HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

IFS INSTITUTE FOR SOFTWARE

Bachelor Thesis, Institute for Software

# BLINQ BookingTool

University of Applied Sciences Rapperswil

Spring Semester 2016
17.06.16

| | |
|---|---|
| *Supervisor:* | Prof. Dr. Farhad Mehta, IFS |
| *External Examiner:* | Vikram Kriplaney |
| *Auditor:* | Prof. Dr. Peter Heinzmann |
| *Authors:* | Benny Gächter & Benjamin Wilhelmin |
| *Partner:* | Jan Berchtold, BLINQ AG |
| *Duration:* | 22.02.2016 - 17.06.2016 |
| *Extent of work:* | 360 Hours, 12 ECTS per student |

# Abstract

The BLINQ app developed by BLINQ AG in Zurich Switzerland is a dating app, that uses Bluetooth beacon technology to locate its users. These beacons are distributed mainly in stores, bars, and restaurants in the Zurich area. They allow the accurate localization of BLINQ users even within enclosed spaces.

This beacon infrastructure can not just be used for the dating app. It is beneficial for precise ad targeting. The goal of this thesis is to use BLINQ's beacons to provide a platform, where advertisers can book ads, which are displayed in BLINQ's dating app and are triggered by the proximity of beacons. This allows a more precise and efficient targeting of customers.

The project is realised using agile software development methods. After a thorough requirement analysis, mock-ups have been created and implemented while continuously receiving feedback from BLINQ.

To fit into the existing Infrastructure the BookingTool is implemented in Python using the Django framework. The BookingTool uses the Model-view-controller architectural pattern. Parts of the model will be stored remote either on BLINQ's servers or on BLINQ's ad publishing network Google DoubleClick for Publishers.

The goal of this thesis is to allow advertisers to book advertisements that are triggered if a user has a beacon check-in. A check-in means that the user is within the reach of a beacon. The BookingTool provides an overview with statistics of past, running an future ad campaigns. An administration interface will provide an overview of all companies and advertisement campaigns in the tool. The BookingTool will provide BLINQ a source of income.

## Declaration of Authorship

We declare that this bachelor thesis and the work presented in it was done by ourselves and without any assistance, except what was agreed with the supervisor. All consulted sources are clearly mentioned and cited correctly. No copyright-protected materials are used in this work without permission of the respective copyright holders. The LaTeX source code for this document is based on "HSR-LaTeX-Template" by Florian Bentele.

Rapperswil, 17.06.16 — Place and date

B. [signature] — Benny Gächter

Rapperswil, 17.06.16 — Place and date

[signature] — Benjamin Wilhelmin

# Management Summary

The main goal of this bachelor thesis was to develop a platform for location-based advertising. Advertisers are able to book location-based advertisements. These advertisements are delivered to the user's smartphone while he is in the proximity of specified beacons. The BookingTool is accessible via a standard web browser running on a computer or on a smart phone. It provides advertisers an overview of their campaigns and for the provider an overview of all advertisers and campaigns. Additionally, the BookingTool provides various overviews that are helpful to advertisers and the provider.

## Motivation

BLINQ has built a dating app and has distributed several hundred beacons, mainly in the Zurich area. This tool will make use of BLINQ's beacon infrastructure and provide BLINQ a way to generate income. Since the BLINQ app is free of charge, BLINQ depends on advertisements for funding. The BookingTool provides an easy and convenient way for advertisers to create campaigns for the BLINQ app.

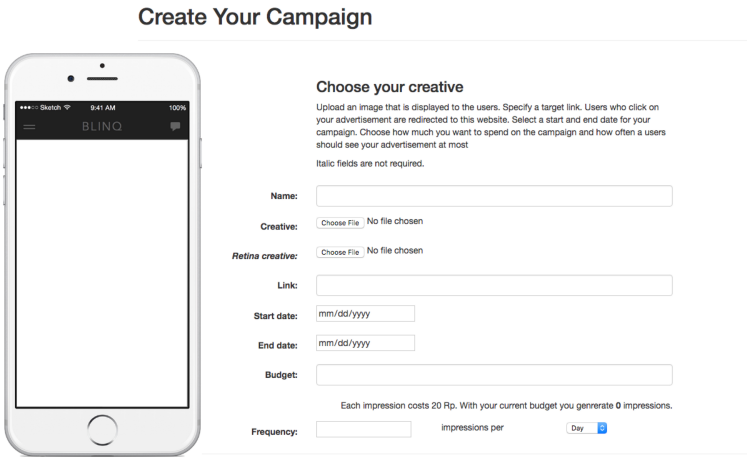


Figure 1: From to book an advertisement with ad preview

## Goals

The BookingTool provides an easy way to create and manage advertisement campaigns for the BLINQ dating app. The BookingTool is accessible via web browser on the computer or smartphone. BLINQ shoud have an overview of all registered companies as well as their booked campaigns. Because the BLINQ app is international, the BookingTool provides features for internationalization.

## Results

The BookingTool is an easily deployable web application that serves as an easy to use interface between advertisers, provider and a number of third party services. The BookingTool provides a robust and easy-to-use platform for both advertisers and provider.

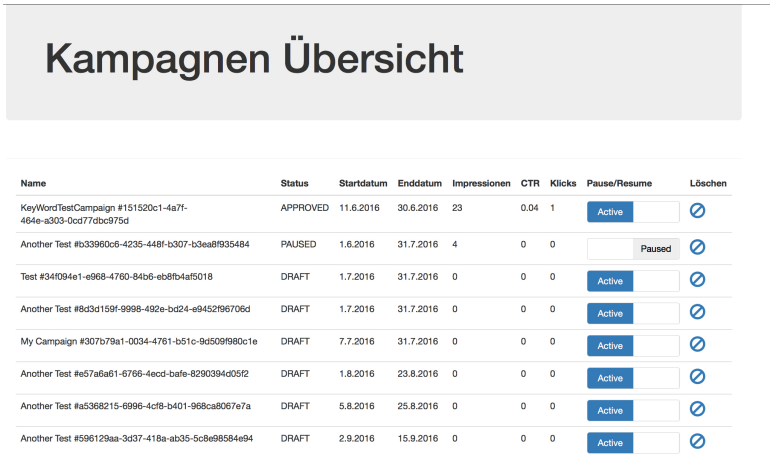


Figure 2: Campaign overview as seen by the advertiser

This screenshot shows the campaign overview in a browser with German language setting. The BookingTool reads the language setting from the browser and displays the right translation. If no translation is found it chooses the default language English.

# Contents

# Analysis

The BLINQ AG in Zurich, Switzerland has roughly 350 beacons deployed. They are used to locate users of their app. BLINQ wants to use this beacons to allow advertisers to book location based advertisements.

## 1.1 Current state

BLINQ uses the Google DoubleClick for Publishers (DFP) platform to deliver advertisements to their app. DFP does not provide beacon targeting support. For this reason BLINQ has its own ad serving infrastructure which is linked to DFP.

So far BLINQ had to create an ad campaign on both systems manually. This solution does not scale and is error prone since the same data has to be entered multiple times in different locations.

## 1.2 Target state

The process of booking and advertisement for the BLINQ app should be automated and simplified. BLINQ wants to provide a platform, called BookingTool where advertisers can book their campaigns by themselves.

The BookingTool allows advertisers of BLINQ to directly book ads on the BLINQ app without the need to get in contact with the BLINQ marketing team. This reduces the effort for BLINQ and makes it easier for advertisers to launch campaigns. After a campaign is booked it has to be approved by BLINQ. This is a security measure to prevent malicious advertisers from propagating inappropriate content.

## 1.3 Task description

BLINQ provided the following task description written as user stories:

**As an advertiser I want to...**

Website

1. See a starting page where the basic concept of the booking tool is explained so I am aware of the advantages
2. See the pricing per TKP to know how much a campaign is going to cost me
3. Use the whole booking tool on a mobile device
4. See the contact details for BLINQ to be able to ask administrative questions

Account

5. Create an account with my email and password to login as well as the fields firstname, lastname, company, street, zip, city and phone number so I can later book campaigns that are assigned to me
6. Login to my account to have an overview of my current campaigns
7. See a list of my current campaigns with campaign name, state, goal, impressions served, clicks, CTR, targeting type, start date, end date
8. Change my account settings as email and password in case I change my email or want to set a new password for security reasons
9. Reset my password if I forgot it so I can access my account again

Campaign

10. Be able to book ad campaigns that are shown on the BLINQ app to advertise my product or company
11. Upload a custom image in the exact size of 300 x 250 px as a creative for my campaign to promote my product or service to the BLINQ users
12. Upload a second image with twice the size (600 x 500px) so users can see my ad in retina resolution on the BLINQ app
13. Set the campaign details start date, end date, frequency cap

14. Set a target URL of my campaign so the user who clicks the ad is redirected to my website

15. Select if BLINQ should send me a programmed Beacon to my company address

16. Set the targeting type of my campaign to either Beacon targeting or area targeting

17. See a list of my Beacons so I can select a Beacon that I already placed at my venue for the Beacon targeting

18. Serve ads only to users who were detected at my beacon if the type is set to Beacon targeting

19. Set a position on a map and set a range for the area targeting so ads are only served to users who were detected at another Beacon or had a GPS checkin in the range

**As an admin I want to…**

Campaign

20. Be notified with an email if a new campaign has been booked so I can check the content and approve it

21. Be notified with an email if an advertiser ordered a Beacon so I can program the Beacon and send it

22. See a list of all campaigns with their respective advertiser on the admin panel to have a good overview of what is running at the moment

In addition the following points have to be covered by this project as well:

- UI Design
- Testing
- Integration
- Mobile accessibility

## 1.4 Milestones and tasks

| Date | Name | Tasks |
|---|---|---|
| 09.03.16 | End of Inception | Setup project management tools<br>Setup documentation environment<br>Define milestones<br>Define project scope<br>Estimate efforts<br>Risk analysis<br>Task definition |
| 28.03.16 | End of Elaboration | Evaluate libraries<br>Familiarise with frameworks and services<br>Write use cases<br>Create UI Mockups<br>Do domain Analysis<br>Implement architecture prototype<br>Write test concept<br>Define sprint goals |
| 02.06.16 | End of Construction | Alpha sprint<br>Beta sprint<br>Final sprint<br>Integration Tests<br>Write and execute unit tests |
| 17.06.16 | End of Transition | Acceptance tests with BLINQ<br>Handover code<br>Finish poster<br>Finish documentation<br>Deploy BookingTool |

## 1.5 Risk analysis

We minimize all the project typical risks like discontinued support for used libraries or scope creep by building a modular application and adjusting our development process accordingly. During our construction phase we will create multiple deliverable, starting with the minimum viable product which will receive more features with each iteration.
However, there is one project specific risk we would like to point out.

### 1.5.1 Interface to BLINQ services

To access advertiser information we will interact with BLINQ infrastructure. BLINQ provides us an interface to their services. Since these APIs are not public it is difficult to estimate the effort to interact with these services. We will work closely together with BLINQ during our elaboration phase. This will minimise the risk.

## 1.6 User interface design

To ensure a good user experience we want to follow the Nielsen usability heuristics. (Nie)

- **Visibility of system status**
  The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
  The BookingTool is using responsive desing to achieve this.

- **Match between system and the real world**
  The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
  The BookingTool uses the terminology used in Marketing teams.

- **User control and freedom**
  Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
  Users can abort any action at any time and start over without leaving the system in an inconsistent state.

- **Consistency and standards**
  Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

- **Error prevention**
  Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

- **Recognition rather than recall**
  Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

- **Flexibility and efficiency of use**
  Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

- **Aesthetic and minimalist design**
  Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

- **Help users recognize, diagnose, and recover from errors**
  Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

- **Help and documentation**
  Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

## 1.7 How ad delivery works

Whenever a user interacts with the BLINQ app, so called key words are sent from the BLINQ servers to the app. These keywords contain information about the location based on GPS, beacon check-ins, age, and gender of the user. The BLINQ app sends these keywords to the Google DoubleClick for Publishers service where the ad details are stored. Each ad has a set of keywords assigned. If the keywords match the ad is delivered to the BLINQ app user.

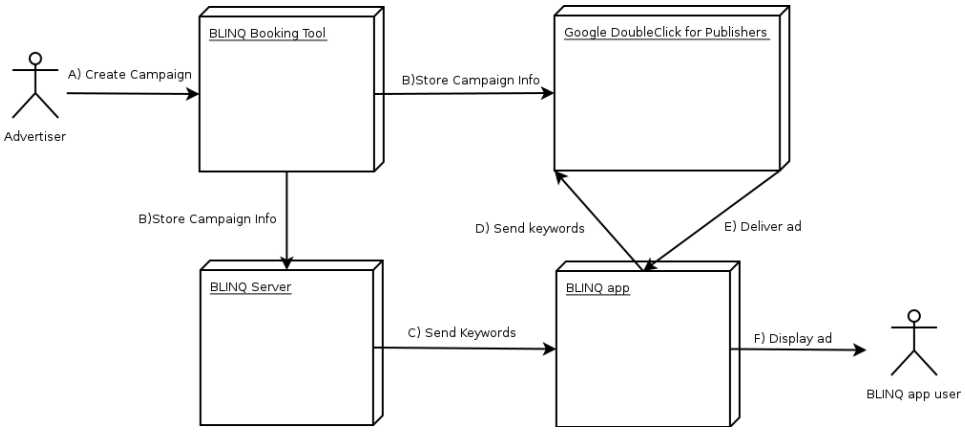


Figure 1.1: System Overview

A An advertiser creates a campaign

B The campaign information is stored on BLINQ's servers and in Google DFP

C The BLINQ server sends keywords to the BLINQ app

D The app sends keywords to Google DFP

E Google DFP checks if there is a LineItem (Campaign) that fits the keywords. If there is a match a creative is sent to the BLINQ app

F The ad is displayed to the user

## 1.8 Similar solutions

- **booking.20min.ch**
  The newspaper "20min" provides a web application to book ads on their homepage. The user interface has the same approach as the BLINQ BookingTool will have. It is possible for unregistered users to easily create an ad and see a preview of it. This way, the user can examine the possible settings and has therefore less obstacles to evaluate the tool.

