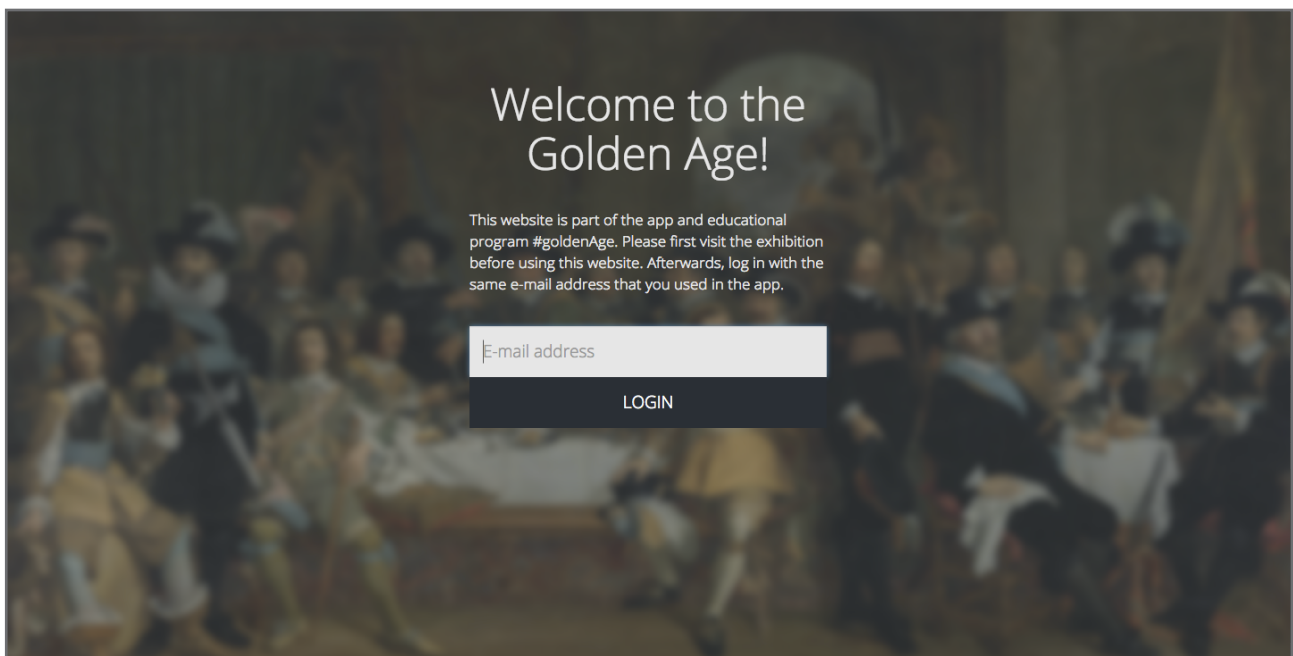


Hollanders van de Gouden Eeuw backend code

This repository contains the backend code for the project Hollanders van de Gouden Eeuw which is an app which uses Bluetooth beacons to provide contextual information to museum visitors.

To try to reach a young audience, the information is presented to the app user as information “cards” which look like social media status updates.

The app has been built in native iOS and Android versions. When visitors return home, they can login onto the web frontend to re-read the stories they have experienced.



Visit hollandersvandegoudeneeuw.nl for more information about the app. To login with your #goldenAge account visit <http://login.hollandersvandegoudeneeuw.nl>

Installation

This project is built on top of the Zotonic web framework. As such, since Zotonic supports multiple websites, this repository contains the code for a single Zotonic website.

To install the code in Zotonic, do the following:

