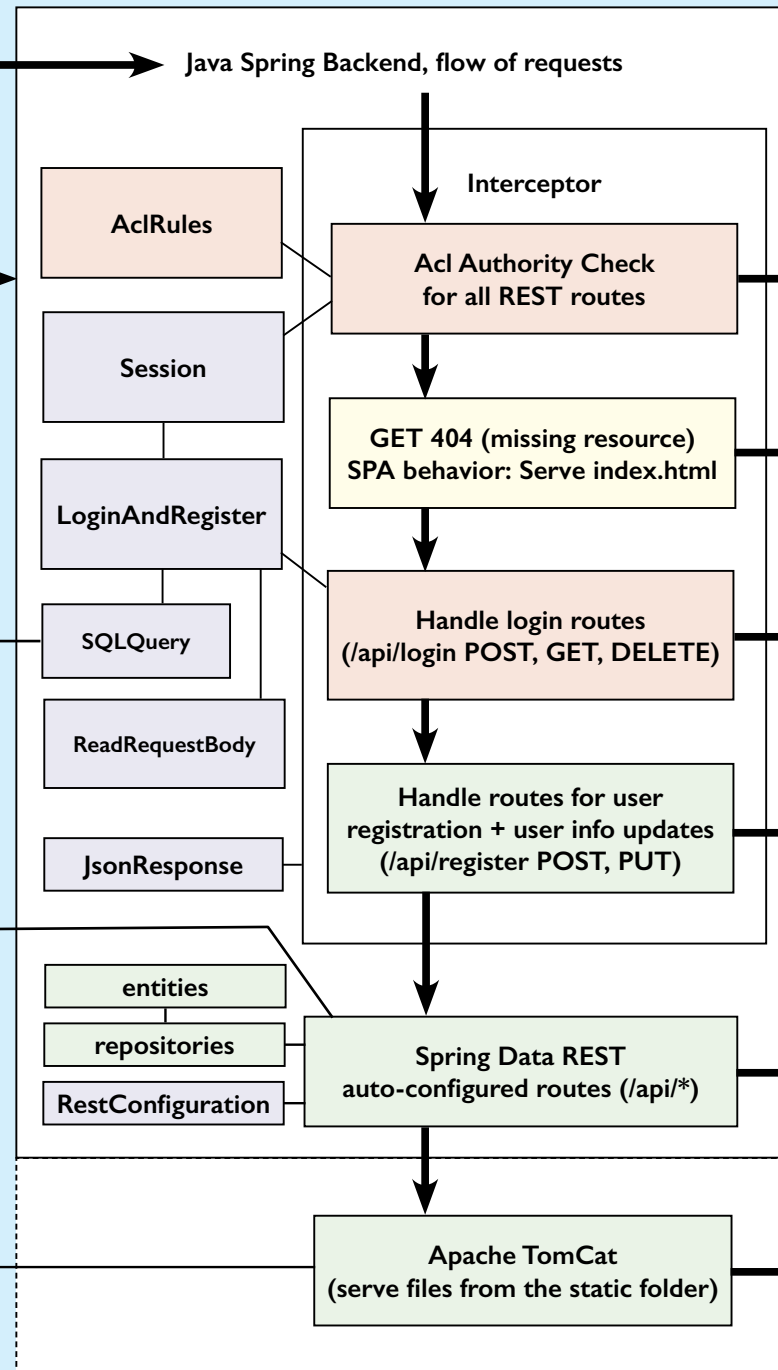
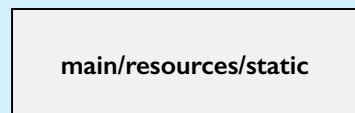
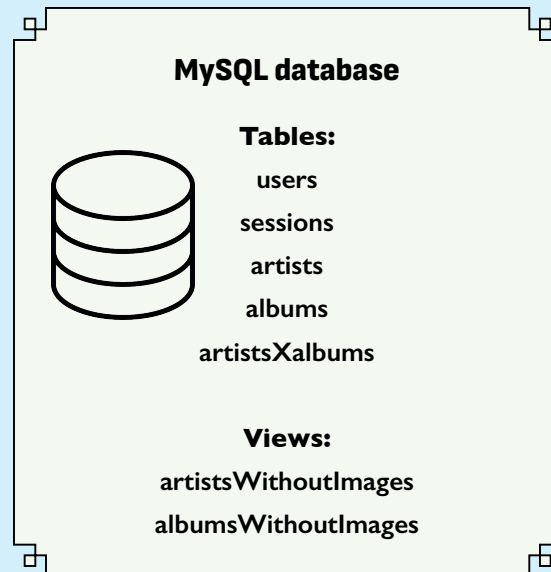
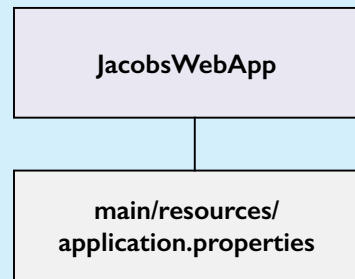