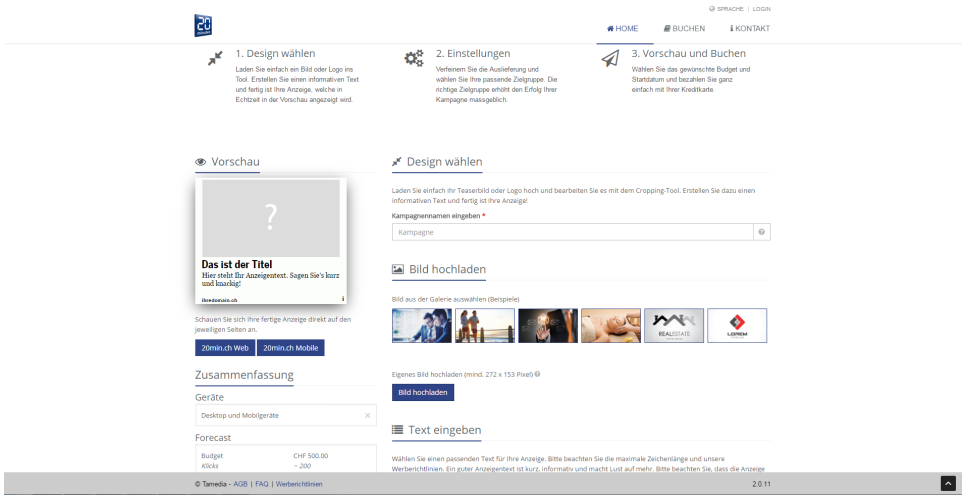


Figure 1.2: booking.20min.ch

The ads are location based. An advertiser can choose the cantons that should be targeted.

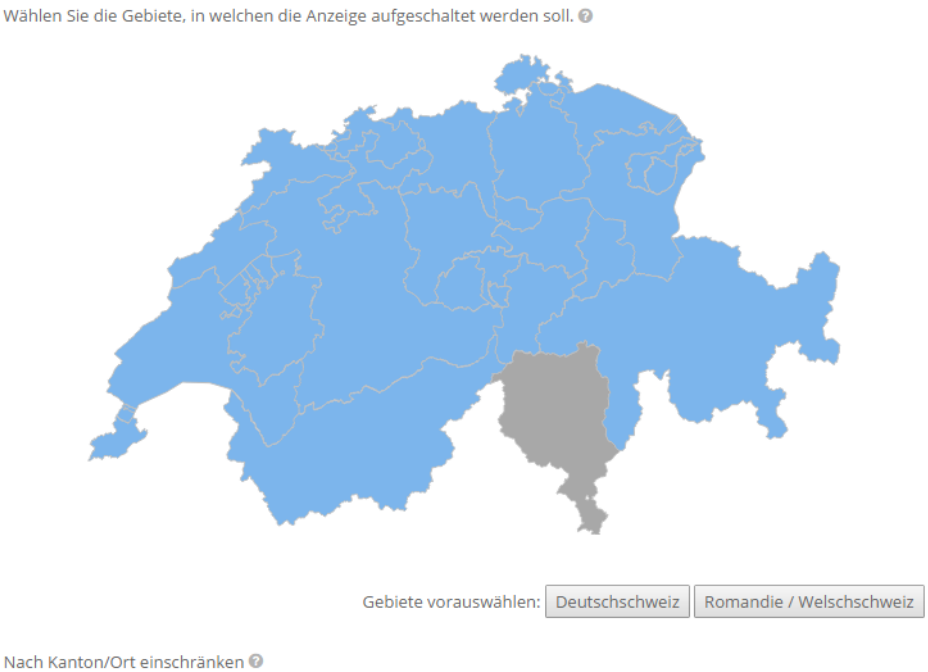


Figure 1.3: booking.20min.ch map

With BLINQs approach of beacon targeted advertisements, ads can be delivered more precise. This is an advantage of the BookingTool.

- **Google AdWords**
  AdWords provides a functionality to create location based advertisements. It locates users through a variety of technologies: IP address, GPS, Wi-Fi, Google's cell ID (cell tower) location database. The most accurate of those technologies need the users approval. With those technologies enabled and the huge user base of Google, this is the biggest concurrence to the BookingTool. The only advantage the BookingTool has, is its accuracy indoors. For example a dress shop that wants to only target customers walking through their women's clothing section.

## 1.9 BeaconsInSpace

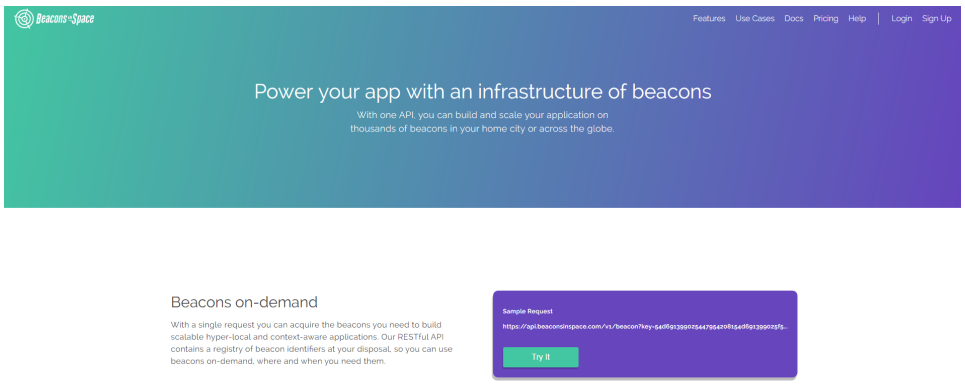


Figure 1.4: BeaconsInSpace homepage

BeaconsInSpace is a marketplace for beacon networks. It is possible to rent beacon networks from different owners at a standardized pricing. Services like this would enable the Blinq BookingTool to get a higher coverage for its location based service.

| Unique Beacon Bumps | Price Per Request |
|---|---|
| First **1,000** | **Free** |
| Next **99,000** | **$0.001** |
| Next **900,000** | **$0.0005** |
| Next **9,000,000** | **$0.0003** |
| Next **90,000,000** | **$0.0001** |
| Above **100,000,000** | **$0.00001** |

Figure 1.5: BeaconsInSpace pricing

However BeaconsInSpace is not a direct competitor to the BookingTool because only beacons are rented and not a whole advertising delivery platform.

## 1.10 Domain analysis

Advertisers can book campaigns by either choosing area or beacon targeting. Each campaign consists of an order which can consist of one or more LineItems. A LineItem has stored at least one creative. This is the image that is displayed to the BLINQ app user. A LineItem also contains the Targeting. For each selected beacon the targeting contains a so called keyword. This keyword is used to identify the selected beacon and trigger the correct advertisements.

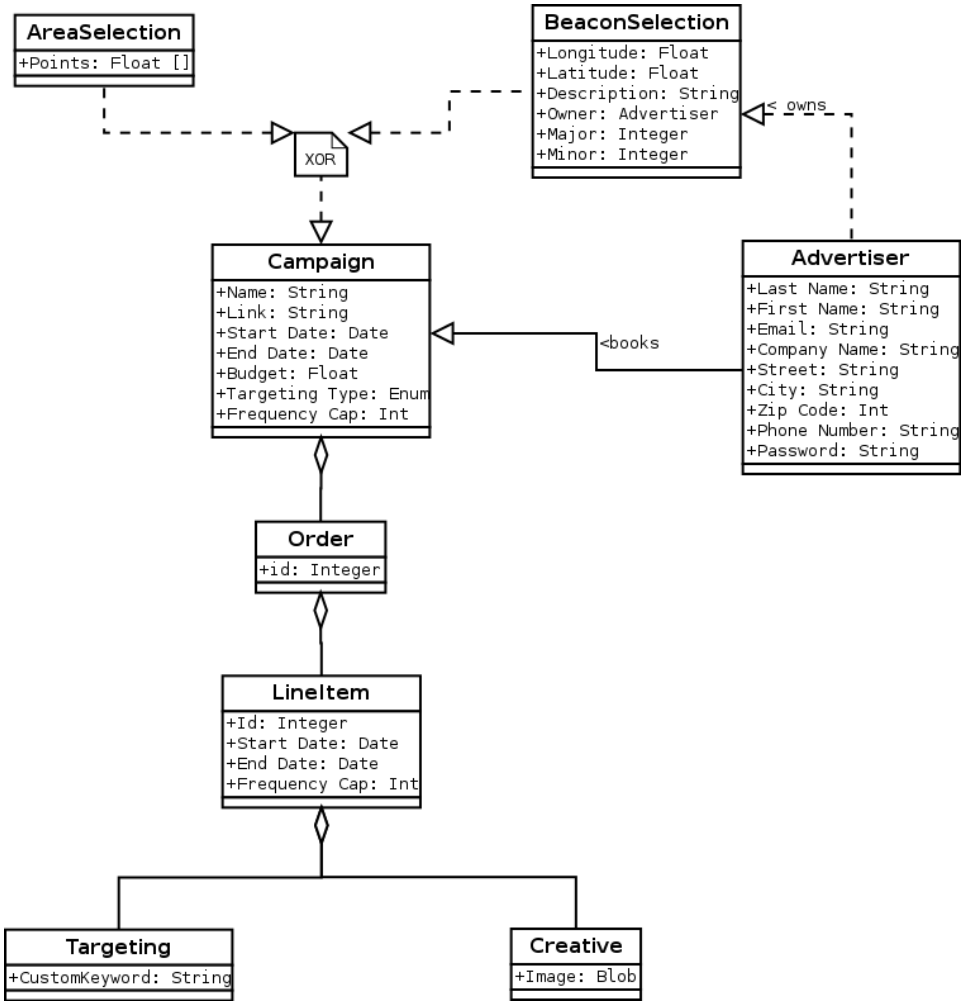


Figure 1.6: Domain analysis

## 1.11 Security concept

The BookingTool is openly available on the Internet. Since it provides access to private data the following points have been evaluated in a security analysis:

### 1.11.1 Authentication

Advertisers and BLINQ employees must authenticate themselves by providing a valid user name and a password. The user name is the Email address. This makes it easier to check the authenticity of an advertiser because the email domain will most probably be affiliated with the company. To ensure password security we enforce a minimum length of 8 and it has to contain at least one digit. This leaves $2.90 * 10^{12}$ possible passwords, if a users chooses a minimal secure password. We abstained from implementing additional measures like two-factor authentication in favor of usability.

### 1.11.2 Authorization

There are three levels of authorization. A unregistered user can register and create a campaign and use the contact form. A registered user can login/logout, create campaigns, change his credentials and use the contact form. BLINQ employees have access to the administrative interface where they can manage advertisers and campaigns. Privileges to the administrative interface can only be granted manually by a BLINQ employee.

### 1.11.3 Confidentiality and privacy

Company details as well as campaign details are treated as confidential information. The BookingTool ensures that this data is only accessible by its owner or BLINQ employees. By tying the information to an advertisers account the BookingTool always keeps control of who is able to access it.

### 1.11.4 Integrity

Integrity of data is paramount for the BookingTool. It relies on data being transmitted unaltered, therefore only secure channels are used to communicate with other services.

## 1.12 Description of frequently used Terms

During this thesis we will use a specific set of terms, which are defined here.

### 1.12.1 Google DFP

Google DoubleClick for Publishers (DFP) is an ad management solution that can schedule, deliver, and measure a digital ad inventory. BLINQ uses DFP to schedule and deliver ads.

### 1.12.2 Advertiser

An advertiser is anyone who wants to or does advertise on the BLINQ app using the BookingTool.

### 1.12.3 Campaign

A campaign consists of an order which contains the actual advertisement. This means a campaign is equivalent to an advertisement in the context of this work. However this abstraction is still made because at some point in the future the BookingTool may be expanded and it might be possible to start a campaign with multiple ads.

### 1.12.4 Ad or LineItem

An ad belongs to a campaign and is the unit that is actually displayed to the user. Google DFP uses the term LineItem which is equivalent to an ad.

### 1.12.5 Umbrail

Umbrail is a BLINQ server that provides the interface all campaign and company related data.

### 1.12.6 Furka

Furka is a BLINQ server that provides the interface all beacon related data.

