



Electronic Ticket and Check-in System for Indico Conferences

September 2013

Author:
Bernard Kolobara

Supervisor:
Jose Benito Gonzalez Lopez

CERN openlab Summer Student Report 2013

Project Specification

This project should build on the existing participant registration module of Indico and provide additional functionalities for managing the check-in process. While in small conferences it is easy to keep track of participants with a simple paper list, such techniques become inefficient when the need to scale the process up arises. Therefore Indico's participant registration module would be extended with the functionality to generate electronic tickets. This will allow conference organizers to keep track of attendees after they finish the registration process. As part of this project it is also necessary to develop a mobile application that will be used to scan the electronic tickets, identify the user and mark them as checked in when they arrive at the conference. Additionally Indico's HTTP API would be extended to be used by the mobile application to retrieve data about conferences and attendees.

Abstract

The main goal of this project is to simplify the check-in process for conferences that use the Indico conference management system. This is achieved by extending Indico's core to include electronic ticket generation functionality and developing a mobile application that is used to scan the electronic tickets during the check-in process. Indico's HTTP API is also extended to provide the mobile application with the necessary data.

Acknowledgments

I would like to thank everyone involved in the CERN openlab Summer Student Programme. It was a great experience for me and I learned so much more than I expected. Especially, I would like to thank my supervisor Jose Benito Gonzalez Lopez who was always there when I was stuck on a problem to provide me with advice. And also a big thanks to the Indico team, it was a pleasure working with you all.

Table of Contents

1	Introduction	6
2	Indico	7
2.1	Indico's architecture	7
2.1.1	Flask	7
2.1.2	Request handlers.....	7
2.1.3	Views	7
2.1.4	HTTP API.....	7
3	Check-in system architecture.....	8
3.1	Indico core extensions	8
3.1.1	User interface changes.....	8
3.1.2	Electronic ticket generation	10
3.1.3	E-mail attachment.....	10
3.2	Mobile application	11
3.2.1	PhoneGap.....	11
3.2.2	User interface	11
3.2.3	Architecture.....	12
3.3	HTTP API	13
3.3.1	URL /export/user/<userId>.json.....	13
3.3.2	URL /user/logout.....	13
3.3.3	URL /export/checkin/<registrantId>.json	14
3.3.4	URL /export/registrant/<registrantId>.json.....	14
3.3.5	URL /export/registrants/<confId>.json	14
3.3.6	URL /export/events/<userId>.json	15
4	Conclusion and future work	16

1 Introduction

It is very common nowadays for big international organizations such as CERN to organize events in the order of hundreds of attendees. In such large scale events, keeping track of all those people can be a very tedious and unproductive task. Therefore it's necessary to provide event organizers with better tools that can make the job of managing attendees easier. This will allow them to focus their effort on human-oriented tasks and leave repetitive, simple tasks to the care of computers. Indico tries to provide event managers with tools to work on actually important things, while leaving the task of keeping track of the event workflow to a web-based application.

The goal of this project was to extend Indico with tools to manage the check-in process for conferences. Before, Indico only assisted conference attendees until the registration was finished. Conference organizers would get a list of all the attendees and were left on their own to organize the check-in process. This project tries to build on Indico's existing participant management module by extending it to include additional tools for the check-in process.

This report can be split in two parts. In the first part an overview of Indico and its architecture is given. The second part shows the design and implementation of the systems responsible for the new check-in functionalities. This second part can be additionally split in three sub-parts:

- 1) Indico core extensions – *This part describes the new user interface changes in Indico, the e-ticket generation and the new interface for e-mail attachments.*
- 2) Mobile application – *This part gives an overview of the technologies used to develop the mobile application.*
- 3) HTTP API – *In this part an overview of the HTTP API used by the mobile application is given.*

This report ends with a conclusion and information about future work that can be done to further improve the check-in system.

All code developed for this project is available online on GitHub^{1 2}.

¹ <https://github.com/bkolobara/indico/tree/1345-e-ticket-module>

² <https://github.com/bkolobara/indico-check-in-mobile>

2 Indico

Indico (Integrated Digital Conferences) is a web-based, world-wide, multi-platform content management system and event agenda. It is also the long term archiving tool for documents and metadata related to conferences, meetings and lectures at CERN. Apart from CERN Indico is also used in over 100 other institutions world-wide. The Indico software was originally developed in the framework of the EU InDiCo project. Nowadays, Indico is free software licensed under terms of the GNU General Public License.

