XD-Bike: A Cross-Device Repository of Mountain Biking Routes

Maria Husmann, Linda Di Geronimo, and Moira C. Norrie

Department of Computer Science, ETH Zurich CH-8092 Zurich, Switzerland,
{husmann|lindad|norrie}@inf.ethz.ch

Abstract. Despite the high level of interest in cross-device interaction, only few fully functional example applications exist. With XD-Bike, we built one of our own to not only showcase the use of our cross-device framework but also gather insights into the development process. XD-Bike is an application for mountain bikers that is built with web technologies and adapts to the set of devices at hand. The user interface is distributed across all available devices taking into account the space requirements of the UI elements, their importance, and the available space on devices. While using the cross-device framework eased the development process, we found the testing and debugging challenging due to the distributed nature of the application and the large set of possible device combinations.

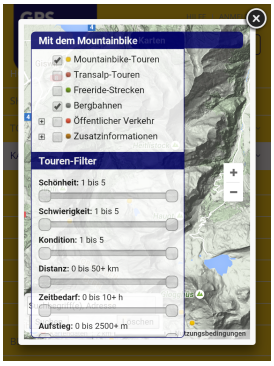
Keywords: distributed user interface, cross-device, liquid applications, web

1 Introduction

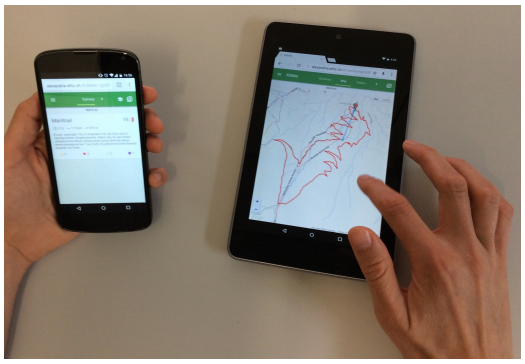
Cross-device or liquid applications [1] that adapt to the set of devices at hand have received a lot of attention from the research community in recent years. The user interfaces of such applications spread across multiple devices and interaction with one device affects paired devices. Frameworks [2, 3] and design tools [4, 5] have been built to facilitate the development, but few example applications exist. Our goal was to build a complete example application as a showcase for our own web-based framework XD-MVC[1] which is based on the MVC pattern [6]. Furthermore, we wanted to explore the development process of cross-device applications to find difficulties and gather requirements for better tools. As an application domain, we chose a tour repository for mountain bikers and named the project XD-Bike.

Planning a mountain bike tour can be challenging. For the tour to be successful and fun, many factors come into play. The technical difficulty should not be too high (frustrating) or too low (boring). It should neither be too short nor too long. The same goes for the elevation gain. Not only is the total elevation gain relevant, but also the grade of the ascent - too steep and it is unridable. Last

[1] https://github.com/mhusm/XD-MVC



(a) GPS-Tracks.com on a Nexus 4 device. The filter occludes the map and can not be hidden.



(b) XD-Bike on a Nexus 4 and a Nexus 7 device showing a map and corresponding summary. The map is shown on the larger device.

Fig. 1. An existing mountain bike repository and XD-Bike.

but not least, the location of the tour is an important factor. Even though most riders will be faced with these questions and challenges during planning, there is no one-size-fits all solution. Technical skills and endurance vary among bikers. Also, the form on the day of the riders and weather conditions may require plans to be changed at short notice.

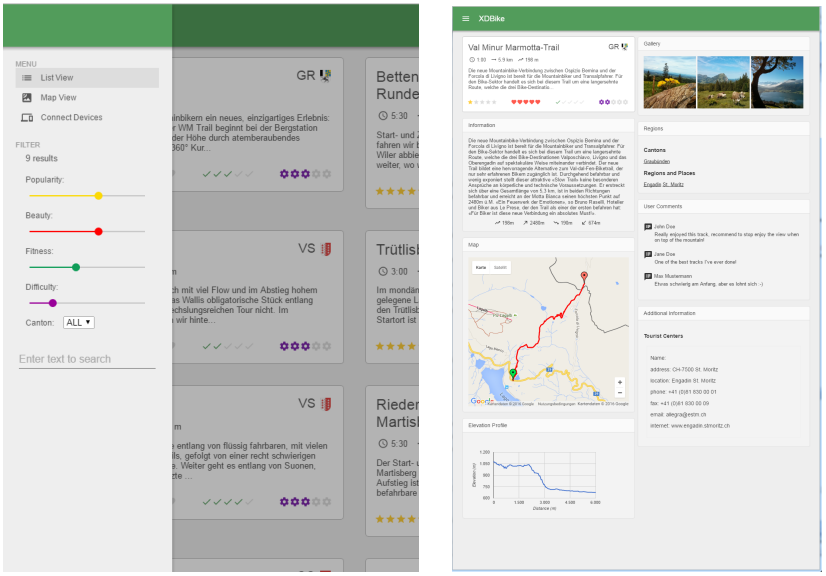
These challenges have been recognised and there are books and magazines that propose routes, special bike maps, and online route repositories that can help bikers plan their tours. In terms of online repositories for Switzerland, two popular examples are Mountainbikeland[2] and GPS-Tracks.com[3]. Both offer a fully featured desktop website and an app for mobile devices. The apps offer only a reduced set of features compared to the website. GPS-Tracks.com has recently been changed to a responsive design, but its use on small screens is still problematic (Fig. 1(a)). It certainly is challenging to present such a wealth of information with limited screen real estate.