### 1.12.7 Beacon

BLINQ uses the bluetooth beacon technology to locate its users. Beacons are small wireless sensors that can be attached to an object at any location. They broadcast a bluetooth signal that is received by a smartphone.

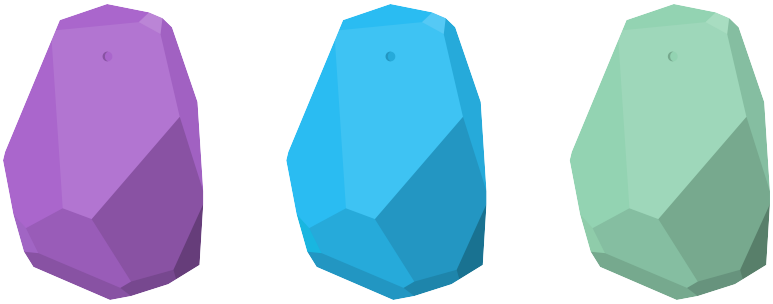

Figure 1.7: Esimote beacons as used by the BLINQ AG

### 1.12.8 BLINQ Admin

The superuser is a BookingTool user that has access to the admin interface. This is typically a BLINQ employee.

## 1.13 Code Style Guidelines

In order to simplify the collaboration and reduce the effort to introduce new developers to the code, we agreed on the following Guidelines:
https://www.python.org/dev/peps/pep-0008/

## 1.14 Internationalization

BLINQ has most of its beacons deployed in Switzerland but the app itself is international. In respective of a potential expansion into other countries the BookingTool features multiple language support.

## 1.15 Campaign states

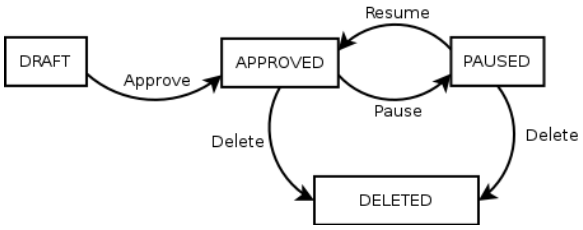


Figure 1.8: state diagram for campaigns

When a campaign is created by an advertiser it is in the "DRAFT" state. In this state the campaign is created but awaits approval by BLINQ. Once BLINQ hase approved a campaign it is "APPROVED". An approved campaign can be paused, resumed and deleted by the advertiser.

# Architecture

## 2.1 Use Cases

Large use cases are fully dressed, the small ones in brief form. Optional functionalities are recognizable through dotted lines in the respective graphics.

### 2.1.1 UC1: Create campaign

| UC1: Create campaign | |
|---|---|
| Goal | Create a new Campaign |
| Primary Actor | Advertiser |
| Stakeholders & interests | Advertiser: Wants to easily create a campaign which is then displayed in BLINQ App<br>BLINQ Admin: Wants to be notified when advertiser created a new campaign |
| Preconditions | Advertiser is on campaign creation page |
| Postconditions | A new campaign is created and stored in BLINQ Booking Tool with status stopped. Admin is notified. |
| Main success scenario | |
| 1. Advertiser (A.) enters campaign name and URL<br>2. A. uploads image<br>3. A preview of the campaign is shown to advertiser<br>4. A. chooses beacon targeting<br>5. A. chooses one of his beacons as target beacon<br>6. Budget is set by A.<br>7. System calculates click goal and displays it. Click goal is calculatet as following: budget/price per click.<br>8. UC2: A. signs in<br>9. A. Sends campaign | |
| Alternative flows | |
| 4b. A. chooses area targeting<br>1. A. selects area on map | |

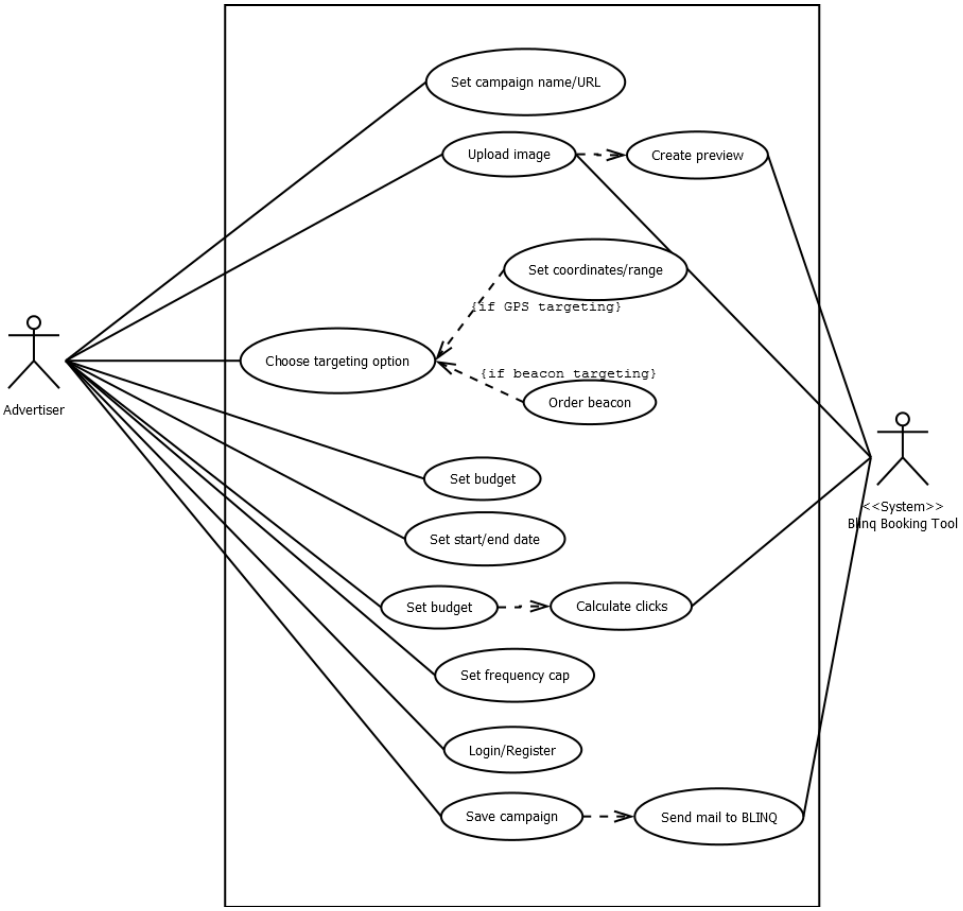


Figure 2.1: Create campaign

### 2.1.2 UC2: Sign in

| UC2: Sign in | |
|---|---|
| Goal | Register a new account or sing in with existing one |
| Primary Actor | Advertiser |
| Stakeholders & interests | Advertiser: Wants to log in to be able to send a campaign request |
| Preconditions | Advertiser is not signed in |
| Postconditions | Advertiser is signed in |
| Main success scenario | |
| 1. Advertiser signs in with his credentials | |
| Alternative flows | |
| 1b. Advertiser forgot his password<br>1. A. selects "forgot password"<br>2. An E-Mail is sent to A. to reset password | |

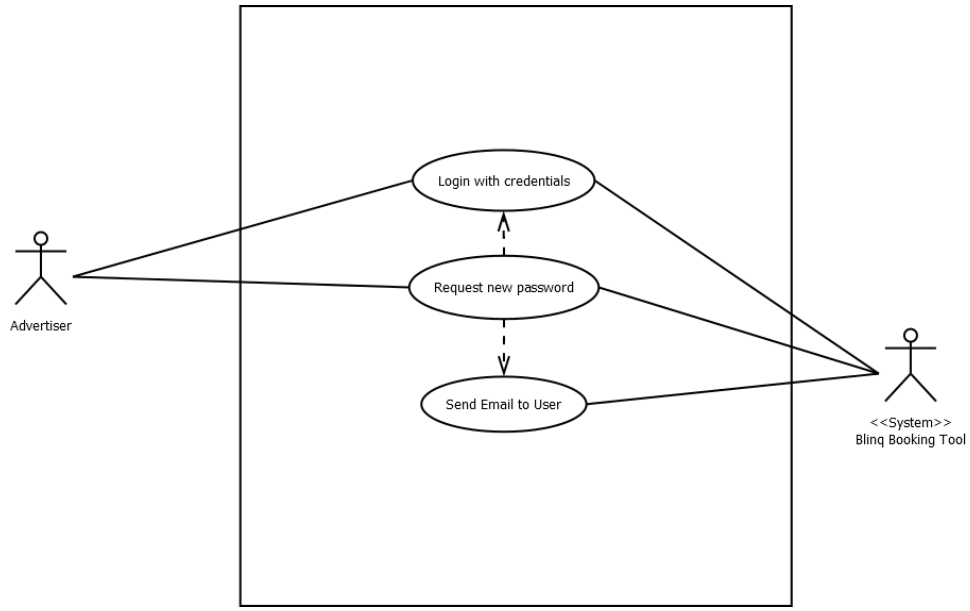


Figure 2.2: Sign in

### 2.1.3 UC3: Edit campaign

| UC3: Edit campaign | |
|---|---|
| Goal | Advertiser can change campaign |
| Primary Actor | Advertiser |
| Stakeholders & interests | Advertiser: Wants to be able to change his campaign<br>BLINQ Admin: Wants the users to be able to change campaigns so he doesn't need to do it manually |
| Preconditions | Advertiser is signed in and on campaign overview page |
| Postconditions | Campaign is changed according to the input of the advertiser |
| Main success scenario | |
| 1. Advertiser starts/stops campaign | |
| Alternative flows | |
| 1b. Advertiser changes frequency cap<br>1c. Advertiser changes budget | |

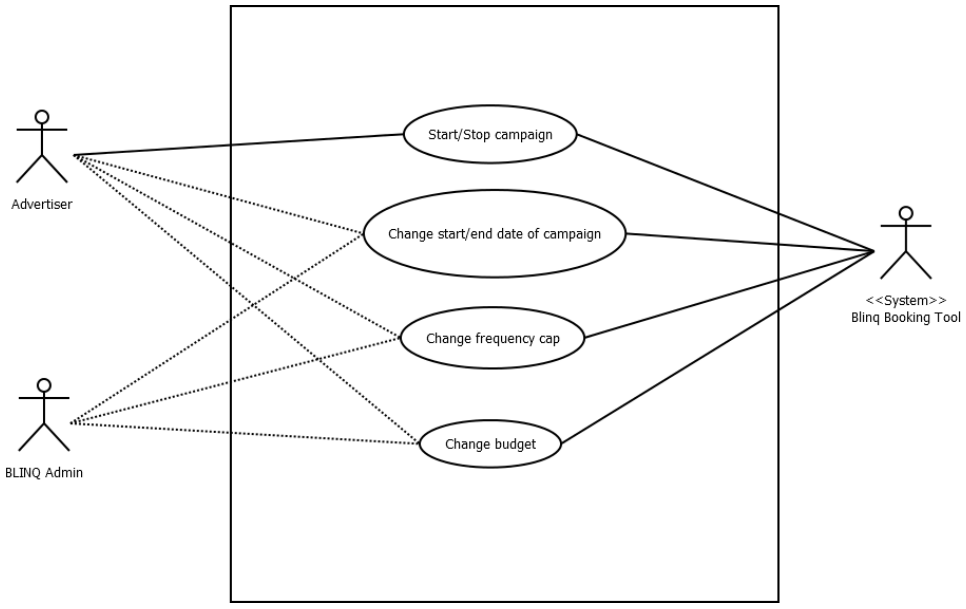


Figure 2.3: Edit campaign

### 2.1.4 UC4: advertiser CRUD

An advertiser can change his credentials. This includes following attributes: First name, last name, email, company, street, city, zip code, phone number and password. This also includes the CRUD operations from the administrator interface.

### 2.1.5 UC5: Contact BLINQ

An advertiser - with or without account - can contact BLINQ via a form in the BookingTool. The advertiser has to enter his name, email, subject and message. Those informations are sent to the BLINQ admin via email.

## 2.2 User Interface

The BLINQ BookingTool has to be usable via browser from a desktop computer and via smartphone. We decided to use Twitter Bootstrap (Twi) to achieve out of mobile usability. While this works for most elements some will still need customizations. Therefore, we decided to draw separate mock ups for the mobile and desktop interface.