2.1 Indico's architecture

Understanding the basics of Indico's architecture will allow us to easier understand the modifications made to Indico during this project. Indico is a really big application and structured in many layers. To provide an overview of all layers would be a too big undertaking for this document, so we will only focus on those parts that are touched in this project.

2.1.1 Flask

Flask is used by Indico for routing, session and request handling. It provides the foundation on which all other components build on. Blueprints are used to define routes. All blueprints are located in the folder *indico/web/flask/blueprint*. Depending on their function rules are put in different files. And this is the place where all URL rules for this project are defined.

2.1.2 Request handlers

Most of the request handlers are located in the *folder indico/MaKaC/webinterface/rh*. But indico is trying to move to another folder structure where the request handlers are defined in *indico/web/handlers*. This is why all the new request handlers for this project are created in this location.

2.1.3 Views

Similar to request handlers, Indico is trying with newer development to change the location where all views are located from *indico/MaKaC/webinterface/pages* to *indico/web/views*. The new location is used by this project.

2.1.4 HTTP API

The purpose of the HTTP API is to expose Indico's data to external applications. In this project it is used to provide conference data to the mobile application. All files used for the HTTP API are located in *indico/web/http_api* folder. The HTTP API doesn't use flask to provide URL routing, it uses a method based on regular expressions.

3 Check-in system architecture

The check-in system consists of many components but they can be split in three main parts. The first part extends the Indico core with additional tools for managing registrants, generating e-tickets and sending e-tickets by e-mail. The second part is more independent and represents the mobile application. The third part is an extension of the existing HTTP API in Indico to provide a way for the mobile application to communicate with indico.

3.1 Indico core extensions

Indico’s core was extended with new views to provide conference managers with additional tools for managing registrants, and allowing registrants to download their e-ticket in PDF format. The already existing PDF engine was extended to generate the e-tickets. And also the e-mail sending in Indico was partly rewritten to allow Indico to attach documents to the e-mails.

3.1.1 User interface changes

We will first look at the additions from the perspective of a conference manager. The first change we will look at is the new e-ticket tab in the Registration section.



For now this tab contains only the option to enable or disable the e-ticket system for the conference. The idea is to additionally extend this tab to have the possibility to change the layout and design of the e-ticket for more personalisation. This was not implemented because of time constraints.

After enabling the e-ticket system the manager has the possibility to enable fields in the Registrants tab that show if the registered users have already checked in into the conference and the check-in date.



Showing 2 registrants

Columns to display | Static URL for this result

Add new	Remove	Check in	Email	Print Badges	Attachments	Show stats	Export
			Phone	Country	Checked in	Check in Date	
			ADMINISTRATOR, Admin	ANGOLA	Yes	22 Aug 2013 14:15	
			DOBARA, Bernard	+38598577444	CROATIA	No	-

This allows the conference manager to keep track of users directly from the web interface. A Check in button was also added to this tab, where one or more selected user can be marked as checked in.


When the manager selects a registrant by clicking on him/her more information about this registrant is displayed. To this registrant information screen an additional tab E-Ticket was added where the conference manager can download the ticket for the selected registrant and mark or unmark them as checked in.

Registrant

SNOW, Jon

Go back to: [Registrants list](#)

Main **E-Ticket**

 [Download registrant's ticket](#)

Checked in: **Yes**

Check in date: 4 Sep 2013 14:31:28

From the regular user perspective not many things have changed. The only addition is a download link for the e-ticket after registration.

Indico developer conf

19-31 August 2013
CERN
Europe/Zurich timezone

Overview

Scientific Programme

Timetable

Contribution List

Author List

My Conference

Registration

└ Modify my Registration

└ [Download e-ticket](#)

Registration Form

Thank you for completing the registration form which has been sent to the organisers for their attention.