Yet, a common scenario is that a couple of riders spend several days in a hotel in the mountains with limited equipment. To keep luggage light, notebooks are often left at home and using just the apps may be inconvenient. However, typically each rider will have at least a smartphone. In addition, one or multiple tablets might be present and a modern hotel may offer a smart TV in the rooms. This results in a scenario where a cross-device application could be beneficial to the users. There is the need to display a lot of information and individually none of the the devices are well suited. The phones have small screens and the TV, if present at all, typically has limited input capabilities. But taken together, these devices offer increased screen real estate and input capabilities that XD-Bike takes advantage of by distributing its user interface across all of them (Fig. 1(b)).

[2] http://www.mountainbikeland.ch
[3] http://gps-tracks.com/

2 Design

XD-Bike contains a set of mountain bike routes that can be accessed from either a list view or a map. For both views, a filter is available that allows the tracks to be selected according to a set of criteria including difficulty, location, beauty, and keywords (Fig. 2(a)). There is a detailed view for each track where additional information such as a textual description, a map, elevation profile, and pictures are displayed (Fig. 2(b)).



(a) The list view with the filter.

(b) The detailed view on a single device.

Fig. 2. XD-Bike design

Multiple devices can be paired and used in combination thereafter. The pairing is done by either sharing a URL, for example using Android Beam[4] or an instant messenger, or by manually entering the ID of a device. Figure 3 shows the pairing view where all connected devices are listed. A number in the top toolbar that is constantly displayed shows how many devices are currently paired. This provides feedback to the user and lets them verify if the pairing process was successful or could indicate if a device is disconnected. Once the devices are paired, the first device is assigned the role of a *controller* and all other devices are *viewers*. Only the controller can select a route for which details will be displayed on all paired devices.

The detailed view is made up of multiple tiles or *fragments*. Each fragment contains a coherent unit of information for a chosen track, for example a map

[4] https://support.google.com/nexus/answer/2781895?hl=en

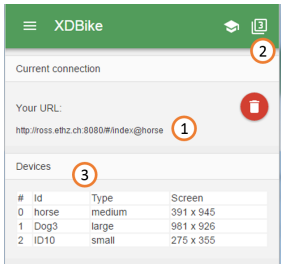


Fig. 3. The pairing menu, showing the pairing URL (1), the number of paired devices including this one (2), and a list of all paired devices (3).

or an elevation profile. When multiple devices are paired, these fragments are distributed across the devices. The distribution is determined by an algorithm that takes into account the available space on the device, the space requirements of the fragment and the priority of the fragment. For example, the map fragment needs the most space and has a high priority. The track summary has a low space requirement but the highest priority, while the additional info fragment has both a low space requirement and low priority. Devices can display one or multiple fragments depending on their size. In a first step, we calculate how many fragments can be displayed over all devices. We then sort the fragments by priority and discard those with the lowest priority that would not fit on any device. In the next step, the remaining fragments are distributed across all devices giving larger fragments the bigger slots. This computation is done client-side on each client. To ensure that the algorithm is deterministic and that each device reaches the same global distribution, unique device IDs assigned by XD-MVC are used to achieve a total order.

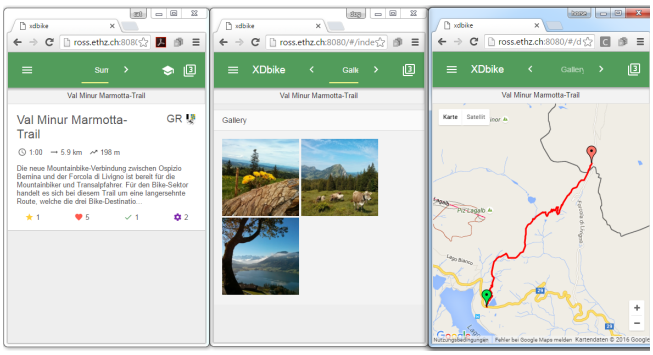


Fig. 4. XD-Bike on three small devices. Each device displays one fragment. The left-most device is the controller which is indicated by an icon at the top right next to the number of connected devices.