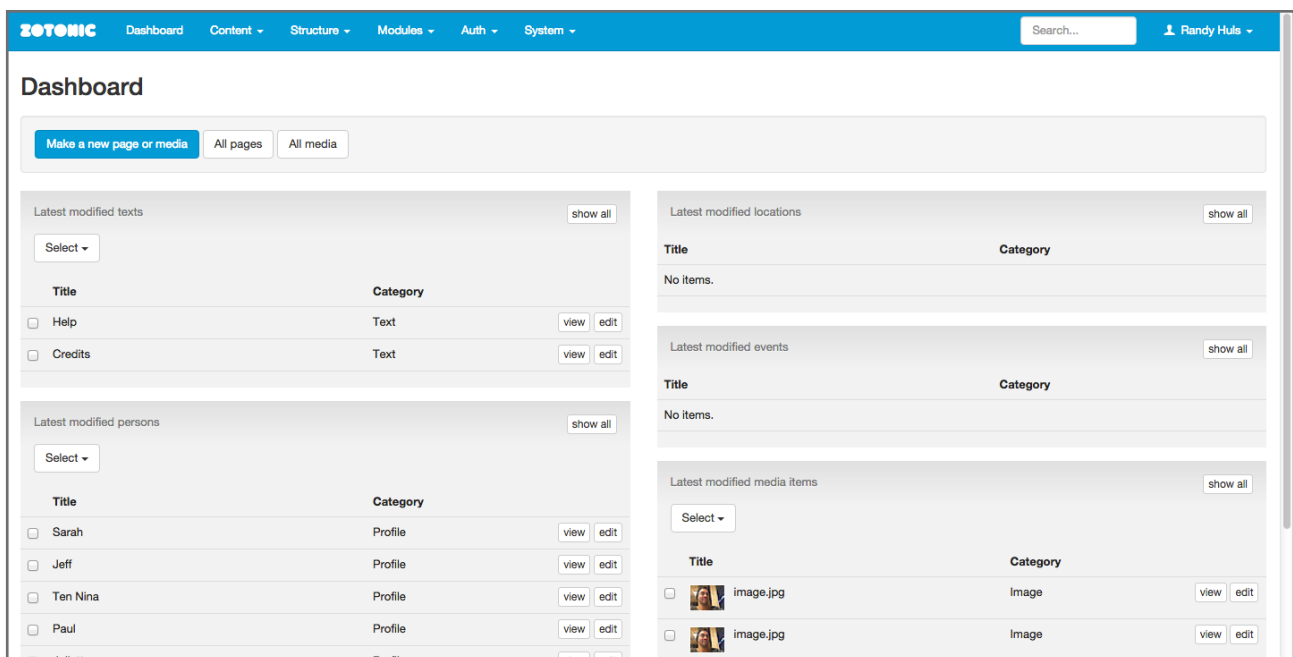
- Follow the generic installation instructions to get Zotonic running.
- Install this repository by cloning it into the `$ZOTONIC/user/sites/goldenage` folder.
- Copy the `config.in` file to a file named `config` in the same directory, and edit it to provide the required values (most importantly, make sure the database credentials are valid, and that the

hostname property resolves to a valid domain (it defaults to goldenage.dev; you can add this to your /etc/hosts file to get a local dev environment running).

- Now, type make in the zotonic folder and start Zotonic (run \$ZOTONIC/bin/zotonic debug). You should see messages scroll by, including a message like this:

```
11:08:51.045 [info] http://goldenage.dev:8000 - running
```

- Go to <http://goldenage.dev:8000/admin> to open the backend's CMS.



The Zotonic dashboard

Data model

In Zotonic, every thing in the system is a “page”. Each page has a category. Pages can link to each other using “page connections” (also called “edges”). Edges have a “label” on them, also called the “predicate”. This datamodel is very powerful and can be used to model a lot of data that you otherwise would use a normal relational database for.

In the goldenage backend, we use these categories to model the relation between cards, stories and chapters.

The following page categories are defined:

- Story - the different stories the user can choose from in the app. Each museum experience starts with the audience choosing a story.
- Chapter - stories consist of chapters. While the visitor is walking through the museum, only one chapter is activated. The active chapter is defined by the presence of beacons in the museum.
- Beacon A single iBeacon instance. It has special edit-fields in the admin to allow the CMS user

to edit the beacon's UUID, major and minor numbers.