 [Download ticket](#)

Registrant ID: 1

Registration date: 23-August-2013 11:09

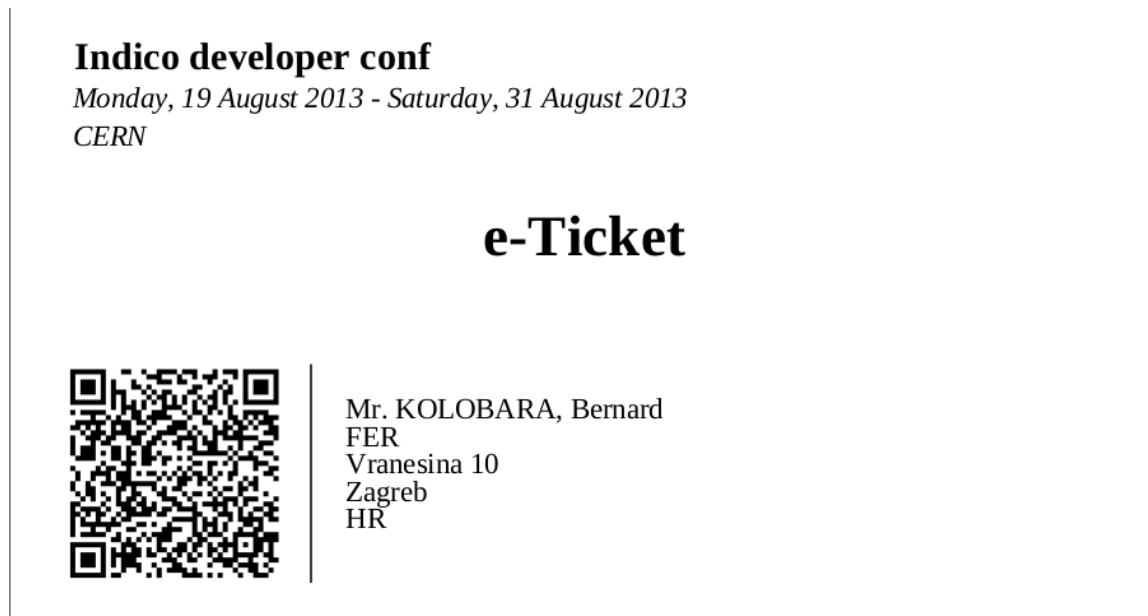
Personal Data

Title	Mr.
First Name	Bernard
Surname	Kolobara
Position	Student
Institution	FER
Address	Vranesina 10
City	Zagreb
Country	CROATIA
Phone	+38598577444
Fax	+41355555
Email	bernard.kolobara@cern.ch
Personal homepage	bkolobara.info

If the user is registered in Indico he will additionally have the option to download the e-ticket from the side menu when he comes back to the conference site.

3.1.2 Electronic ticket generation

The electronic ticket is a PDF document containing information about the attendee, conference and a QR code that can be later used to uniquely identify the attendee.



Indico already used the *reportlab* library for generating PDF documents in other parts of the application. The existing PDF module was extended to include e-ticket generation. A new class (*TicketToPDF*) was added to the existing file *indico/MaKaC/PDFinterface/conference.py*.

The *TicketToPDF* class constructor receives two arguments: the conference object and the registrant object. This class also inherits the method *getPDFBin* that can be used to get the PDF binary blob for the ticket.

The QR code is a JSON structure and contains the following data:

- application – *The application name. Currently hardcoded to indico. The application name is necessary for a better user experience. If the ticket is scanned by a regular QR code scanner, the user gets information about the application and not just a cryptic id.*
- target – *Conference id.*
- id – *Registrant id.*
- secret – *A uniquely generated hash to offer protection against faking tickets by knowing the conference id and guessing the registrant id.*

3.1.3 E-mail attachment

Indico also already provided the functionality to send a confirmation e-mail to newly registered users, but this interface didn't have the flexibility to attach documents to the e-

mails that are sent. Therefore the interface was partly rewritten to offer backwards compatibility and to add the option to attach files to e-mails.

The following example demonstrates the usage of the new e-mail interface.

```
attachment = {}
attachment['name'] = 'Ticket.pdf'
attachment['binary'] = pdf.getPDFBin()

email = self._regForm.getNotification()\
        .createEmailNewRegistrant(self._regForm, rp)
email['attachments'] = [attachment]
GenericMailer.send(email)
```

This example was taken from the code responsible for attaching tickets to registration e-mails. As we can see the attachment is a dictionary containing the filename and the content of the file to be sent.

3.2 Mobile application

The mobile application was developed using web technologies (HTML and JavaScript) and PhoneGap is used to compile it to a native application and provide access to native APIs like the camera one.