### 2.2.1 Landing Page

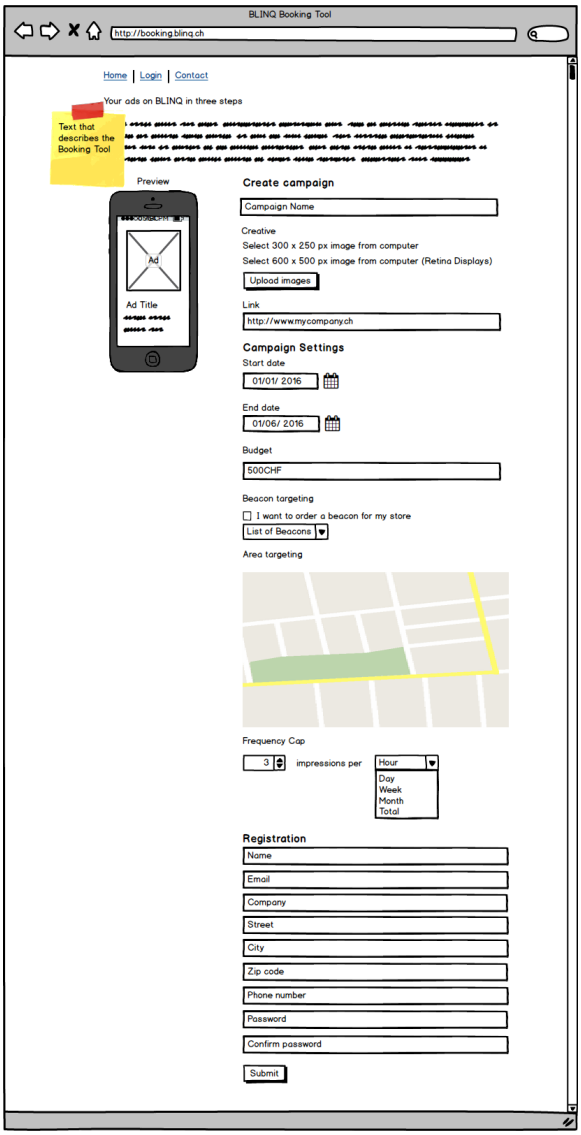


Figure 2.4: Landing Page


Figure 2.5: Mobile Landing Page

The landing page is the first page a potential advertisers sees. It is important that it is clear, easy to understand and supports the advertiser with creating his campaign.

### 2.2.2 Campaign Overview

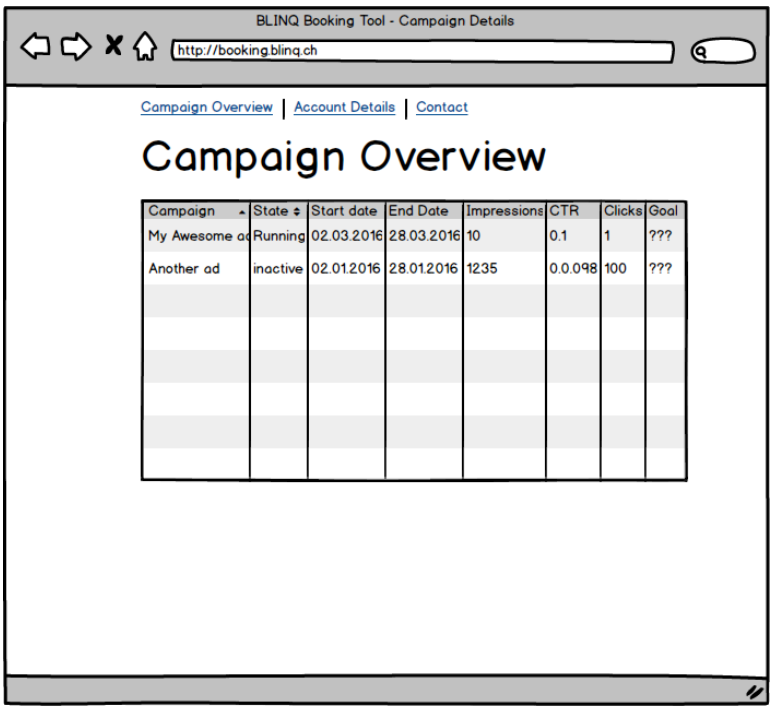


Figure 2.6: Campaign Overview Page


Figure 2.7: Mobile Campaign Overview

The campaign overview is the dashboard for all running campaigns. It displays all relevant information about all campaigns at one glance.

### 2.2.3 Advertiser Account Details

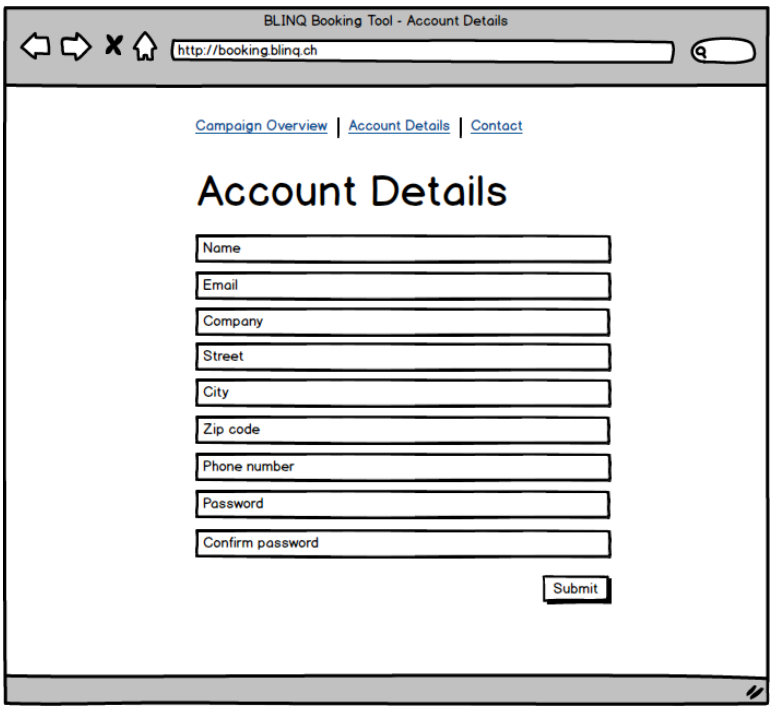


Figure 2.8: Advertiser Account Details Page

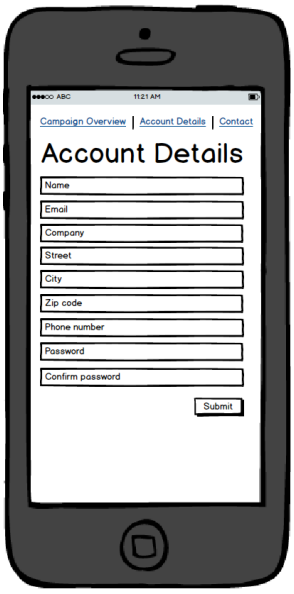


Figure 2.9: Mobile Advertiser Account Details Page

The accounts detail gives advertisers the ability to change and review their contact and billing information.

### 2.2.4 Contact

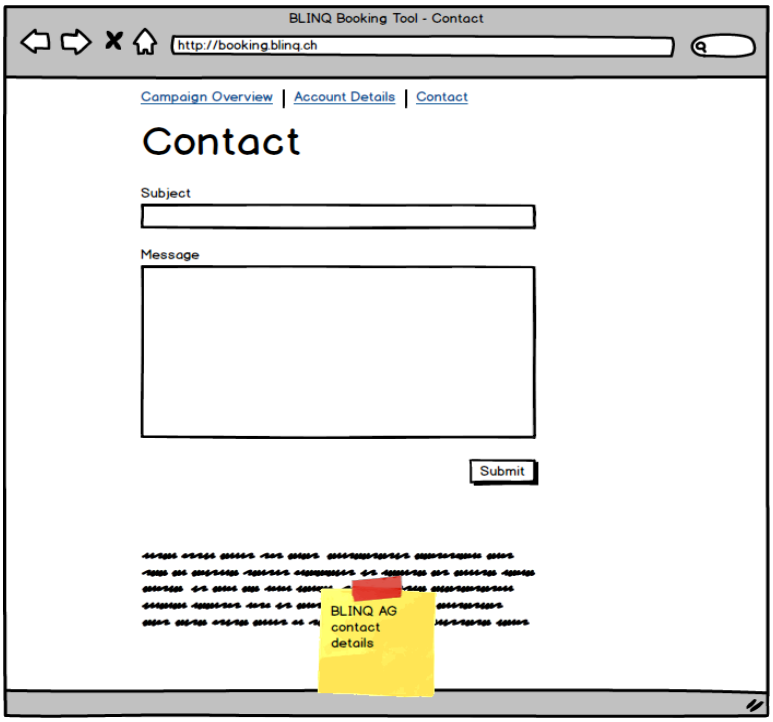


Figure 2.10: Contact page


Figure 2.11: Mobile Contact page

Through the contact form advertisers are able to reach out to BLINQ and ask for support.

## 2.3 Software Architecture

The BLINQ Booking Tool is an interface between BLINQs internal services, Google DoubleClick for Publishers (DFP) and the advertiser.

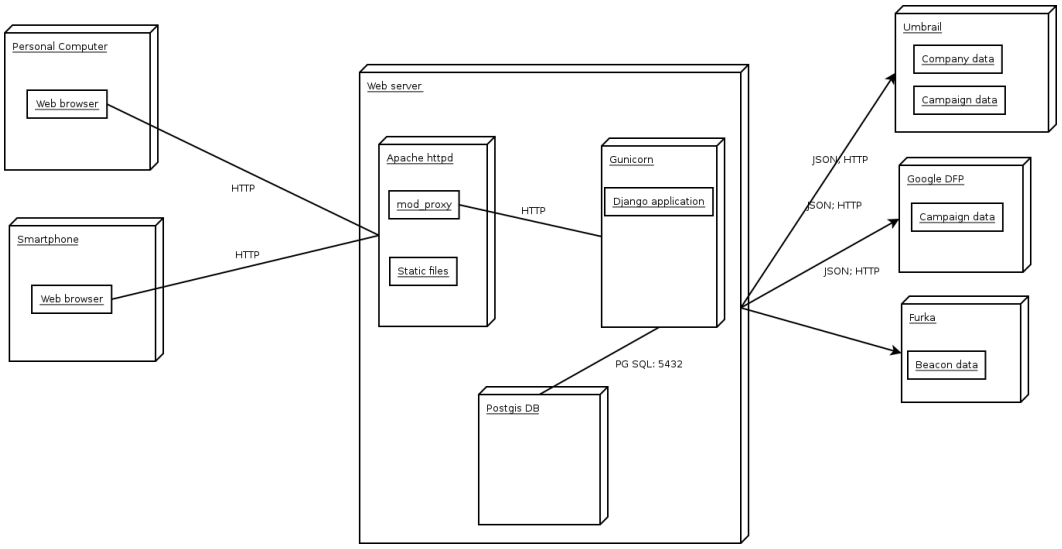


Figure 2.12: Deployment Diagram

Advertisers access the BLINQ BookingTool via web browser on a computer or smartphone. The front end is served by an apache web server with mod_proxy which directs the requests to a Gunicorn application server. The Gunicorn application server is running the BLINQ BookingTool. It uses a PostGIS Database to store relevant information and has access to the BLINQ servers Umbrail and Furka and to DFP via HTTP/JSON. BLINQ stores all the company related information on Umbrail. Furka has all the beacon related data.

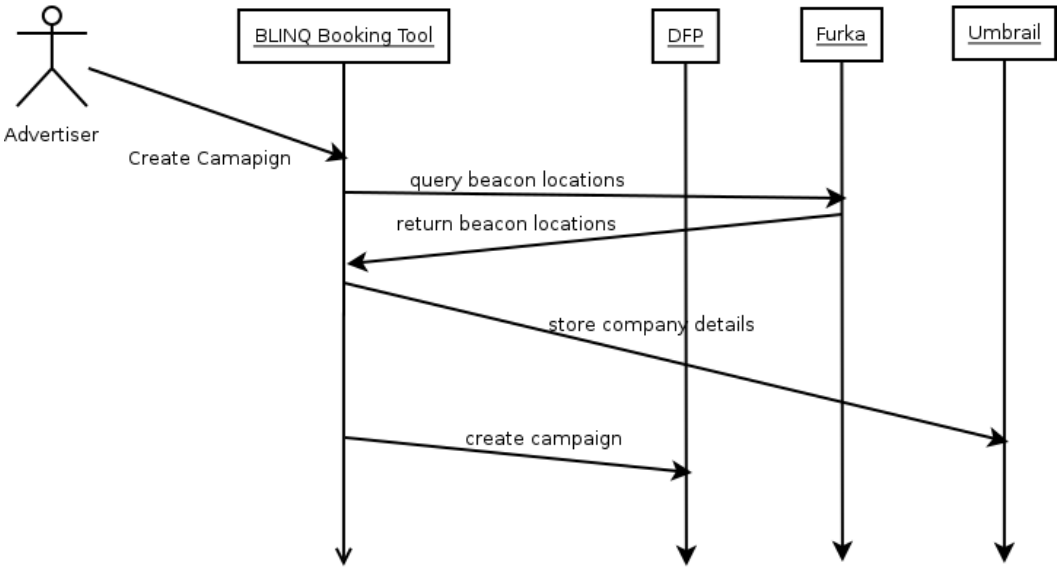


Figure 2.13: Sequence Diagram

This sequence diagram shows the communication between the services. In the pictured example a new advertiser creates an area targeted campaign. The BLINQ Booking tool queries Furka for all beacons within the selected area. Then stores the advertiser's information on Umbrail. At last, it creates the campaign in DFP.

## 2.4 Software quality

To achieve a high software quality we take the following measures

### 2.4.1 Code review

During the construction phase we plan an informal meeting once a week. In this meeting we review the code of the respective person.

### 2.4.2 Pair programming

Essential parts of the software will be pair programmed to ensure their quality right from the beginning

## 2.5 Construction

We divide our construction phase into three stages. At the end of each stage we will have a working product. Each point represents one functional requirement.