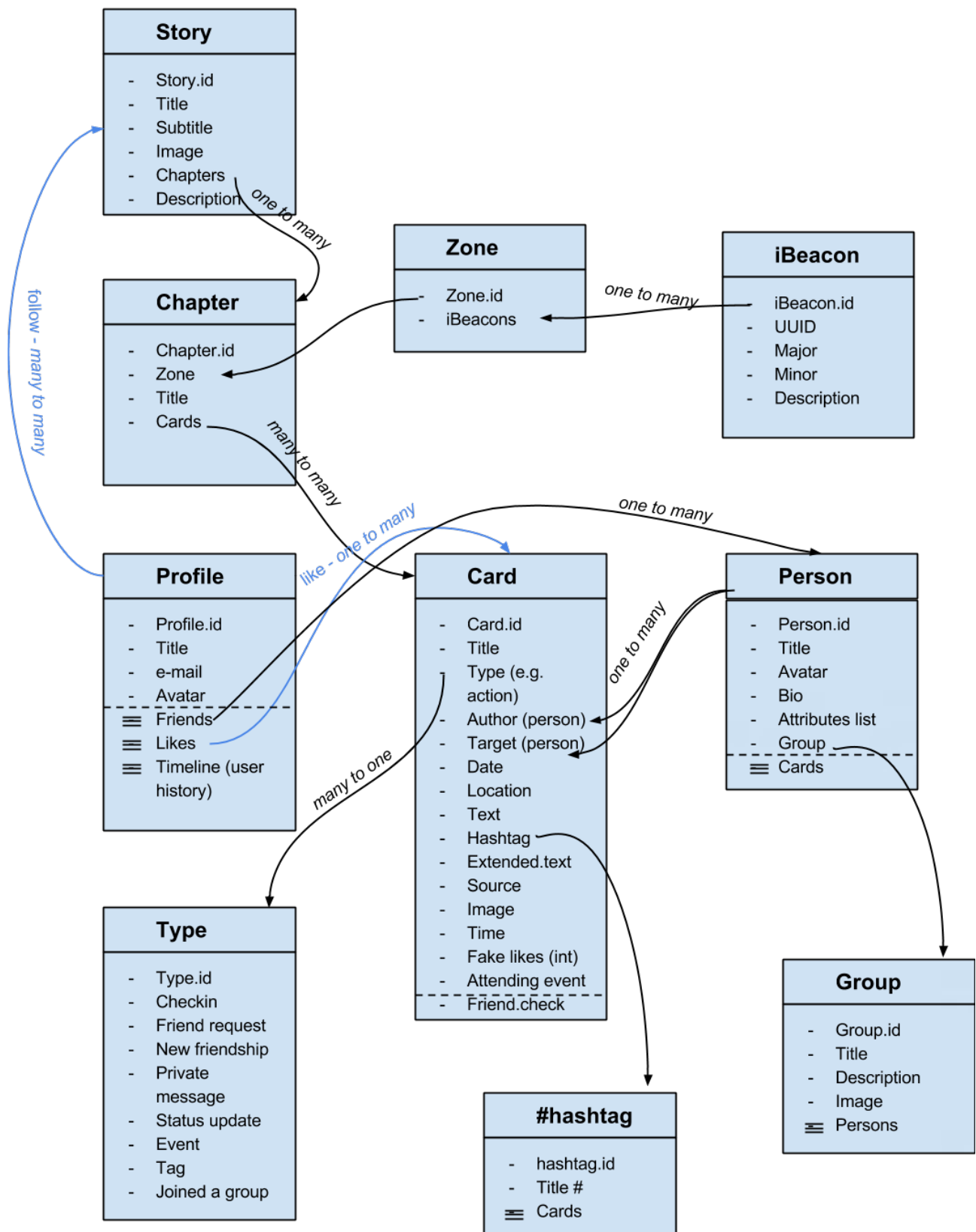
- Zone - To signify which chapter belongs to which beacon, we group the beacons into a zone.
- Card - Each chapter contains an ordered list of cards. Each card represents a status update. The cards have a delay property that dictates how much time (in seconds) it takes until the next card will be presented to the user.

The left screenshot shows the 'Make a new page' dialog with the following fields: 'Page title' (Beacon 5), 'Category' (Beacon), and 'Published' (checked). The right screenshot shows the same dialog with the 'Category' dropdown menu open, displaying a list of categories including 'Uncategorized', 'Golden Age', 'Story', 'Chapter', 'Zone', 'Beacon', 'Card', and various card types like 'Card - Checkin', 'Card - Friend request', 'Card - New friendship', 'Card - Private message', 'Card - Status update', 'Card - Attending event', 'Card - Tag in picture', 'Card - Joined group', 'Card - Photo prompt', 'Person group', 'Hashtag', 'Text', 'Person', 'Historical person', 'Profile', 'Location', 'Event', and 'Media'.

A new 'page' is defined by its category, in this case a 'Beacon'.

The screenshot shows the ZOTONIC dashboard with the 'Beacon 5' editing page. The sidebar on the left contains navigation links: Dashboard, Content, Structure, Modules, Auth, and System. The main content area has a header with 'editing beacon: Beacon 5' and a search bar. Below the header, there are two main sections: 'Basic' and 'Content'. The 'Basic' section contains fields for 'Title' (Beacon 5), 'Beacon UUID' (B9407F30-F5F8-466E-AFF9-2556B57FE6D), 'Beacon major' (58028), and 'Beacon minor' (44583). The 'Content' section contains a rich text editor with a toolbar. On the right side, there is a 'Publish this page' section with buttons for 'Save', 'Save & view', and 'Cancel'. Below this, there are checkboxes for 'Published' (checked), 'Featured', and 'Protect'. There are also buttons for 'Delete' and 'Duplicate'. Further down, there are sections for 'Access control', 'Publication period', 'Date range', and 'Page connections'. The 'Page connections' section shows 'Author' and a button to '+ add a connection'. At the bottom, there is a button to 'View all referers'.

