A Case Study of Application Development for Mobile and Location-Based Services

Daniel Granlund, Dan Johansson, Karl Andersson, Robert Brännström
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology
SE-931 87, Skellefteå, Sweden
{daniel.granlund, dan.johansson, karl.andersson, robert.brannstrom}@ltu.se

ABSTRACT

In this paper we present a case study of a location-dependent service, aimed for tourists. Two applications are developed, one based on native technologies, the other on HTML5 and related frameworks. We provide implementation details and compare the different solutions in terms of location support, compliance with required features, and cross-platform functionality. Our experiments show that web-based approaches may lead to significant benefits over using native technologies; both versions of the application displayed comparable location support and compliance with required features, while the web version superseded its native counterpart in cross-platform functionality.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Network services, Location based services

Keywords

Location based services, Applications, User mobility

1. INTRODUCTION

Mobile computing is about providing data access anywhere, anytime [7]. The accessed data can be very heterogenous in its nature, e.g. static textual information, audio or video connections (multimedia sessions), mobile e-services, or context data gathered by GPS or sensors. Location is one of the most important context parameters in mobile computing. Especially relevant is user location. Informing the system about the user's current location can improve the quality of a service, delivering customized solutions to the individual user. [11] Consider e.g. applications offering discounts to users depending on what stores they are near, emergency services, or navigation services that emanates from the user, compared to similar services lacking location support. The former clearly supersedes the latter as it has the potential to deliver a richer service with better precision and actuality. Location-dependent applications along with location-based services (LBS) thus play a central role in our modern computing environment.

There is much to consider when developing LBS for mobile devices. Mobile devices in general have limitations in terms of e.g. screen size, battery, and network interfaces. Also, mobile devices comes in many shapes, manufactured by different vendors, equipped with different hardware and running different operating systems. There are also often big dissimilarities between older and more recent versions of the same operating system, both in terms of functionally and which applications they support. When developing mobile applications, two main alternatives exist regarding implementation technology: either you go for native applications, or you deliver mobile web in the shape of web apps [3]. A native application is developed for a specific platform, i.e. pre-defined device models and a set range of operating system versions. As the platform is known to the developer, the application can be optimized to the device OS and hardware, e.g. built-in cameras, GPS and the OS's standard applications. A web application on the other hand is developed using web technologies, thus accessible from a mobile web browser, at least in theory independent of device model or operating system. A key technology is the emerging HTML5 standard[1] along with related frameworks, providing developers with functionality that was traditionally reserved for applications created with native applications. As the platform is not specified, exploiting or even accessing hardware functionality is generally harder for a web application compared to a native counterpart.

In this paper we present a case study of a LBS. Two applications are developed, one based on native technologies, the other on HTML5 and related frameworks. We provide implementation details and compare the different solutions in terms of location support, compliance with required features, and cross-platform functionality. Our intention is to examine how well state-of-the-art web technologies can match (or supersede) native approaches when it comes to the development of location-dependent applications.

2. RELATED WORK

Examples of LBS implementations include GeoHealth [4], an interactive service for supporting distributed mobile collaboration within the healthcare area. Relying on web tech-

[1]http://www.w3.org/html/wg/drafts/html/master/

nology, GPS positioning and Google Maps, GeoHealth delivers ad-hoc exchange of work tasks, prompt alarm functionality and information tailored to the user and the location. Person Wide Web [12] is a location-based web service architecture that recognizes web resources based on the user's geographical position. The architecture is evaluated through an Android application, displaying performance on service establishment and message delivery time. Wetzel, Blum and Oppermann [16] present a location-based game called Tidy City, allowing inhabitants and tourists to explore a city by the use of an location-dependent application. A hybrid approach of both web-based interfaces and mobile tools are used in creating service. The Lewis system [9] realizes personalized LBS delivery and web service utilization. When using Lewis while on the move, the system can recommend nearby spots that the user might be interested in and include social-network-based recommendations. Kenteris, Gavalas and Economou [8] address mobile tourism, the use of mobile devices as tourist guides. Their prototype is portable, has rich content, and matches user preferences. However, it lacks full cross-platform support.

As shown, the importance of mobile applications for LBS cannot be underestimated. There are two main approaches to mobile application development [13] [2]: either the application is built using a native programming language, or the application is based on web technology. Some general properties of native vs. web are often emphasized. Native apps are created for specific devices. Such apps can make use of the hardware capabilities of the targeted device (e.g. camera and operating system features), and adheres to the device-specific design conventions. The native app is typically downloaded from a digital marketplace, managed by the provider of the operative system, e.g. Apple's AppStore or Google's Google Play. The web app on the other hand is basically a web site, optimized for mobile web browser access in terms of e.g. screen size, cellular network speed and mobile navigation controls.