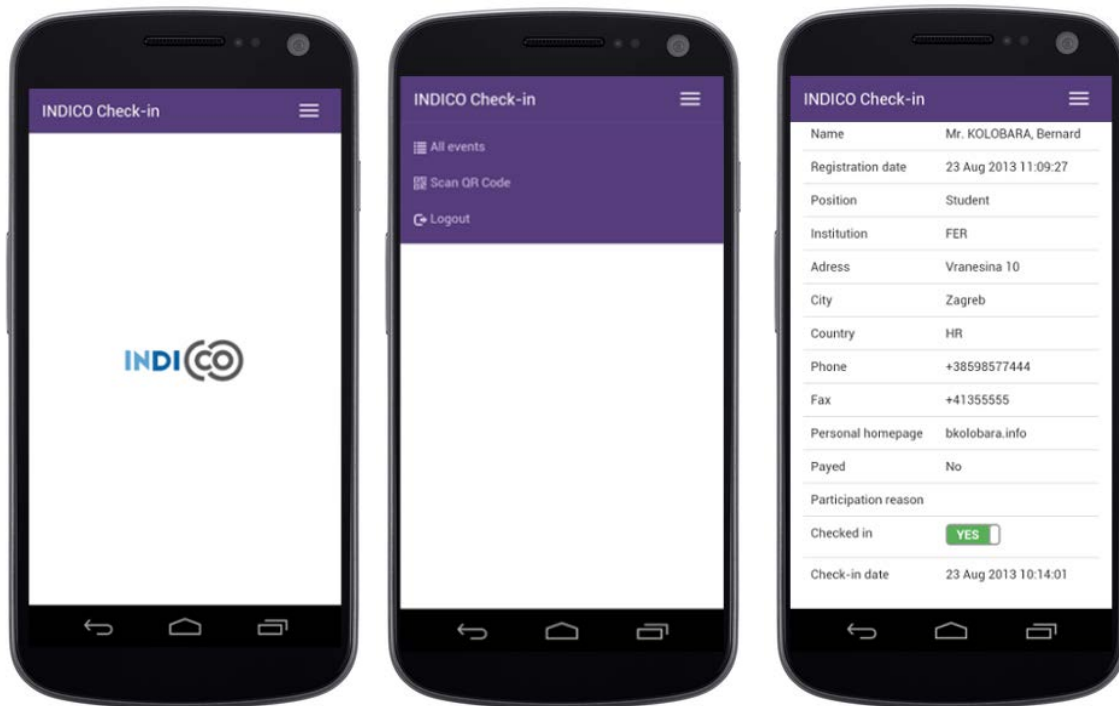
3.2.1 PhoneGap

PhoneGap is a mobile development framework produced by Nitobi, purchased by Adobe Systems. It enables software programmers to build applications for mobile devices using JavaScript, HTML5, and CSS3, instead of device-specific languages such as Objective-C or Java. The resulting applications are hybrid, meaning that they are neither truly native (because all layout rendering is done via web views instead of the platform's native UI framework) nor purely web-based (because they are not just web apps, but are packaged as apps for distribution and have access to native device APIs).

Using PhoneGap for this project has the benefit that the created application is not limited to one platform. And using web technologies allows for a faster development cycle.

3.2.2 User interface

Bootstrap 3 was chosen as the CSS framework to simplify the development and provide a nice looking user interface optimized for mobile screen sizes. The following screenshots are showing the user interface for the mobile application:



3.2.3 Architecture

Angular.js is used as the JavaScript framework for the mobile application. Because of that most of the design choices are influenced by best practices in the Angular.js community.

Only the `www` folder is included in the version control system, other folders are platform dependent and are generated by PhoneGap. The `www` folder contains the following files:

- `config.js` – Main configuration file for the application where the base URL, consumer and secret keys are defined.
- `config.xml` – Configuration file for PhoneGap containing many fields. Most of them have the default values. One of the more important fields is `access origin = "*" , that gives the application privileges to access all domains.`
- `icon.png` – The icon for the application.
- `index.html` – The starting point of the application.
- `spec.html` – File for testing (currently not used).

Though the main entrance into our application is the `index.html` file, it contains only enough code to call the initialize function of PhoneGap that is located in the `www/js/app.js` file. This function will bootstrap Angular.js when PhoneGap has started the application and is ready.

All Angular.js routes are defined in the `www/js/router.js` file. When a route is matching the URL an appropriate partial view is loaded from the `www/partials` folder and the matching controller is triggered in the `www/js/controllers.js` file.

Most of the application's logic is encapsulated as an Angular.js service in the `www/js/services.js` file. The service is exposed under the name `OAuth` and provides many functions like `authenticate`, `logout`, `ifAuthenticated`, `getRegistrantsForEvent`, etc.