But a new 'page' can be anything ranging from a 'Story' or 'Card' to a 'Beacon' or 'Zone'



The complete data model showing the various relations

API call overview

The main thing that this website exposes, is a collection of API calls which the Android and iOS apps use to retrieve their configuration and present the information cards to the user. Below is a description of those calls.

The source code for all of these calls lives in the **services/** subdirectory.

Authorization

Add **Authorization: E-mail foo@bar.com** header to each request to do stuff as the user with that e-mail address. Before doing this you need to “login” so that the system knows about that e-mail address.

POST /api/goldenage/login

Logs a user in / creates an account.

POST parameters:

- email
- name
- avatar

When posting an avatar, you need to do a multipart encoded form post. Return value:

```
{
  "category": "profile",
  "email": "foo@bar.com",
  "id": 361,
  "image": null,
  "title": "Arjan",
  "likes": [459],
  "cardseen": [143, 459, 125]
}
```

image is an URL which contains the avatar image, or *null*.

likes is the list of card ids that the user likes. Handy for when you want to display the state of the 'heard' without getting all liked cards. note I mix the term "like" and "favorite", sorry about that.

cardseen is the list of card ids that the user already has seen. When loading a story, the initial timeline should be filled with the "seen" cards that participate in that story.

POST /api/goldenage/favorite

Adds a card to the users's list of favorites.

Parameters:

- card_id (required, integer)
- delete (optional, boolean)

When **delete=true**, the favorite gets removed instead of added.

GET /api/goldenage/favorites

Returns a favorites object for the user, which is a JSON object with 'cards' and 'persons' entries.

GET /api/goldenage/stories

Returns a list of all available stories.

```
[
{
  "id": 2661,
  "category": "story",
  "title": "Joan Huydecoper story (EN)",
  "subtitle": "#storytelling #flinckapp",
  "summary": "This story is part of the painting 'Company of Captain Joan Huydecoper and Lieutenant Frans van Waveren Celebrating the Treaty of Münster' of Govert Flinck.",
  "publication_start": "2015-09-04T20:01:00+00:00",
  "lang": "en",
  "image": "http://goldenage.miraclethings.nl/image/2015/9/3/tag_govertflinck-97587513.png%28600x%29%2867A6097321A388FFE4B1F8FD1E3805A9%29.jpg"
}, ... ]
```

Live example: <http://goldenage.miraclethings.nl/api/goldenage/stories>

GET /api/goldenage/storydata

Returns all story data for a single story.

Live example: <http://goldenage.miraclethings.nl/api/goldenage/storydata?id=2661>

GET /api/goldenage/userinfo

Return information about the current user. The response is identical to the “login” response.

POST /api/goldenage/cardseen

Mark the given card as ‘seen’ in the backend.

Parameters:

- card_id (required, integer)
- delete (optional, boolean)


When ***delete=true***, the “seen” state gets removed instead of added.

GET /api/goldenage/personinfo

Return information about a historical person

```
{
  "category": "historical_person",
  "title": "Jan van Grindbergen",
  "subtitle": "Kapitein der Gravenlanden",
  "id": 361,
  "image": "http://...",
  "cards": [ list of cards of this person ],
  "persons": { persons referenced by the person's cards },
  "groups": [ list of groups ]
}
```

ZOTONIC
Dashboard
Content
Structure
Modules
Auth
System



editing historical person:

Rembrandt van Rijn

Basic

Title

Rembrandt van Rijn

Subtitle (job description)

Kunstschilder

Summary (biography)

Rembrandt is een van de belangrijkste Hollandse meesters van de Gouden Eeuw. Zijn bekendste werk is De Nachtwacht (1642). Hoewel hij nooit in Italië is geweest, wordt zijn werk tot de barok gerekend. Zijn privé leven kent vele ups en downs, waaronder een faillissement. Hij leeft ver boven zijn stand, koopt veel kunst, wapens en exotische voorwerpen.

Historical person details

Key	Value	
Geboren	Leiden 1606	✕
Overleden	Amsterdam 1669	✕
Getrouwd met	Saskia van Uylenburgh	✕

Add row

Person name

First	Middle	Sur. prefix	Surname
-------	--------	-------------	---------

Page connections
?
—


Author

+ add a connection

Document

+ add a connection

Part of group


Schilders Person group
✕

+ add a connection

Keyword

+ add a connection

View all referrers

Historical persons have more detailed biographies. They can also be part of a group.