Concerning advantages and disadvantages of native and web applications, some general pros and cons are often emphasized [2] [6]. Due to the fact that the native app is designed for a specific operating system, it often has full access to device functionality like GPS, camera etc. In addition, the developer of a native app has greater control of the user interface design. Speed and efficiency also distinguishes the native app, although the gap is closing. Some go as far as denoting the difference in performance as negligible or unnoticeable unless the application has heavy reliance on e.g. image processing or 3D gaming [3]. Further, the digital marketplaces give native apps an advantage when it comes to findability and exposure. Although web apps can be deployed anywhere on the Internet, the user needs to know the specific URLs to get access. Another area where the native app traditionally has been outshining its web counterpart is offline usage. Web apps in general are more dependent on Internet access, while native apps tend to save data locally. Web applications can however be very convenient, both for the developer and the user. Web technology can be used cross-platform, allowing the developer to create only one version of the web app, whereas native apps must be developed for many different devices and operating systems. Web app development thus has the potential of being both fast and cheap. An update in the web app will immediately go through, whereas the user of a native app actively has to update the app to a new version. Regarding revenues, the digital marketplace demands a share of the app price. As no marketplace is needed for the web app, all revenues (if any) go to the provider of the application.

These general properties of native and web applications are also tested in many recent related work. Huy and van Thanh [6] conduct a theoretical evaluation of the different mobile app paradigms, covering the developer, user, and service provider viewpoints respectively. Results are that developers and service providers seem to gain from building web apps, while better user experience in general tips the scale in favor of the native app paradigm in a user perspective. The differences are however very small. An expert evaluation by Anttonen et al [1] match ongoing standardisation work and emerging technologies against prior challenges and shortcomings of web applications, describing how state-of-the art web technology solves many of the problems regarding user interaction, performance, security, compatibility, development, testing, and deployment. Fahlesson and Johansson [5] use simplified cognitive walkthrough to compare strengths and drawbacks of native versus web applications with emphasis on cross-platform functionality. A case study is conducted, recreating the look-and-feel of a native application with web technology and proving functionality in nine different testbeds. All these projects show that the gap between native and web has been lessened, and that application development with rich features using non-proprietary web technologies is no longer impossible.

3. METHOD

To compare how well state-of-the-art web technologies can match native approaches when it comes to the development of location-dependent applications, we conducted a case study. The main goal of a case study is to deliver a holistic description of the application usage within the real world setting [10]. The description can be used to draw conclusions from the specific case, discussing the possibility of applying these to a particular domain, i.e. application development for mobile and LBS. This type of interpretive research approach is discussed by e.g. Walsham [15].

As a part of an ongoing research project called TG4NP[2], an idea arose to implement a mobile application that would allow visitors at a tourist center in Gammelstad, Luleå in northern Sweden to take part of outdoors tourist attractions, even when there is no staff around. The Gammelstad site is an old church town that dates back to the 14th century and holds a number of attractions within walking distance. The tourist center is typically manned during daytime, and guided tours are given periodically in several languages.

According to the owners of the tourist center, many visitors might arrive in the evening when there is no service available. One of the goals of the application is to provide an experience as close as possible to a guided tour, solely by using technology available in a mobile device such as a smartphone. Another desired functionality from the project owner is that the app should provide directions to various public facilities such as restrooms, parking lots, and restaurants. The requested functionality included a detailed map where points of interest, POIs, are clearly marked. When a user approaches a certain POI, textual information about the point will be displayed, possibly along with a audio

[2]http://www.tg4np.eu/

recording and/or images. By providing a poster outside the tourist center with short information about the app and how to use it, along with a scannable QR code that allows the user to load the application, it will be easy to access for any smartphone-equipped visitor.

To observe the effects of our design decisions, we used a comparative prototyping research methodology [14]. The basic idea behind this methodology is to conduct a comparative evaluation similar to a laboratory experiment, but in a relatively unconstrained, real world use case. Our evaluation was carried out in three steps: 1) *Design with alternatives.* Here we designed two prototypes with the same basic functionality, but with varying design. In our case, we made one prototype based on native programming technology (an Android application), and also a prototype based on state-of-the-art web technology (HTML5 and related frameworks). The following requirements where decided upon for both versions of the application: Location representation through a Point of Interest (POI) map; Location-aware information (e.g. locations to be marked as already visited); Dynamically loaded content, easy to maintain for non-technical staff; Support for displaying content in different layers (e.g. filter out restaurants and restrooms); Support for multimodal media (e.g. text, links, audio, video).

