платформе Firefox OS (рис. 4).

	Chrome	Opera	Firefox	Safari	WebView
WiFi Information API					
Web Bluetooth API					
Network Information API					
Chrome.bluetoothLowEnergy API					
Chrome.bluetooth API					
W3C Geolocation API					

Рис. 4. Таблица, иллюстрирующая, какие мобильные браузеры (а также компонент WebView) поддерживают или не поддерживают рассмотренные веб-API. Зеленый цвет означает “поддерживают”, красный – “не поддерживают”.

В дальнейшей работе можно рассмотреть API, предназначенные для получения остальной контекстной информации как уровень освещение или шума, запись с микрофона или камеры. Конечной целью является создание удобного унифицированного API для физического браузера, веб-приложения которого имеют доступ к физическому окружению устройства. К созданию подобного API, способного обращаться к сенсорам устройства, можно добавить прикладное API, решающее более общие задачи.

БИБЛИОГРАФИЯ

[1] Namiot D., Sneps-Sneppe M. Wireless networks sensors and social streams //2013 27th International Conference on Advanced Information Networking and Applications Workshops. – IEEE, 2013. – С. 413-418.

[2] G. Schilit, B. Theimer, “Disseminating Active Map Information to Mobile Hosts”,. IEEE Network, vol. 8, no. 5, 1994, pp. 22-32.

[3] Namiot D., Sneps-Sneppe M. On mobile Bluetooth tags //arXiv preprint arXiv:1502.05321. – 2015.

[4] Волосникова П. М., Намиот Д. Е. Использование веб-интерфейсов для сервисов на основе сетевой близости //International Journal of Open Information Technologies. – 2020. – Т. 8. – №. 6. – С. 83-90.

[5] Намиот Д. Е., Макарычев И. П. Об альтернативной модели отметки местоположения в социальных сетях //International Journal of Open Information Technologies. – 2020. – Т. 8. – №. 2. – С. 74-90.

[6] Spachos P., Plataniotis K. N. BLE beacons for indoor positioning at an interactive IoT-based smart museum //IEEE Systems Journal. – 2020. – Т. 14. – №. 3. – С. 3483-3493.

[7] Статистика использования мобильных и десктопных веб-браузеров https://gs.statcounter.com/platform-market-share/desktop-

mobile-tablet/worldwide/#monthly-201612-202110. Retrieved: Oct, 2021
[8] Bluetooth SIG official specification, online source: https://www.bluetooth.com/specifications/specs/. Retrieved: June, 2021
[9] M.A.Sofi, “Bluetooth Protocol in Internet of Things (IoT), Security Challenges and a Comparison with Wi-Fi Protocol: A Review,” International Journal of Engineering and Technical Research V5, November 2016
[10] K.E.Jeon, J.She, P.Soonsawad, P.C.Ng, “BLE Beacons for 'Internet of Things Applications: Survey, Challenges and Opportunities,” IEEE Internet of Things Journal, PP(99):1-1, April 2018
[11] P.Spachos, K.Plataniotis, “BLE Beacons in the Smart City: Applications, Challenges, and Research Opportunities,” IEEE Internet of Things Magazine, vol.3, April 2020, pp. 14-18
[12] Wi-Fi Direct. https://www.wi-fi.org/discover-wi-fi/wi-fi-direct. Retrieved: June, 2021
[13] Promise in JavaScript https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise. Retrieved: June, 2021
[14] WiFi Information API https://developer.mozilla.org.cach3.com/en-US/docs/WebAPI/WiFi_Information. Retrieved: June,2021
[15] Web Bluetooth API https://developer.mozilla.org/en-US/docs/Web/API/Web_Bluetooth_API. Retrieved: June, 2021
[16] Network Information API https://developer.mozilla.org/en-US/docs/Web/API/Network_Information_API. Retrieved: June,2021
[17] Chrome Bluetooth API https://developer.chrome.com/docs/extensions/reference/bluetooth/ . Retrieved: June, 2021
[18] Chrome BluetoothLowEnergy API https://developer.chrome.com/docs/extensions/reference/bluetoothLowEnergy/. Retrieved: June, 2021
[19] W3C Geolocation API https://www.w3.org/TR/geolocation/. Retrieved: June,2021
[20] Android class WebView https://developer.android.com/reference/android/webkit/WebView. Retrieved: June, 2021
[21] Test Physical Browser model https://github.com/Anna-Sl/physical-browser. Retrieved: June, 2021
[22] Namiot D. Context-Aware Browsing--A Practical Approach //2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies. – IEEE, 2012. – С. 18-23.
[23] Namiot D., Sneps-Sneppe M. Proximity as a service //2012 2nd Baltic Congress on Future Internet Communications. – IEEE, 2012. – С. 199-205.
[24] Kupriyanovsky V. et al. On internet of digital railway //International journal of open information technologies. – 2016. – Т. 4. – №. 12. – С. 53-68.
[25] Куприяновский В. П. и др. Цифровая железная дорога-прогнозы, инновации, проекты //International Journal of Open Information Technologies. – 2016. – Т. 4. – №. 9.
[26] Sneps-Sneppe M., Namiot D. About M2M standards and their possible extensions //2012 2nd Baltic Congress on Future Internet Communications. – IEEE, 2012. – С. 187-193.
[27] Куприяновский В. П. и др. Цифровая экономика и Интернет Вещей-преодоление силоса данных //International Journal of Open Information Technologies. – 2016. – Т. 4. – №. 8. – С. 36-42.