With each iteration we will add more functionality. The prioritisation of the functionality was done by BLINQ AG.

### 2.5.1 Alpha release

The alpha release is the deliverable of the first stage. It only contains basic features - it is the minimum viable product. The features of the Alpha release are:

1. Create campaign with beacon targeting
2. User registration
3. See overview of running campaigns as registered user
4. Contact form and contact details to BLINQ support
5. Email notification for each created campaign
6. Campaigns have to be approved before they launch

### 2.5.2 Beta release

The beta release is the deliverable of the second stage. It adds the following features to the alpha release.

7. Area targeting for campaigns
8. Preview of the ad while creating campaign
9. Administrative interface with an overview of all running campaigns
10. Companies can change their account details

### 2.5.3 Final release

The final release should contain all features and may contain some optional functionality.

11. Internationalization support
12. Migrate existing beacon owners to the Booking Tool
13. Registered companies can pause their campaigns

### 2.5.4 Nonfunctional requirements (NFR)

1. Responsive design which works for web browser on smartphones and computers

2. Browser compatibilty for current version of Firefox, Chrome, Internet Explorer and Safari

3. Modular architecture

4. Well documented code

5. The BookingTool has to be self-explaining. There is no user training.

## 2.6 Integration Tests

At the end of each development stage a integration test is done to check the functionality. The tests covers user stories (US)1.3, functional requirements (FR)2.5, and non functional requirements (NFR)2.5.4.

| User Story (US) | Functional Requirement (FR) | NFR | Test description |
|---|---|---|---|
| 1,2 | 1,2,5,6,8 | 1,2,5 | I1 Create Campaign with beacon selection |
| 1,2 | 2,5,6,7,8 | 1,2,5 | I2 Create Campaign with area selection |
| 2,5 | 10 | 1,2 | I3 Alter company profile |
| 2,5 | 3 | 1,2 | I4 Display campaign overview |
| 2,5 | 2,10 | 1,2 | I5 Reset password |
| 5 | 4 | 1,2 | I6 Contact BLINQ through contact form |
| 1,4 | 9 | 1,2 | I7 Alter advertisers (as Administrator) |
| 3 | 13 | 1,2 | I8 Pause/Resume campaign |
| - | 11 | 1,2 | I9 Change browser language |
| - | 12 | 1,2 | I10 Beacon migration |

The integration tests are done using different browsers on different systems including smartphones. NFR 5 was tested by letting other students book campaigns with our tool.

### 2.6.1 Integration text exeeution

| Alpha (08.04.16) | Beta (30.04.16) | Final (28.04.16) | Test |
|---|---|---|---|
| ✓ | ✓ | ✓ | I1 |
| X | X | ✓ | I2 |
| X | ✓ | ✓ | I3 |
| ✓ | ✓ | ✓ | I4 |
| ✓ | ✓ | ✓ | I5 |
| X | ✓ | ✓ | I6 |
| X | X | ✓ | I7 |
| X | ✓ | ✓ | I8 |
| X | X | ✓ | I9 |
| X | X | ✓ | I10 |

### 2.6.2 Functional Testconcept

**Test Run Details**

| | |
|---|---|
| **Date** | |
| **Software version** | |
| **Browser version** | |

**Test scenario 1: Create campaign**

| | |
|---|---|
| **Description** | Create campaign as new user |
| **Prerequisites** | Startpage loaded, not logged in, Testuser doesnt exist in database |
| **Steps** | 1. Fill in data from "Testdata-Sheet 1" and submit form<br>2. For each value reported as invalid, fill in data form "Testdata-Sheet 2" and submit form |
| **Expected Result** | • Each red colored value from "Testdata-Sheet 1" is reported invalid.<br>• Preview is shown when image is selected<br>• After input of "Testdata-Sheet 2" data, user is being forwarded to confirmation page<br>• Email is sent to BLINQ-Admin |
| **Result** | |
| **Comments** | |

**Test scenario 2: Login**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", not logged in |
| **Steps** | 1. Sign in with user credentials from "Test scenario 1" |
| **Expected Result** | • User is forwarded to campaign overview page where campaign from "Test scenario 1" is listed with state "draft" |
| **Result** | |
| **Comments** | |

**Test scenario 3: Pause campaign**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to campaign overview<br>2. Change state of campaign to paused<br>3. Log out<br>4. Sign in again |
| **Expected Result** | • State is permanently saved and did not change after fresh sign in |
| **Result** | |
| **Comments** | |

**Test scenario 4: Cancel campaign**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to campaign overview<br>2. Click on cancel button<br>3. Click "ok" on confirmation dialog |
| **Expected Result** | • State of campaign is "canceled". It can no longer be paused/resumed. |
| **Result** | |
| **Comments** | |

**Test scenario 5: Contact**

| | |
|---|---|
| **Description** | Contact admin via contact-form |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to contact page<br>2. Fill form with data from "Testdata-Sheet 4"<br>3. Submit form |
| **Expected Result** | • Name and Email are prefilled with data from logged in user<br>• After submit, user is redirected to confirmation page<br>• Email to admin is sent |
| **Result** | |
| **Comments** | |

**Test scenario 6: Account**

| | |
|---|---|
| **Description** | Change account details of logged in user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to account page<br>2. Fill form with data from "Testdata-Sheet 3"<br>3. Submit form |
| **Expected Result** | • All account informations are changed according to input |
| **Result** | |
| **Comments** | |

**Testdata-Sheet 1**

| Field | Value |
|---|---|
| Name | Testcampaign |
| Creative | - |
| Link | NOURL |
| Start date | 02.04.2016 |
| End date | 09.04.2016 |
| Beacon targeting | - |
| First name | Test |
| Last name | User |
| Email | test.user@email |
| Company | Testcompany |
| Street | Teststreet |
| City | Zürich |
| ZIP code | zipcode |
| Phone number | phonenumber |
| Password | 11223344 |

**Testdata-Sheet 2**

| Field | Value |
|---|---|
| Name | Testcampaign |
| Creative | - |
| Link | http://www.google.ch |
| Start date | 02.04.2018 |
| End date | 09.04.2018 |
| Beacon targeting | selected |
| First name | Test |
| Last name | User |
| Email | test.user@email.com |
| Company | Testcompany |
| Street | Teststreet |
| City | Zürich |
| ZIP code | 8000 |
| Phone number | 044 444 44 44 |
| Password | aabb3344 |

**Testdata-Sheet 3**

| Field | Value |
|---|---|
| First name | Test1 |
| Last name | User1 |
| Email | test.user1@email.com |
| Company | Testcompany1 |
| Street | Teststreet1 |
| City | Bern |
| ZIP code | 9000 |
| Phone number | 033 333 33 33 |
| Password | ccdd5566 |

**Testdata-Sheet 4**

| Field | Value |
|---|---|
| Subject | Testcontact |
| Comment | Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet |

## 2.7 Interface to BLINQ server

BLINQ provides us access to two of their services called umbrail and furka. Umbrail stores all the campaign, advertiser, and targeting information while furka has all the beacon data. Both services provide an interface created with swagger(IO). The interface uses HTTP GET, POST, PUT, PATCH, and DELETE to allow interaction.

## 2.8 Google DoubleClick for publishers

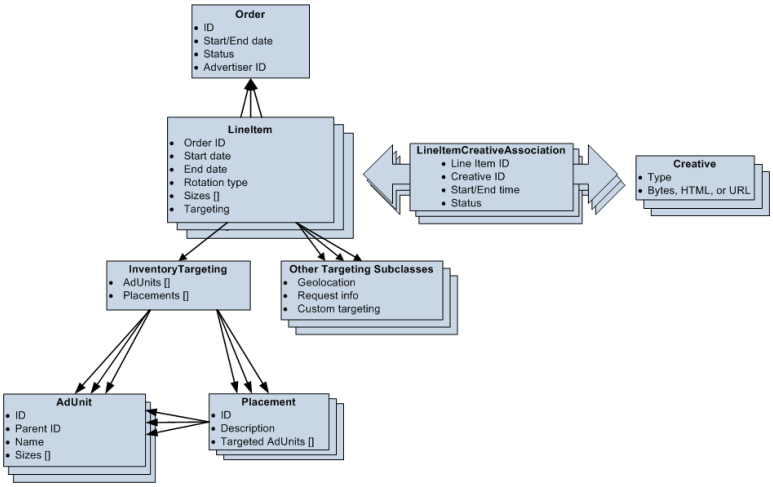


Figure 2.14: DFP class diagram

This diagram from Google's official documentation(Gooa) shows the class diagram of DFP Application Platform Interface (API). This structure resembles BLINQ's internal structure. Advertisers can place orders. Each order can contain several LineItems and each of them can be associated with one or more creatives. A LineItem has a targeting, in our case it is a CustomTargeting with CustomKeywords.

Inventory targeting and everything that belongs to it is not used by BLINQ and therefore not relevant to us. BLINQ also requested that each order just contains one LineItem.

## 2.9 Evaluation of libraries and frameworks

Some of the tasks are common problems that are already solved by a number of libraries and frameworks. In order not to reinvent the wheel, the BookingTool makes use of carefully evaluated libraries and frameworks.

### 2.9.1 Web Framework

The following web frameworks have been evaluated

1. Django
2. web.py
3. Zope

The criteria for the evaluation are the result of the analysis chapter.

| Criteria | Django | web.py | Zope |
|---|---|---|---|
| Separation of concerns (e.g. MVC) | X | - | X |
| Integration into existing infrastructure | X | X | X |
| Built in authentication and authorization | X | - | X |
| Established community and commity support | X | X | X |
| Python 2.7.x compatibility | X | X | X |
| Extensible and customizable | X | X | X |

**Result**

Web.py is a very lightweight framework for web development, however its missing some core functionality. Zope and Django both fulfill all criteria. Zope comes with its own application server which makes it bloated. Django brings the right set of features and can be deployed easily as a WSGI application. In addition Djanngo is already used at BLINQ.

This results in the selection of **Django** as web framework.

### 2.9.2 Geo Information System

To process geo location data of beacons and area selection the BookingTool needs Geo Information System (GIS). Django already provides a module called GeoDjango that includes functions to deal with coordinates and selected areas on maps. Therefore, this evaluation is limited to GeoDjango compatible GIS backends.

1. PostgreSQL (With postGIS)
2. MySQL
3. SQLite

For the GIS a certain set of functions is needed and as before the solution has to integrate into the existing infrastructure.

| Criteria | PostgtreSQL | MySQL | SQLite |
|---|---|---|---|
| 'contains' lookup is supported | X | - | X |
| Integration into existing infrastructure | X | - | - |
| Coordinate transformation | X | X | X |

**Result**

MySQL does not provide a "contains" function, which is required to retrieve all beacons within a selected area. PostgreSQL and SQLite provide all the necessary functions. **PostgresSQL** is selected because it is already used at BLINQ for storing geographical information about their beacons.

### 2.9.3 User Interface

There is a number of libraries that provide user interface elements for web pages that are browser and mobile friendly.

1. JQueryUI
2. Bootstrap
3. HTML5 Boilerplate

The criterias for the evaluation derive from the analysis chapter.

| Criteria | JQuery | Bootstrap | HTML5 Boilerplate |
|---|---|---|---|
| Mobile and browser view | X | X | X |
| Established community and community support | X | X | X |
| Supports all modern browsers | X | X | X |
| Lightweight | X | X | X |

**Result**

All libraries meet the criteria. **Bootstrap** is selected because it is already required for GeoDjango and the developers are already familiar with it.

# Implementation

## 3.1 Code structure

The BookingTool is implemented as Django project with an application called advertiser. This app contains all implementations for booking advertisements.
A project is the top level entity in Django it can contain several applications. The BookingTool consists of one Project with one application.

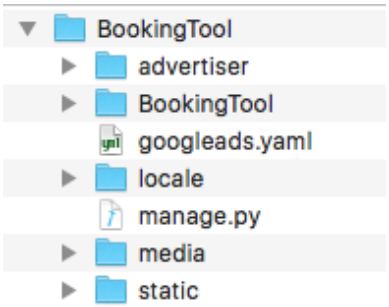


Figure 3.1: Project file structure

Some files were automatically generated by Django. More details about them can be found in the django documentation(Tea).

### 3.1.1 advertiser

The advertiser package contains most of the code for the BookingTool. It is divided into several files. The following sections describe all relevant files.

#### admin

In this file models are registered for the admin interface. Every model that is registered here can be edited by BLINQ in their admin interface.

#### blinq

The blinq class provides all methods to interact with BLINQ's API.

**bookingtool**

This file contains all logic and could be referred to as controller.

**dfp_api**

This class provides all methods to interact with DFP.

**forms**

Here are the classes for all forms with their validation code.

**furka**

furka contains all methods to interact with BLINQ's beacon services.

**models**

All data models are defined in the models file.

**static**

The static folder contains all application wide static files like images, .css, and .js files.

**tests**

This file contains all unit tests.

**urls**

All app specific routes are defined here.

**views**

The views file contains all logic regarding the delivery of web pages.

### 3.1.2 BookingTool

This package contains application wide settings.

**settings**

All project wide settings like database, email configuration, plug-ins and further configuration are defined here.

**urls**

This file contains all project wide routes.