The second step was 2) *Vary deployment situations.* As we had a clear case, we focused on as similar deployment situations a possible, making the final step, 3) *Compare and contrast*, easier for us to analyze. During the third step, qualitative (and some complementary quantitative) data was analyzed and used to make a comparison between design alternatives, in this case native vs. web.

Our alteration of the methodology's second step was done deliberately. We wanted both applications to be evaluated under the same conditions, so that any detected dissimilarities in app functionality would not be biased by externals. A variation of deployment situations would require parallel experiments with both versions of the app being tested in every chosen deployment situation.

4. RESULTS

The main idea behind the software architecture was to make a clear distinction between content and functionality. The actual informational contents provided by the app could be expressed in an easily editable format and adapted to suite a large variety of applications. We chose Keyhole Markup Language (KML), whose file format is based on XML. It provides human readable text files which are easy to understand and edit. Furthermore, the format allows a content provider to create different layers of informational objects which makes abstraction of undesired information easy. The KML file format allows geographical information objects such as placemarks or arbitrary polygons expressed with geographical coordinates to be complemented with HTML coded contents such as text, images and embedded media. An example of this might be that objects of general character, such as restaurants and restrooms, might be hidden from the map until the user wants to display them. Thus they will not interfere with the object of main interest. KML files can either be edited directly in a text editor, or created using a specialized editor where objects are drawn on a map and additional contents are added in an integrated HTML editor. Storing the KML file on a central server allows for easy updates and dynamic handling of contents.

The KML file contains Folder, Placemark, Name, Point and (optional) Description elements. Folder elements are used to create layers of placemarks on the map. Each Placemark element contains a name element and a Point which indicates the position. Furthermore, a description element can be added to provide contents for a popup window that displays rich HTML based content for the user. The themes element is used to create sub-groups of information items that can be displayed when clicked in the popup window.

When it comes to the basic functionality of the app, a set of basic features are provided. These include a main screen with a map view component that displays information from a map provider and the chosen KML file information along with the users own position. The map view screen is based on the Leaflet API[3] which is a javascript based API that is used to display maps and various overlays. Leaflet allows for a large variety of map providers, including Google Maps[4], Bing Maps[5], Open Street Maps[6] etc. Also, map material from an image or vector based file can be used as a map source. An automated function keeps track and indicates which locations are already visited by greying that object on the screen. This way, it is easy to keep track of visited locations and helps guiding the users between different locations. The information on visited locations is stored on a session basis and reset whenever the app is reloaded. An automated survey functionality is provided that allows the app to automatically remind the user to fill out a survey to give feedback on the experience when leaving a location. By measuring the distance between the user and a certain location and comparing it to a configurable distance value, the application can detect if a user is leaving the location. The survey is provided as a digital form, accessible through the app.

Since it cannot be assumed that a user has continuous network access, an offline function has been implemented. It allows the app to download the KML file along with a map and all other information required to run the app and cache it locally in the device. This way it is possible to e.g. cache the app at a WiFi hotspot and then run it in offline mode if cellular coverage is low or service cost is expensive. Whenever network access is detected, the local cache with data is verified against the most recent data on the server and updated if necessary. If the provided content is large in terms of data storage, it is possible for an administrator to specify which content is the most relevant, and only allow for that specific subset of data to be cached for offline use. Some map providers (e.g. Google Maps) does not allow caching and offline usage of map material. Depending on the choice of map provider, there might a need for a specific offline map, possibly less detailed.

4.1 The Native App

The native app is implemented using the Android platform and centered around the Google Maps API, which is distributed as part of the Google Play services SDK[7]. The app is thereby limited to using only one map provider. The map component is configured in full-screen mode on the de-

[3]http://leafletjs.com/
[4]https://maps.google.se/
[5]http://www.bing.com/maps/
[6]http://www.openstreetmap.org/
[7]http://developer.android.com/google/play-services/index.html

vice, and all additional content is shown on top of the map. Content, in the form of a KML-file, images, and sound clips, are stored on a web server. When the app is loaded, the KML-file is downloaded from the server and parsed using a customized parser. Placemarks are drawn as icons on an overlay on the map. By using the menu button, a dialog can be shown that allows a user to choose which layers of POIs to show on the map and each layer of icons is color coded. If an internal GPS is present in the device, an icon is shown in the center of the image to indicate the user's own location. If a POI is within 50 m, or if a POI placemark is clicked by the user, a popup window is shown that displays the HTML-coded information from the KML-file, including images etc. that corresponds to the chosen placemark. When a POI is visited, the icon is redrawn in a semi-transparent way to provide a clear indication of visited points. Menus and dialogs used in the app are general components provided by the Android SDK. In order to support offline-use of the app, the KML-file along with all referenced textual material is downloaded and stored in a local storage. If the app is started when there is no Internet connection available, the data is loaded from the local storage instead. In order to provide a map, even for offline use, a map image with limited coverage can be stored in the local cache and used as an overlay until online map data is available.