The following code shows an example of using the `OAuth` service:

```
OAuth.getRegistrantsForEvent($scope.event_id, function (result) {
    $scope.registrants = result.registrants;
    $scope.$apply();
});
```

This example is taken from the `www/js/controller.js` file and shows how the service encapsulates all the complexity in a simple function that takes an event id and callback function.

Other files and folders worth mentioning are:

- `www/css` – Contains all the CSS files. The only CSS file that is not generated is the `index.css` file.
- `www/fonts` – Contains the glyph icon fonts for twitter bootstrap.
- `www/img` – Contains all the images. For now there is only one image `spinner.gif`.
- `www/res/screen` – Contains splash screens for different platforms.
- `www/spec` – Contains all the tests (currently not used).
- `www/js/lib` – Contains JavaScript library files like `angular.js`, `bootstrap.js`, etc.

3.3 HTTP API

The HTTP API acts as a communication channel between the mobile application and the Indico core part. All communication is done using the JSON protocol.

The HTTP API is defined in the `indico/web/http_api/eticket.py` file. This file contains 5 classes and every class is mapped to a URL except the `RegistrantBaseHook` class, which is used to abstract some security checks that are inherited by the other classes. Now follows a more detailed overview of the URLs used by the mobile application.

3.3.1 URL `/export/user/<userId>.json`

This URL is not defined in the `indico/web/http_api/eticket.py` file, it already existed before. By providing the user id this URL will return information about the user like username, full name, etc. It is used by the mobile application to get the username of the logged in conference manager and it is also used to make a dummy call to the server to check if the user is still logged in after awaking the mobile application from sleep. If this URL returns `401 Unauthorized` we know that the user is logged out.

3.3.2 URL `/user/logout`

This URL is special; it is not a HTTP API URL like the other starting with the prefix `export`. This URL is provided by Indico to logout the currently logged in user. It's included here because it's used as a trick to delete the cookies from the mobile application browser used by Phonegap. If we don't make a call to this URL after the user

logs out he will be logged in again when PhoneGap starts his browser and sends an *OAuth* request. We don't have direct access to PhoneGap's browser so we need Indico to delete those cookies for us.

3.3.3 URL `/export/checkin/<registrantId>.json`

By providing the registrant id and the following GET request arguments:

- `target` – *Conference id.*
- `secret` – *Unique secret generated for the ticket.*
- `checked_in` – *New value for the check in status.*

this API call will change the checked in status for that registrant. This URL will return *success: true* and *checkin_date* if successful.

3.3.4 URL `/export/registrant/<registrantId>.json`

This URL takes the same arguments as the `/export/checkin` except the *checked_in* value and returns the following data about the registrant:

- `registrantId` – *Id of the registrant (the same one that was submitted).*
- `fullName` – *Full name of the registrant.*
- `checkedIn` – *Check in status (true or false).*
- `checkInDate` – *Check in date.*
- `registrationDate` – *Registration date.*
- `position` – *Position in the institution.*
- `institution`
- `address`
- `city`
- `country`
- `phone`
- `email`
- `payed` – *If the registrant has paid for the conference (true or false).*

Not all fields need to be returned. If a field is omitted the mobile application will not display it.

3.3.5 URL `/export/registrants/<confId>.json`

Returns a list of registrants for the requested conference as *registrants*. Every registrant in the list contains the following data:

- `id` – *Registrant id.*
- `full_name`
- `checked_in`
- `secret` – *The same secret generated on the registrant's ticket.*

3.3.6 URL `/export/events/<userId>.json`

This URL will return all the conferences that this user is managing. The returned list has the name *events* and every event contains the following data:

- *id* – Conference id.
- *title* – Conference title.
- *date* – Conference start date.
- *registrants* – Number of registrants for this conference.

4 Conclusion and future work

The main parts of the check-in system are done. Those additional tools should significantly simplify check-in management of attendees for conference organizers. There is still room for further improvements. The e-ticket generation can be extended to allow conference organizers to include simple design elements on tickets like the conference logo. Giving the possibility to personalise the ticket per conference would make the e-ticket system more appealing for a wider audience. During the work on this project other departments at CERN showed interest in using the mobile application for their needs. Therefore the mobile application could be made more generic to support other systems. Some steps have been already taken in this direction, like making the QR code more general by including the application name in it, so that the mobile application is capable of differentiating between systems.