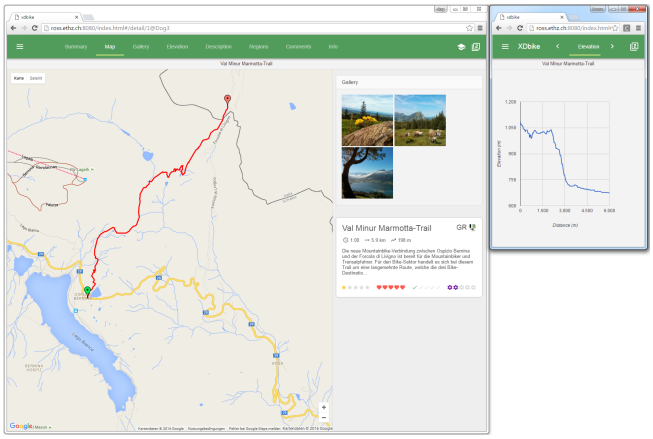


Fig. 5. XD-Bike on a small and a large device. The large device shows a large fragment (map) and two smaller ones.

Figure 1(b) shows the application distributed across two small devices. Each device displays one fragment. The summary and the maps have the highest priority and are thus displayed. The map requires more space than the summary. Consequently, the map is displayed on the larger of the two devices. Similarly, Figure 4 shows the interface distributed across three small devices and Figure 5 across a large and a small device. The large device displays three fragments with the maps assigned to the largest space. Each time a new device is paired, the system re-evaluates the available space and re-distributes the fragments. On each device, the user can manually override the distribution and choose a fragment to be displayed on the device.

On the controller device, routes can be selected from the list view, but navigating back and forth between the list and the detail view could be tedious. To make route selection easier, we also implemented tilt-and-tap interactions [7]. To go the next or previous route on the list, the user can quickly tilt the controller device to the right or to the left.

3 Implementation

XD-Bike was implemented as a semester project by three master students using our own cross-device framework XD-MVC and the Polymer[5] library. XD-MVC handles the device to device communication and data synchronisation among the paired devices. To ensure all paired devices show information for the same route, the currently selected route is configured to be synchronised. The distribution algorithm receives information about the devices from XD-MVC. The framework provides the size of each paired device. In addition, the framework also supports roles that we used to differentiate controller and viewer devices.

[5] https://www.polymer-project.org/

4 Discussion and Conclusion

The project was limited by the duration of the semester. Given more time, further cross-device functionality could be implemented. For example, a collaborative mode could be introduced where all devices (not just the controller) can select routes. This would then introduce potential conflicts when two users try to change the route at the same time. Another extension could be to not only distribute the detail view, but the whole application. For example, the filter could be moved to a small device and the map showing the results to a larger one. How the users can be informed of the possibilities available in a cross-device application and how they can use all their devices also remains a question that could be addressed in the future.

The project showed that having a cross-device framework facilitates the development. Most of the time was spent on the general implementation and only a small part was spent on integrating with the cross-device framework. At the same time, our experience showed that developing a cross-device application is still challenging. As the application is web-based, browser tools were used to debug the application and multiple browser windows simulated multiple devices. Occasionally, we also tested the application on real mobiles and tablets. This process is rather tedious and devices have to be paired again every time the application is reloaded. This has sparked us to explore better tools for testing and debugging cross-device applications [8].

Acknowledgements. This project was supported by grant No. 150189 of the Swiss National Science Foundation (SNF). We would like to thank Dhivyabharathi Ramasamy, Alexander Richter, and Marko Zivkovic for their contributions to this project.

References

1. Mikkonen, T., Systä, K., Pautasso, C.: Towards Liquid Web Applications. In: Proc. ICWE. (2015)
2. Yang, J., Wigdor, D.: Panelrama: Enabling Easy Specification of Cross-Device Web Applications. In: Proc. CHI. (2014)
3. Badam, S.K., Elmqvist, N.: PolyChrome: A Cross-Device Framework for Collaborative Web Visualization. In: Proc. ITS. (2014)
4. Nebeling, M., Mintsi, T., Husmann, M., Norrie, M.C.: Interactive Development of Cross-Device User Interfaces. In: Proc. CHI. (2014)
5. Husmann, M., Nebeling, M., Pongelli, S., Norrie, M.C.: MultiMasher: Providing Architectural Support and Visual Tools for Multi-Device Mashups. In: Proc. WISE. (2014)
6. Krasner, G.E., Pope, S.T.: A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80. J. Object Oriented Program. (1988)
7. Di-Geronimo, L., Aras, E., Norrie, M.: Tilt-and-tap: Framework to support motion-based web interaction techniques. In: Proc. ICWE. (2015)
8. Husmann, M., Heyder, N., Norrie, M.C.: Is a Framework Enough? Cross-Device Testing and Debugging. In: Proc. EICS. (2016)