Physical Browser: Concept and Overview of Existing API Solutions

Anna Slabouzova, Dmitry Namiot

***Abstract*— A physical browser is a context-sensitive browser in which a web application has access to information about the environment or context. Context refers to the location, identity of nearby people and objects, and changes in those objects. Information about available Wi-Fi networks and Bluetooth devices can serve as contextual information for an ordinary smartphone. The work assumes that the browser is a mobile application based on the concept of network proximity. In this model, the computation of user coordinates in systems using location information is replaced by information about the availability of wireless nodes. It is the proximity to existing or custom-built wireless network nodes that replaces geo-computing in this approach. This model also completely eliminates the transfer of location information to a third-party provider.**

The paper discusses the concept of a physical browser, as well as current API implementations that can be used by a web application to obtain the necessary interaction with the device's system API.

***Keywords*— Physical Web Browser, Network proximity, Web API, Bluetooth, Wi-Fi.**

REFERENCES

[1] Namiot D., Sneps-Sneppe M. Wireless networks sensors and social streams //2013 27th International Conference on Advanced Information Networking and Applications Workshops. – IEEE, 2013. – S. 413-418.
[2] G. Schilit, B. Theimer, “Disseminating Active Map Information to Mobile Hosts”,. IEEE Network, vol. 8, no. 5, 1994, pp. 22-32.
[3] Namiot D., Sneps-Sneppe M. On mobile Bluetooth tags //arXiv preprint arXiv:1502.05321. – 2015.
[4] Volosnikova P. M., Namiot D. E. Ispol'zovanie veb-interfejsov dlja servisov na osnove setevoj blizosti //International Journal of Open Information Technologies. – 2020. – T. 8. – #. 6. – S. 83-90.
[5] Namiot D. E., Makarychev I. P. Ob al'ternativnoj modeli otmetki mestopolozhenija v social'nyh setjah //International Journal of Open Information Technologies. – 2020. – T. 8. – #. 2. – S. 74-90.
[6] Spachos P., Plataniotis K. N. BLE beacons for indoor positioning at an interactive IoT-based smart museum //IEEE Systems Journal. – 2020. – T. 14. – #. 3. – S. 3483-3493.
[7] Statistika ispol'zovanija mobil'nyh i desktopnyh veb-brauzerov https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-201612-202110. Retrieved: Oct, 2021
[8] Bluetooth SIG official specification, online source: https://www.bluetooth.com/specifications/specs/. Retrieved: June, 2021
[9] M.A.Sofi, “Bluetooth Protocol in Internet of Things (IoT), Security Challenges and a Comparison with Wi-Fi Protocol: A Review,” International Journal of Engineering and Technical Research V5, November 2016
[10] K.E.Jeon, J.She, P.Soonsawad, P.C.Ng, “BLE Beacons for 'Internet of Things Applications: Survey, Challenges and Opportunities,” IEEE Internet of Things Journal, PP(99):1-1, April 2018
[11] P.Spachos, K.Plataniotis, “BLE Beacons in the Smart City: Applications, Challenges, and Research Opportunities,” IEEE Internet of Things Magazine, vol.3, April 2020, pp. 14-18
[12] Wi-Fi Direct. https://www.wi-fi.org/discover-wi-fi/wi-fi-direct. Retrieved: June, 2021
[13] Promise in JavaScript https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise. Retrieved: June, 2021
[14] WiFi Information API https://developer.mozilla.org/en-US/docs/Archive/B2G_OS/API/WiFi_Information_API. Retrieved: June,2021
[15] Web Bluetooth API https://developer.mozilla.org/en-US/docs/Web/API/Web_Bluetooth_API. Retrieved: June, 2021
[16] Network Information API https://developer.mozilla.org/en-US/docs/Web/API/Network_Information_API. Retrieved: June,2021
[17] Chrome Bluetooth API https://developer.chrome.com/docs/extensions/reference/bluetooth/ . Retrieved: June, 2021
[18] Chrome BluetoothLowEnergy API https://developer.chrome.com/docs/extensions/reference/bluetoothLowEnergy/. Retrieved: June, 2021
[19] W3C Geolocation API https://www.w3.org/TR/geolocation/. Retrieved: June,2021
[20] Android class WebView https://developer.android.com/reference/android/webkit/WebView. Retrieved: June, 2021
[21] Test Physical Browser model https://github.com/Anna-Sl/physical-browser. Retrieved: June, 2021
[22] Namiot D. Context-Aware Browsing--A Practical Approach //2012 Sixth International Conference on Next Generation Mobile Applications, Services and Technologies. – IEEE, 2012. – S. 18-23.
[23] Namiot D., Sneps-Sneppe M. Proximity as a service //2012 2nd Baltic Congress on Future Internet Communications. – IEEE, 2012. – S. 199-205.
[24] Kupriyanovsky V. et al. On internet of digital railway //International journal of open information technologies. – 2016. – T. 4. – #. 12. – S. 53-68.
[25] Kuprijanovskij V. P. i dr. Cifrovaja zheleznaja doroga-prognozy, innovacii, proekty //International Journal of Open Information Technologies. – 2016. – T. 4. – #. 9.
[26] Sneps-Sneppe M., Namiot D. About M2M standards and their possible extensions //2012 2nd Baltic Congress on Future Internet Communications. – IEEE, 2012. – S. 187-193.
[27] Kuprijanovskij V. P. i dr. Cifrovaja jekonomika i Internet Veshhej-preodolenie silosa dannyh //International Journal of Open Information Technologies. – 2016. – T. 4. – #. 8. – S. 36-42.