### 3.1.3 locale

This folder contains the translation. Currently English is the default and every additional translation has it's own file in this folder.

### 3.1.4 media

The uploaded creatives are stored in the media folder

### 3.1.5 static

The static folder contains all project wide static files like images, .css, and .js files.

## 3.2 Code metrics

**Number of Python files**: 38
**Lines of Python code**: 5838
**Number of classes**: Not representative because python is not solely object-oriented. For this project, very few classes were needed.
**Lines of HTML/CSS/JS code**: The project has different libraries whose HTML/JS/CSS code needs to be within the project and some of our JS had to be included into the HTML-files because of ad blockers. So this value would not be representative.

## 3.3 Web application security

Because the BookingTool is a web application that is accessible over the Internet, security is very important. The following technical measures have been implemented to protect the application and its users.

### 3.3.1 Secure communication with our services

The BLINQ services are within the same local network in which the BookingTool is going to be deployed. The communication with DFP happens using a TLS encrypted connection. The BookingTool itself will be reachable over HTTPS.

### 3.3.2 Cross Site Request Forgery (CSRF)

A malicious website could contain a link, a button or some sort of Javascript that is intended to perform an action on the BookingTool using the credentials of a logged in user. The BookingTool prevents this by using a CSRF cookie. This cookie is set

to a random value. It is meant to be permanent, but since there is no way to set a cookie that never expires, it is sent with every response. The value is changed each time a user logs in.

Each form has a hidden input field, generated with the $\{\%csrf\%\}$ token. It contains the value from the CSRF cookie. All incoming requests must contain this value and possess a cookie. If this is not given, the user will get a 403 error.

### 3.3.3 Clickjacking Protection

This type of attack occurs when a website tricks the user into clicking on a concealed element of another site that has been loaded in a hidden frame over the displayed site.
There is a solution to this problem and its called X-Frame-Options. The X-Frame-Options is a HTTP header which works with all modern browsers. If this header is set to "DENY" the browser will block the resource from loading in a frame.

## 3.4 Challenges and decisions

This section documents how various challenges and decisions impacted the implementation of the BookingTool.

### 3.4.1 Python 2.7 instead of Python 3

After evaluating our frameworks and libraries, we decided to use Python 3. However, after implementing the architecture prototype we realized that Google's DFP client library was not Python 3 compatible. Even though the documentation stated it was (Goob). This caused us to rewrite parts of our DFP interface code.

### 3.4.2 Area targeting and GPS

To complement the beacon targeting and to give advertisers the possibility to target BLINQ users in larger areas the BookingTool provides area targeting. Originally these areas should use GPS signals to track a user's location and trigger the advertisement. This feature was canceled due to a resource shortage on BLINQ's side. Instead the BookingTool will select all beacons within the selected target area. If there is no beacon in the selected area the advertiser is notified and can change his area selection.

### 3.4.3 Performance

From the start DFP was not quick but especially in the later development stages when queries got more complex it became slow. After investigation, we found two possible solutions to increase the performance and decrease the loading time for requests to DFP. The first solution was to store all data locally which would

increase the number of redundantly stored data even more, which we did not want. The second solution would have been to restructure the application and make use of asynchronous calls. This would have required massive changes in the architecture, which was at the advanced point of this thesis too much of a risk. We decided together with BLINQ to put up with the occasional poor performance in favor of duplicating data or risking an unfinished implementation

## 3.5 User Instructions

### 3.5.1 Create campaign

The campaign creation is done in three easy steps.

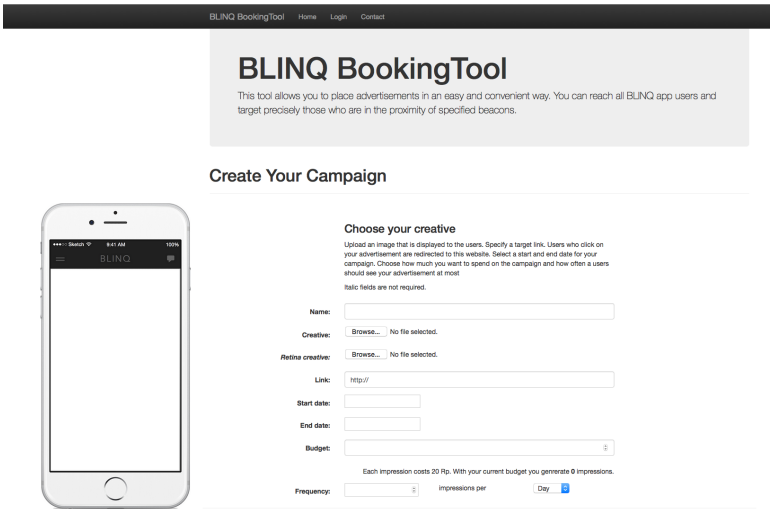


Figure 3.2: General campaign settings

First, an advertiser has to provide general information like a campaign name, a creative that is displayed in the app, a start and end date, a target link, a budget and a frequency cap. The BookingTool provides a preview of the ad.

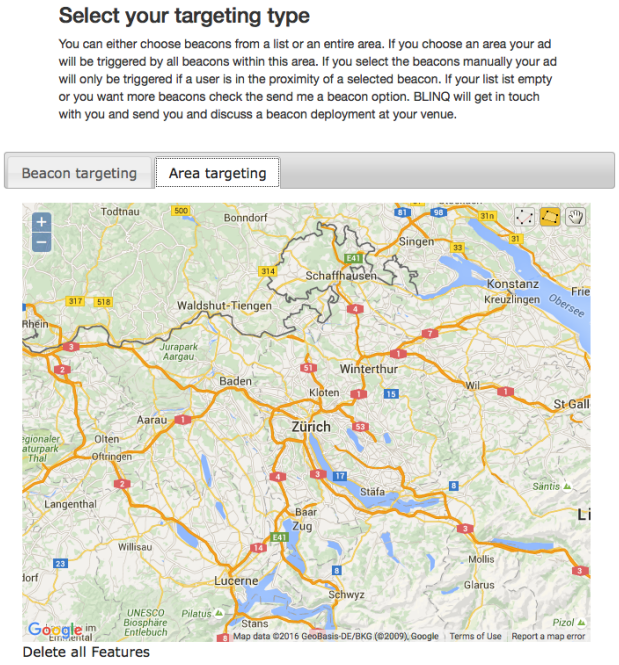


Figure 3.3: Targeting settings - Area targeting

Next, an advertiser can choose between beacon and area targeting. The picture 3.3 shows the area targeting. The advertiser can select an area on the map. The ad is then delivered to every BLINQ app user who is in the proxmity of a beacon within this area. If there is no beacon within the selected area the advertiser is informed.

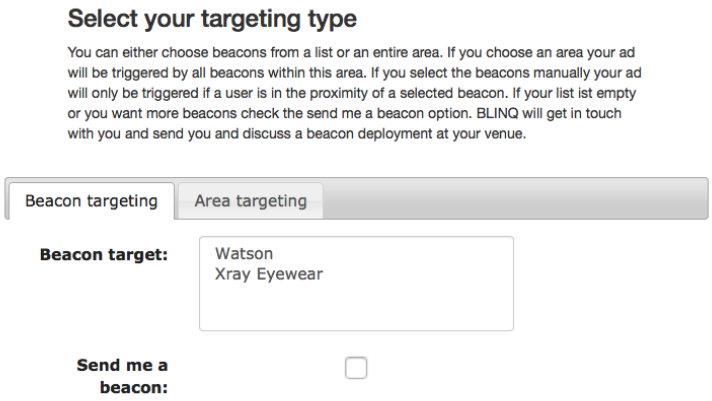


Figure 3.4: Beacon targeting

The beacon targeting shows a list of beacons that belong to the advertiser. It works similar to the area targeting but instead of selecting an area on the map an advertiser can specify the beacons directly.

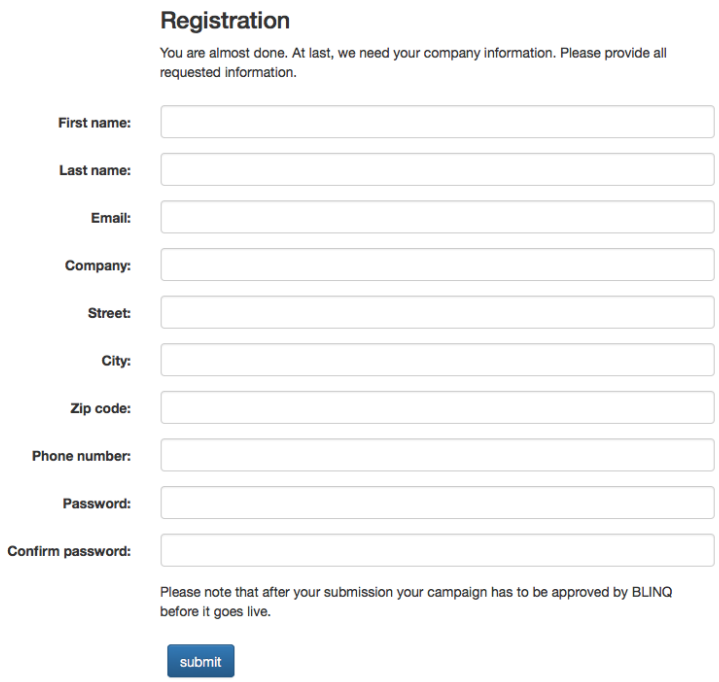


Figure 3.5: Beacon targeting

Last, the advertiser has to fill out a registration form. This step is omitted for registered advertisers.

### 3.5.2 Manage campaigns

Once an account has been created, advertisers can log in and see their campaigns on the overview page.

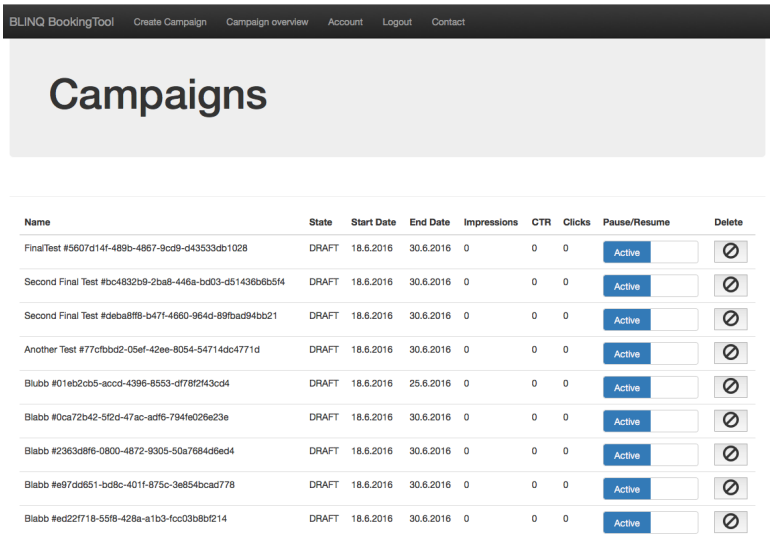


Figure 3.6: Campaign overview

In this overview an advertiser can see the status of their campaigns. They can pause, resume or delete their campaigns.

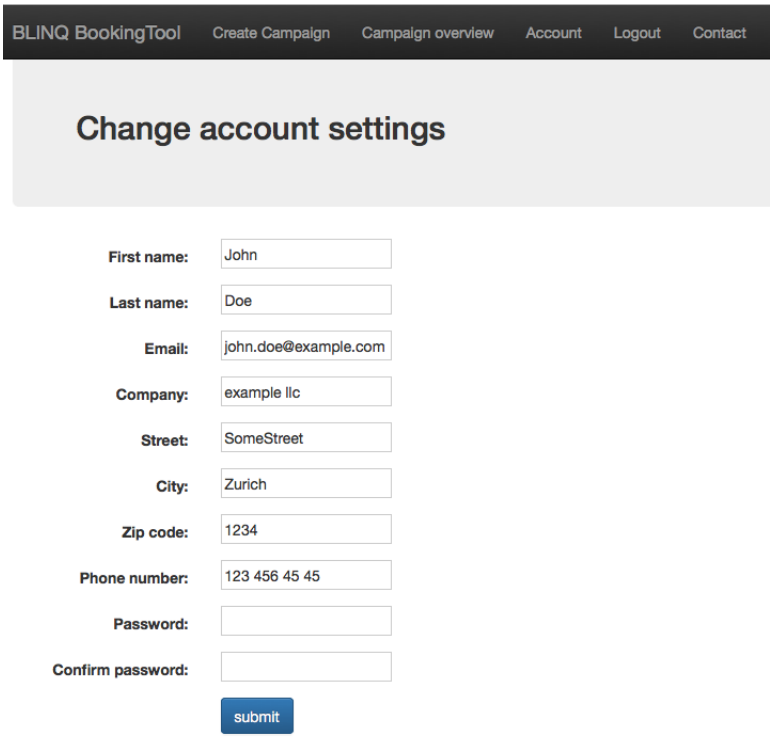


Figure 3.7: Advertiser: change account settings

Advertisers can change their account information.

### 3.5.3 BookingTool administration

The administration interface is kept very simple and straight forward.