4.2 The HTML5 Web App

The HTML5 version of the app has many similarities with the native one. The first major difference is that it is based on code that is HTML5-compliant, enabling it to execute on a large variety of devices. Since the application is intended to run in a web browser with JavaScript support, a large number of map API solutions are available. The open source Leaflet API is chosen since it is specifically designed to support mobile devices. It allows for a variety of map providers, comprehensive support for zooming, panning and dragging, and it is well documented for developers. Also, Leaflet API provides a built-in set of controls, such as markers and popups. However, it was determined during development that the built-in controls did not offer the desired level of flexibility so custom versions were implemented to suit the specific application. Since the popups are likely to contain more than just a few lines of text, there were problems with scrolling and layout for some devices. It turned out to be quite challenging to achieve the same layout and basic functionality on many different platforms and custom handling was required in some cases, especially on older devices. A KML file of the same format as the one used in the native app is loaded from a web server and displayed using color coded markers. By clicking on an icon in the top right corner, a list is shown that allows the user to choose between layers of interest. The Geolocation API[8] is used to retrieve the users own location and functionality similar to that of the native app is implemented that allows for automatic popups when the user approaches a POI. The app uses the HTML5 session storage functionality to keep track of visited locations. To support the offline mode, the app uses the HTML5 application cache which allows for local caching of web contents that can be used for offline browsing. The caching is controlled by a manifest file on the server side. In our application the manifest is generated dynamically depending on which files that need to be cached. The application uses a file checksum in order to determine which files need to be updated in the cache. Also, by controlling which files are cached, offline map data can be downloaded from a provider that allows local caching of map tiles. Finally, an additional functionality is implemented that allows the app to detect when a user has left the tourist site, and then automatically open up a survey where the user can provide feedback on the experience.

[8]http://www.w3.org/TR/geolocation-API/

4.3 Comparative Evaluation

The two prototypes were designed to support the same basic functionality. As the prototypes were developed on different occasions by different programmers, properties such as graphical design and additional features did differ. In our comparative evaluation we looked solely to the requirements set up for this particular case study, i.e. location support, compliance with required features (see detailed description under section III), and cross-platform functionality. The differences in e.g. GUIs and extra features between the two versions were not studied, and thus such differences have not affected our evaluation in any significant way.

When it comes to location support, both versions of the app have comparable functionality. The POI map is implemented in both apps, allowing geographical location representation of important objects. The information is location-aware, e.g. locations already visited by the user are marked with a certain color. Layers are used to e.g. filter out restaurants and restrooms from the POIs. Differences between the app versions are negligible as our evaluation shows that differences in location support and GPS precision comes from hardware.

Compliance with other required features are also equal between the two different versions of the app. Both support dynamically loaded content, that is easy to maintain for non-technical staff. There is also equal support for multimodal media (e.g. text, links, audio, and video in the POI descriptions).

When it comes to cross-platform functionality and backward compatibility, the web application supersedes the native version. The web application was evaluated on a wide range of devices and operating systems (e.g. iPhone, iPad, different Android smartphones and Android tablets), and browsers (Safari, Android Browser, Chrome, Opera) of different versions. Some difficulties with direction were detected using the Opera browser, while Android versions earlier than 4.0 had some problems providing proper scroll/touch functionality, but otherwise cross-platform functionality was pervading. Additional evaluations were also carried out using a Windows phone, where lack of cross-platform functionality was found in Internet Explorer version 6.0, this due to limited HTML5 support.

Summarizing, the application's native and web versions both displayed comparable location support and the same compliance with required features. The web version however superseded its native counterpart in cross-platform functionality, making it possible to run the same application on the most commonly used mobile browsers and platforms.

5. DISCUSSION

The tremendous growth in terms of both device and network deployments globally have given application developers very efficient tools for providing mobile and LBS to end-users. On one hand, the ecosystems for mobile apps provided by Apple, Google, and Microsoft have really made the software development and deployment easy managed. On the other hand, modern web technologies directed towards mobile and LBS are rapidly gaining interest from large communities of developers. When it comes to free of charge, location-dependent applications intended to be used on site, there is little need for a marketplace. As long as the users (in our case the visitors) are provided with an URL to the app, they can easily access and start to use the application on site.