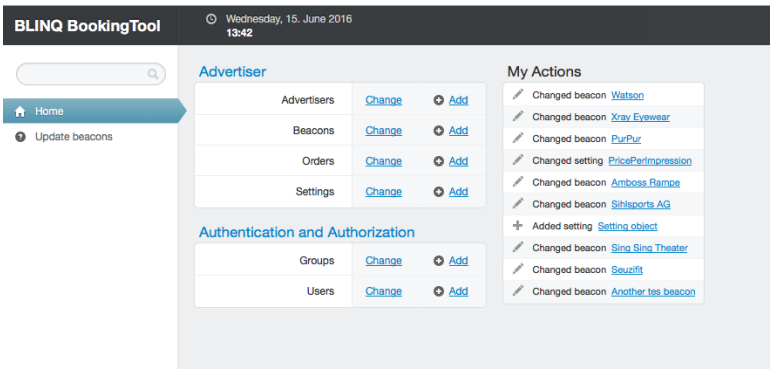


Figure 3.8: Admin interface

This interface allows BLINQ employees to view and modify registered advertiser, orders, beacons and settings. Most of these are CRUD operations.

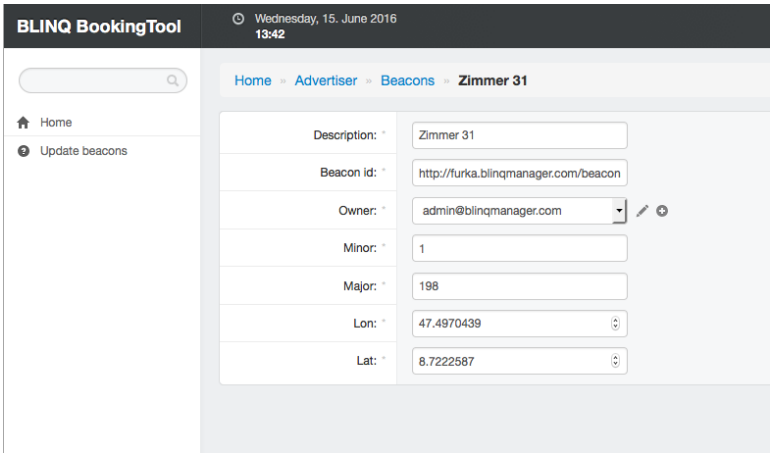


Figure 3.9: Beacon modification

One essential operation is the mapping between beacon and owner. BLINQ did not have this concept of beacon ownership yet. Once a beacon is owned by an advertiser it shows up in his beacon list and he can use it for beacon targeting.

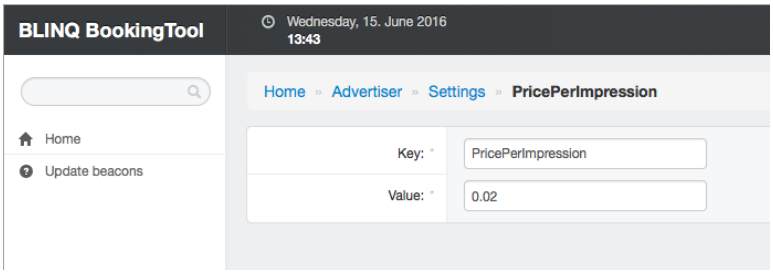


Figure 3.10: Administration settings

Another important feature are the settings. This is a key-value store and is currently just used to set the price per impression. But it can easily be extened and use for any other value that may change frequently.

## 3.6 Outlook

There is still room more features and improvement. Once the BookingTool is launched into production and gathers data form campaigns it could analyze this data and see which parameters may impact the success of an advertisement and provide suggestions.
Another useful feature would be integration into a CRM (Customer relationship management) Tool or account tool. This could improve the binding of customers and would allow automated billing.
The BookingTool could be extended so that advertisers can book campaigns for other apps.

# Transition

## 4.1 Acceptance protocol

During the last integration test BLINQ confirmed that all user stories 1.3 are implemented and working correctly as shown in the integration test matrix 2.6.1.
By request from BLINQ E.19 we did not deploy the BookingTool together with there engineers onto their Amazon Web Services (AWS) cloud. But we will deliver them an easily deployable virtual machine image along with the source code.

## 4.2 Deployment

There are two ways to deploy the BookingTool. The easiest way is to import the virtual machine image. It is using the open virtualization format which is compatible with all major software vendors for hypervisors. The second way is to install the tool on an already running operating system. This approach requires more effort because one has to ensure that the software requirements are met prior the installation.

### 4.2.1 Deploy virtual machine image

BLINQ runs all it services on AWS. Therefore, this guide is specific to deploying the BookingTool on AWS. The BookingTool is provided as Open Virtualization Format (OVF) file that can be imported and deployed directly to AWS (Ama).
The image can be imported as instance using the following command:

```
ec2-import-instance disk_image_filename -f file_format -t ↩
    instance_type -a architecture -b s3_bucket_name -o owner -↩
    w secret_key -p platform_name -O your_access_key -W ↩
    your_secret_key
```

After the import you can run the instance form your AWS control panel.

### 4.2.2 Manual installation

The second way to deploy the BookingTool is to do a manual installation on a running server.

### Software requirements

In order to run the booking tool the following software needs to be installed on the host.

- Python 2.7.11
- Postgres 9.5.1.0
- PostGIS 2.2.2
- geos 3.5.0
- proj 4.9.2
- gdal 1.11.3
- pip 8.1.2
- virtualenv 14.0.5

Before installing the required python modules create a virtual environment

Listing 4.1: Creation of a virtualenv

```
virtualenv -p /usr/bin/python2 venv
. venv/bin/activate
```

Now the moduels can be installed using the requirements.txt file with the following command

Listing 4.2: Installation of python module requirements

```
pip install -r requirements.txt
```

### Installation and setup

There are different ways to setup the BookingTool for use in a production environment. This guide covers the one we consider as best practice.

Listing 4.3: Run gunicorn application server in background

```
nohup venv/bin/gunicorn wsgi:application -b 127.0.0.1:8112 &
```

This spawns two Gunicorn application server processes in the background that serve the BookingTool only to requests from localhost on port 8112. Now we need a web server that serves as reverse proxy so we can access the BookingTool from everywhere. This is an example configuration for an Apache http server.

Listing 4.4: Example apache config with ssl encryption

```
<VirtualHost *:443>


    ServerName booking.blinqmanager.ch

  ProxyPass / http://127.0.0.1:8112/
    ProxyPassReverse / http://127.0.0.1:8112/

    SSLEngine on
    SSLProtocol all -SSLv2
    SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5:!SEED:!IDEA

    SSLCertificateFile /etc/pki/tls/certs/localhost.crt

    SSLCertificateKeyFile /etc/pki/tls/private/localhost.key

</VirtualHost>
```

# Appendices

# Development infrastructure

We developed the BookingTool with the editors VIM[1] and Atom[2]. A private git[3] repository on github[4] was used as source control. After each commit TeamCity[5] pulled the latest commit, executed the tests and deployed the code to our development server. YouTrack[6] was used for project management. All our images and diagrams are drawn with dia[7]. The Charts were made with excel[8]. To write our thesis we used sharelatex[9].

[1] http://www.vim.org
[2] http://www.atom.io
[3] http://www.git-scm.org
[4] http://www.github.com
[5] http://www.jetbrains.com/teamcity/
[6] http://www.jetbrains.com/youtrack/
[7] http://wiki.gnome.org/Apps/Dia
[8] http://products.office.com/en/excel
[9] http://www.sharelatex.com

# License agreement

# Time report

We worked a total of 735 hours during 17 Weeks. We aimed to work 21 hours per week and slightly more in the last two weeks which would result in 720 hours in total.

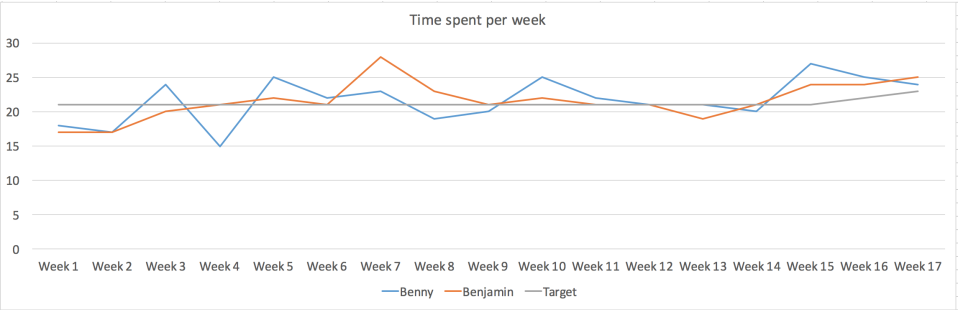


Figure C.1: Time spent per week

We worked 21.6 hours per week on average. The planned effort per phase matches the effective spent time quite good. There are only minor differences. The largest disparity is the inception phase. We finished 7 hours earlier than estimated.

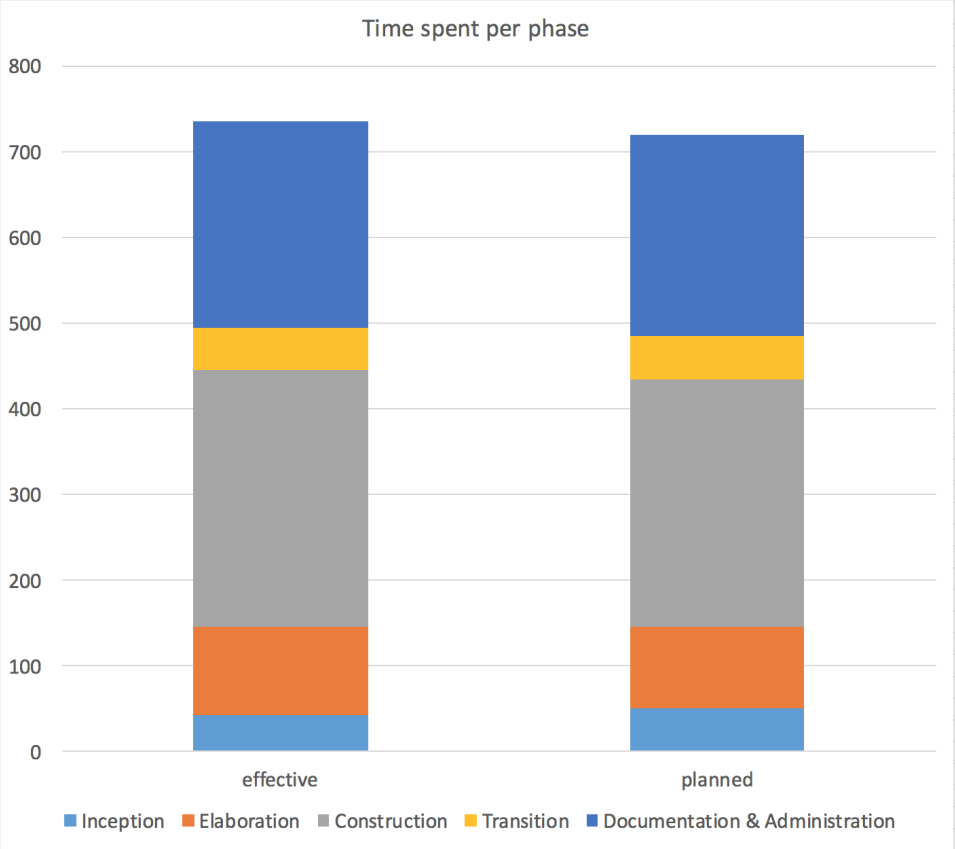


Figure C.2: Time spent per week

All other phases took slightly longer than planned. This resulted in a total of 15 more hours that we spent.

| | Total time spent |
|---|---|
| Benny Gächter | 368 |
| Bejmain Wilhelm | 367 |

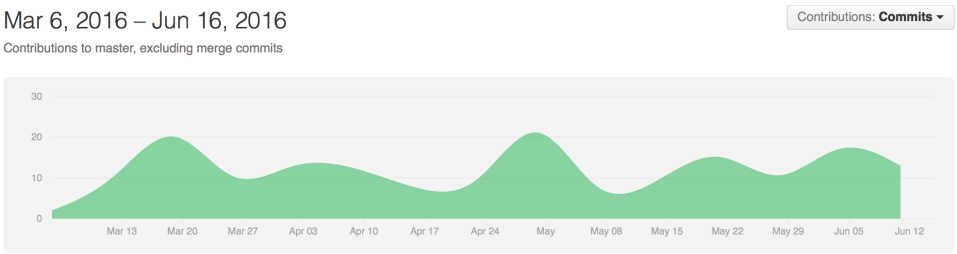


Figure C.3: Github commits

This commit graph shows some minor peeks but is in general very smooth. It

resembles the steady pace of development we had while working on this thesis.