As shown in this paper, the usage of web-based approaches may lead to significant benefits over using native technologies. Reaching all users from one single implementation and also not having to deal with deployment issues related to some of the ecosystems involved are the two most notable merits of that approach. The drawbacks include having to deal with a technology not being fully mature and still lacking full support from all browsers on the market.

One interesting feature in our web-based approach was that the parsing of the KML file was carried out on the client side, using JavaScript technology. This made the web-based implementation equally rich and less dependent of constant network connectivity.

6. CONCLUSION AND FUTURE WORK

We have presented a case study of a location-dependent service, aimed for visitors to a site with various tourist attractions. Two applications were developed; one based on native technologies, the other on HTML5 and related frameworks. We provided implementation details and compared the different solutions in terms of location support, compliance with required features, and cross-platform functionality. Our evaluation showed that the application's native and web versions both displayed comparable location support and the same compliance with required features. The web version however superseded its native counterpart in cross-platform functionality, making it possible to run the same application on the most commonly used mobile browsers and platforms. We conclude that web-based approaches for LBS development can lead to significant benefits over using native technologies.

Future work includes carrying out a comprehensive user evaluation on site with a mixed set of test subjects. Also, future plans include developing a graphical editor for creating KML content. A user friendly editor environment will be developed that allows geographic markers to be drawn on a map, and a simple HTML editor will be integrated for creating the popup window contents.

7. ACKNOWLEDGMENTS

We want to thank the people involved in the TG4NP research project, our students working with prototyping, and the users that participated in the evaluation process.

8. REFERENCES

[1] M. Anttonen, A. Salminen, T. Mikkonen, and A. Taivalsaari. Transforming the web into a real application platform: new technologies, emerging trends and missing pieces. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 800–807, New York, NY, USA, 2011. ACM.

[2] K. Buettner and A. Simmons. Mobile web and native apps: How one team found a happy medium. In A. Marcus, editor, *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, volume 6769 of *Lecture Notes in Computer Science*, pages 549–554. Springer Berlin / Heidelberg, 2011.

[3] A. Charland and B. Leroux. Mobile application development: web vs. native. *Commun. ACM*, 54(5):49–53, May 2011.

[4] C. M. Christensen, J. Kjeldskov, and K. K. Rasmussen. Geohealth: a location-based service for nomadic home healthcare workers. In *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, OZCHI '07, pages 273–281, New York, NY, USA, 2007. ACM.

[5] S. Fahlesson and D. Johansson. A case study of cross-platform web application capability. In *Future Network and MobileSummit 2013 Conference Proceedings*, FNaMS, 2013.

[6] N. P. Huy and D. van Thanh. Evaluation of mobile app paradigms. In *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia*, MoMM '12, pages 25–30, New York, NY, USA, 2012. ACM.

[7] S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Comput. Surv.*, 42(3):12:1–12:73, Mar. 2010.

[8] M. Kenteris, D. Gavalas, and D. Economou. An innovative mobile electronic tourist guide application. *Personal Ubiquitous Comput.*, 13(2):103–118, Feb. 2009.

[9] S.-P. Ma, W.-T. Lee, and C.-H. Kuo. Location explorer with information services: A mobile application to deliver location-based web services. In *Next-Generation Electronics (ISNE), 2013 IEEE International Symposium on*, pages 283–286, 2013.

[10] M. Q. Patton. *Qualitative research & evaluation methods.* SAGE, London, 4. ed. edition, 2004.

[11] J. Schiller and A. Voisard. *Location Based Services.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[12] S. Shin, P. Kim, Y. Yoon, S. Eun, and H. Yoon. Person wide web: Active location based web service architecture using wireless infrastructure. In *TENCON 2010 - 2010 IEEE Region 10 Conference*, pages 2024–2029, 2010.

[13] S. Tarkoma and J. Kangasharju. *Mobile middleware : architecture, patterns and practice.* Wiley, Chichester, U.K., 1. ed. edition, 2009.

[14] J. Trevor and D. M. Hilbert. A comparative prototype research methodology. Technical report, FX Palo Alto Laboratory, Inc. (FXPAL), 2004.

[15] G. Walsham. Interpretive case studies in is research: nature and method. *European Journal of Information Systems*, (4):74–81, 1995.

[16] R. Wetzel, L. Blum, and L. Oppermann. Tidy city: a location-based game supported by in-situ and web-based authoring tools to enable user-created content. In *Proceedings of the International Conference on the Foundations of Digital Games*, FDG '12, pages 238–241, New York, NY, USA, 2012. ACM.