# Protocols

## E.1 23.02.2016 - BA Kickoff Meeting

### E.1.1 Agenda

- The tasks for the thesis have been reviewed

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.2 25.02.2016 - BA Kickoff Meeting @ Blinq

### E.2.1 Agenda

- Blinq wants to specify the requirements and goals as user stories
- We can use blinq's infrastructure for integration tests (aws)
- We will use Python/Django so that our solution integrates nicely with the existing code at blinq
- booking.20min.ch can be used as an example
- The web page for customers should be mobile accessible
- Payment options (e.g. directly via credit card) would be an optional task

**Participants**

- Jan Berchtold
- Benjamin Wilhelm
- Benny Gächter

## E.3 01.03.2016

### E.3.1 Agenda

- Review gantt diagram and project planing
- Estimate risks
- Specify project scope until next week

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.4 08.03.2016

### E.4.1 Agenda

- Scope has been reviewd and is good
- Risk analysis can be limited to errors in time management
- Construction phase should be splitted into different stages. At the end of each stage there should be a deliverable
- A meeting with BLINQ will be held later this week

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.5 10.03.2016

### E.5.1 Agenda

- We reviewed the User Interface Mockups. The following points were decided:
  - Advertisers have to upload their own image, no stock images are provided

  - The booking page should display a calculation of how many impressions will be generated with the given budget
  - The Booking tool is for the swiss market only

- During the construction phase we will implement a minimum viable product and then incrementally add features. We can define the features for the releases ourselves. The only requirement for the minimum viable product is the ability to select beacon targeting.

- On the next meeting (17.03.16) we will define the interface to umbrail to exchange information about advertisers

**Participants**

- Jan Berchtold
- Benjamin Wilhelm
- Benny Gächter

## E.6 15.03.2016

### E.6.1 Agenda

- The co-examinors are Mirko Stocker and Vikram Kriplaney
- We will arrange the mid term presentation and the final presentation with the two co examinors. Proposed dates are 12th of April for the mid term and 8th of August for the final presentation

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.7 17.03.2016

### E.7.1 Agenda

- Organize Test Beacon for HSR to have a full test run to see whether advertisements are triggered with recorded ad.
- BLINQ provide us a system overview

- We will get a subdomain for our build server like booking.beta.blinq.ch
- BLINQ needs to implement area targeting conditions on their ad server
- the following actions will be done via Furka server
  - get beacons by company
  - get areas bu company
- the following actions will be done via Umbrail server
  - Delivery: viewset for company, order, campaign, creative (redirect)
  - Targetting: viewset rules, conditions

**Participants**

- Jan Berchtold
- Tim Bachofen
- Marco Glauser
- Benjamin Wilhelm
- Benny Gächter

## E.8 22.03.2016

### E.8.1 Agenda

- Mid term presentation will be on 15th of April with Vikram Kriplaney
- Review of documents
- End of elaboration artifacts will be reviewd next time
- BLINQ needs to implement area targeting conditions on their ad server

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.9 22.03.2016

### E.9.1 Agenda

- Mid term presentation will be on 15th of April with Vikram Kriplaney, Mr. Mehta will not be present. We will show him a preview version of the presentation on tuesday
- Review of documents
- End of elaboration artifacts will be reviewd next time
- BLINQ needs to implement area targeting conditions on their ad server
- Next meeting will be in two weeks

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.10 05.04.2016

### E.10.1 Agenda

- The midterm presentation should have to following content: Goals of the thesis, work done so far, expected results, and problems. The presentation will not be graded. It should be around 20 minutes.
- The architecture prototype could not be completely finished

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.11 07.04.2016

### E.11.1 Agenda

- LineItem in dfp corresponds to campaign on umbrail
- BLINQ is satisfied with the overall progress
- We enabled API access to BLINQs dfp advertiser network

**Participants**

- Jan Berchtold
- Benjamin Wilhelm
- Benny Gächter

## E.12 12.04.2016

### E.12.1 Agenda

- We gave a preview version our midterm presentation
- Mr. Heinzmann gave input regarding security aspects: We should further evaluation which critical data is stored in our application and how we can protect it.
- Both Mr. Mehta and Mr. Heinzman were satisfied with the progress

**Participants**

- Prof. Dr. Farhad Mehta
- Prof. Dr. Peter Heinzmann
- Benjamin Wilhelm
- Benny Gächter

## E.13 15.04.2016

### E.13.1 Agenda

- We gave our midterm presentation
- Mr. Kriplaney seemed interessted in the topic since they worked with beacons at local.ch, too.

**Participants**

- Vikram Kriplaney
- Benjamin Wilhelm
- Benny Gächter

## E.14 19.04.2016

### E.14.1 Agenda

- We are slightly behind schedule but nothing to worry about
- Mr. Mehta is happy with the our development
- Our documentation should look less like a report and more like a collection of all documents which are needed to maintain our application and understand the decision we made.

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.15 03.05.2016

### E.15.1 Agenda

- We are still making good progress and are in time
- We decided to not implement all features but ensure stability and usabilty of the Booking Tool. In a later meeting with BLINQ we will determine which features can be discarded
- The transition phase will remain the same despite BLINQs situation

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.16 10.05.2016

### E.16.1 Agenda

- We have specify that BLINQ has the rights to use the BookingTool. This can be done in a similar way as it has been done for the semester thesis. We sent Jan the contract.

- Jan and Alex are satisfied with the development of the BookingTool.
- The transition will happen as planned with the exception that we will get less support from BLINQ since their engineers Tim and Marco already left the company. We have to integrate and deploy the BookingTool on our own.

**Participants**

- Jan Berchtold
- Alex Zimmermann
- Benny Gächter

**Excused**

- Benjamin Wilhelm - sick

## E.17 17.05.2016

### E.17.1 Agenda

- The transition phase might change a bit because of the ongoing changes at BLINQ
- Demo of the BookingTool - we made good progress and are in time

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.18 31.05.2016

### E.18.1 Agenda

- We are using the buffer week to prepare the transition and do some fine tuning
- We showed a preview of the poster that we need to redesign
- Review of the abstract and management summery. We recieved some very useful advices.
- The transition is planned for the coming week

**Participants**

- Prof. Dr. Farhad Mehta
- Benjamin Wilhelm
- Benny Gächter

## E.19 02.06.2016 - Email Exchange

### E.19.1 Agenda

- We not deploy the BookingTool because does not have the resources at the moment. We will rather just hand them an easily deployable container and the source code over.

**Participants**

- Jan Berchtold
- Alex Zimmermann
- Benny Gächter
- Benjamin Wilhelm

## E.20 13.06.2016

### E.20.1 Agenda

- We did the last integration test and fulfilled all requirements
- Jan and Alex are very satisfied with the final version

**Participants**

- Jan Berchtold
- Alex Zimmermann
- Benny Gächter
- Benjamin Wilhelm

## E.21 Test protocols

### 2.5.2 Functional Testconcept

**Test Run Details**

| | |
|---|---|
| **Date** | 13.6.16 |
| **Software version** | e8a20fd |
| **Browser version** | Chrome 5.1.0.2709.89 |

**Test scenario 1: Create campaign**

| | |
|---|---|
| **Description** | Create campaign as new user |
| **Prerequisites** | Startpage loaded, not logged in, Testuser doesnt exist in database |
| **Steps** | 1. Fill in data from "Testdata-Sheet 1" and submit form<br>2. For each value reported as invalid, fill in data form "Testdata-Sheet 2" and submit form |
| **Expected Result** | • Each red colored value from "Testdata-Sheet 1" is reported invalid.<br>• Preview is shown when image is selected<br>• After input of "Testdata-Sheet 2" data, user is being forwarded to confirmation page<br>• Email is sent to BLINQ-Admin |
| **Result** | Ok |
| **Comments** | [illegible] |

**Test scenario 2: Login**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", not logged in |
| **Steps** | 1. Sign in with user credentials from "Test scenario 1" |
| **Expected Result** | • User is forwarded to campaign overview page where campaign from "Test scenario 1" is listed with state "draft" |
| **Result** | Ok |
| **Comments** | – |

**Test scenario 3: Pause campaign**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to campaign overview<br>2. Change state of campaign to paused<br>3. Log out<br>4. Sign in again |
| **Expected Result** | • State is permanently saved and did not change after fresh sign in |
| **Result** | Ok |
| **Comments** | should it be possible to pause in state „draft"? |

**Test scenario 4: Cancel campaign**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to campaign overview<br>2. Click on cancel button<br>3. Click "ok" on confirmation dialog |
| **Expected Result** | • State of campaign is "canceled". It can no longer be paused/resumed. |
| **Result** | Ok |
| **Comments** | - |

**Test scenario 5: Contact**

| | |
|---|---|
| **Description** | Contact admin via contact-form |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to contact page<br>2. Fill form with data from "Testdata-Sheet 4"<br>3. Submit form |
| **Expected Result** | • Name and Email are prefilled with data from logged in user<br>• After submit, user is redirected to confirmation page<br>• Email to admin is sent |
| **Result** | ~~OK~~ NOK |
| **Comments** | Confirmation page not loaded |

**Test scenario 6: Account**

| | |
|---|---|
| **Description** | Change account details of logged in user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to account page<br>2. Fill form with data from "Testdata-Sheet 3"<br>3. Submit form |
| **Expected Result** | • All account informations are changed according to input |
| **Result** | OK |
| **Comments** | – |

### 2.5.2 Functional Testconcept

Test made with external person working in a marketing division

**Test Run Details**

| | |
|---|---|
| **Date** | 6.6.16 |
| **Software version** | 0e235e8 |
| **Browser version** | Firefox 97.0 |

**Test scenario 1: Create campaign**

| | |
|---|---|
| **Description** | Create campaign as new user |
| **Prerequisites** | Startpage loaded, not logged in, Testuser doesnt exist in database |
| **Steps** | 1. Fill in data from "Testdata-Sheet 1" and submit form<br>2. For each value reported as invalid, fill in data form "Testdata-Sheet 2" and submit form |
| **Expected Result** | • Each red colored value from "Testdata-Sheet 1" is reported invalid.<br>• Preview is shown when image is selected<br>• After input of "Testdata-Sheet 2" data, user is being forwarded to confirmation page<br>• Email is sent to BLINQ-Admin |
| **Result** | NOK |
| **Comments** | Loading modal doesn't go away after wrong input / typos in text |

**Test scenario 2: Login**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", not logged in |
| **Steps** | 1. Sign in with user credentials from "Test scenario 1" |
| **Expected Result** | • User is forwarded to campaign overview page where campaign from "Test scenario 1" is listed with state "draft" |
| **Result** | Ok |
| **Comments** | - |

**Test scenario 3: Pause campaign**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to campaign overview<br>2. Change state of campaign to paused<br>3. Log out<br>4. Sign in again |
| **Expected Result** | • State is permanently saved and did not change after fresh sign in |
| **Result** | Ok |
| **Comments** | [illegible] |

**Test scenario 4: Cancel campaign**

| | |
|---|---|
| **Description** | Login with newly created user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to campaign overview<br>2. Click on cancel button<br>3. Click "ok" on confirmation dialog |
| **Expected Result** | • State of campaign is "canceled". It can no longer be paused/resumed. |
| **Result** | Nok |
| **Comments** | no confirm dialog |

**Test scenario 5: Contact**

| | |
|---|---|
| **Description** | Contact admin via contact-form |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to contact page<br>2. Fill form with data from "Testdata-Sheet 4"<br>3. Submit form |
| **Expected Result** | • Name and Email are prefilled with data from logged in user<br>• After submit, user is redirected to confirmation page<br>• Email to admin is sent |
| **Result** | Ok |
| **Comments** | – |

**Test scenario 6: Account**

| | |
|---|---|
| **Description** | Change account details of logged in user |
| **Prerequisites** | "Test scenario 1", "Test scenario 2" |
| **Steps** | 1. Go to account page<br>2. Fill form with data from "Testdata-Sheet 3"<br>3. Submit form |
| **Expected Result** | • All account informations are changed according to input |
| **Result** | Ok |
| **Comments** | – |

# List of Figures

# Listings

# Abbreviations

**API** Application Platform Interface. 42, 45

**AWS** Amazon Web Services. 56

**DFP** Google DoubleClick for Publishers. 18, 19, 42, 46–48

**FR** Functional Requirement. 35

**NFR** Non-Functional Requirement. 35

**OVF** Open Virtualization Format. 56

**US** User Story. 35

# Bibliography


[Ama] AMAZON: *AWS VM Import*. Website, . – https://aws.amazon.com/ec2/vm-import/; visited 01.05.2016.

[Fou] FOUNDATION, Free S.: *BSD License*. Website, . – https://directory.fsf.org/wiki/License:BSD_4Clause; visited 20.05.2016.

[Gooa] GOOGLE: *Google DFP Documentation*. Website, . – https://developers.google.com/doubleclick-publishers/docs/; visited 05.03.2016.

[Goob] GOOGLE: *Google DFP Documentation*. Website, . – https://developers.google.com/doubleclick-publishers/docs/start#enable_api; visited 25.03.2016.

[IO] IO, Swagger: *Swagger*. Website, . – http://swagger.io/open-source-integrations/; visited 17.03.2016.

[Nie] NIELSEN, Jakob: *10 Usability Heuristics for User Interface Design*. Website, . – https://www.nngroup.com/articles/ten-usability-heuristics/; visited 02.03.2016.

[Tea] TEAM, Django: *Django Documentation*. Website, . – https://docs.djangoproject.com/en/1.9/; visited 05.03.2016.

[Twi] TWITTER: *Twitter Booststrap*. Website, . – http://getbootstrap.com/; visited 22.03.2016.