Jacob's Backend: Flow, classes and parts

HTTP request



Possible
HTTP responses

{error: not allowed}

index.html

JSON data {user info etc}

JSON data {user info etc}

JSON data {from the DB}
including images

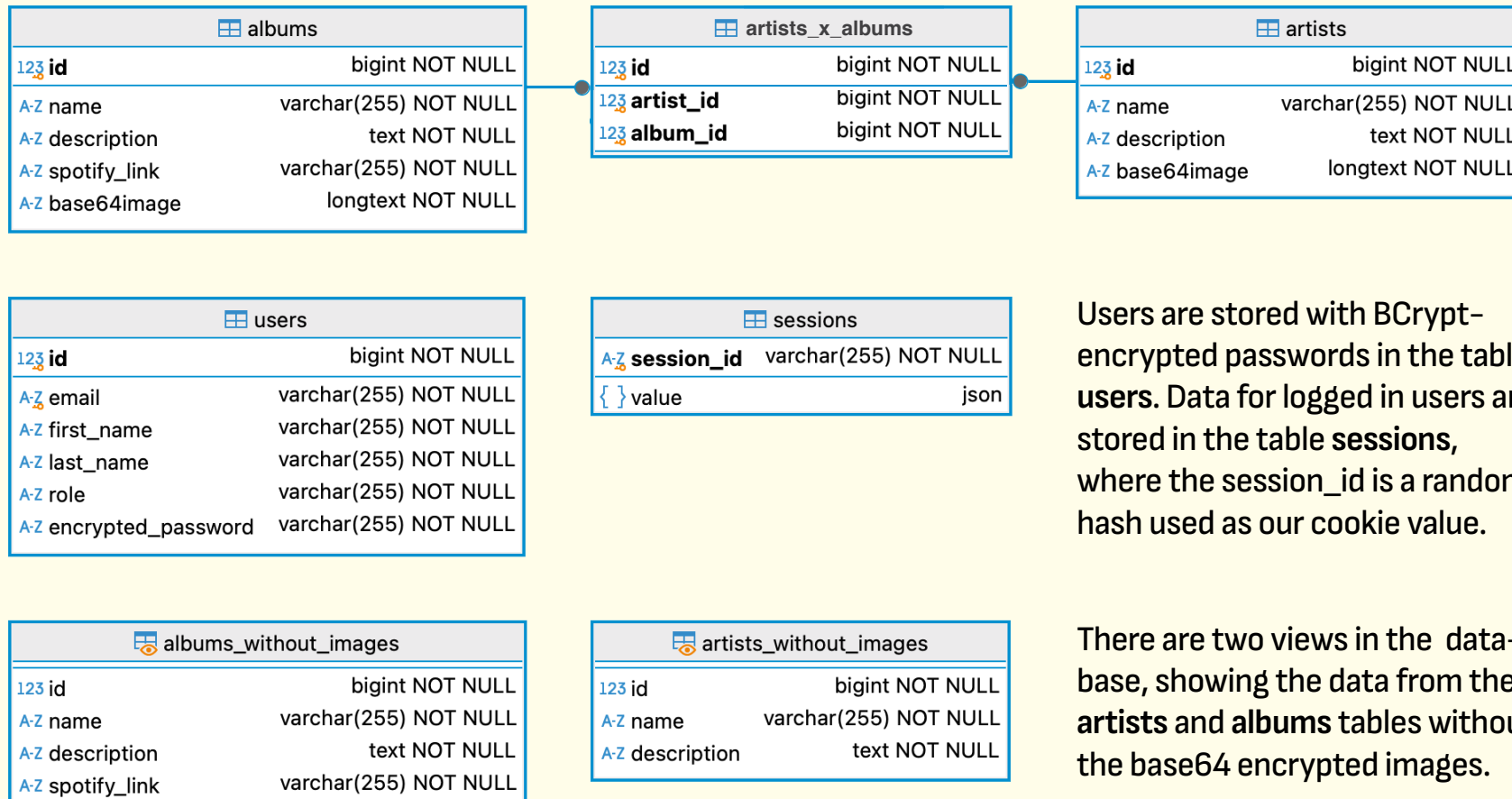
Frontend assets:
HTML, CSS, JS, icons etc

Jacob's MySQL database: ER Diagram



The only relation we have is a many-to-many relationship between albums and artists:

- An album is made by one or several artists, an artist can make one to many albums
- We solve this many-to-many relation in the classic SQL way, by introducing a join table, `artistsXalbums`.



Users are stored with BCrypt-encrypted passwords in the table **users**. Data for logged in users are stored in the table **sessions**, where the `session_id` is a random hash used as our cookie value.

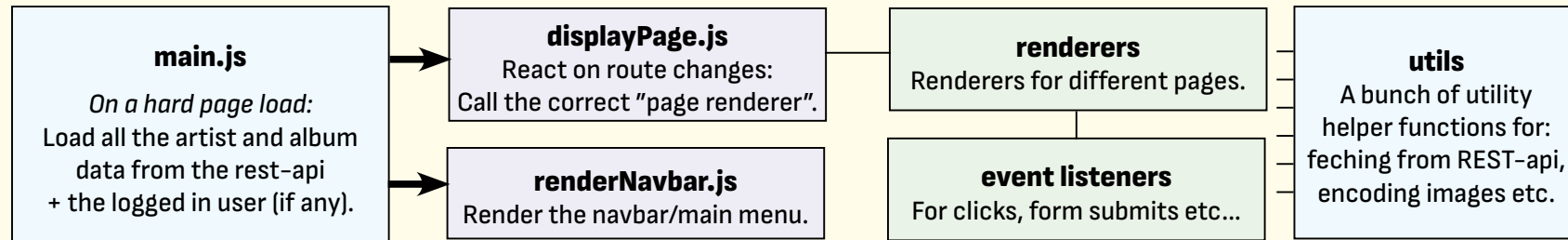
There are two views in the database, showing the data from the **artists** and **albums** tables without the base64 encrypted images.

Jacob's Backend: REST API Routes

| Request method | Route | Access control <i>(by user roles)</i> |
|----------------|-------------------------------|---------------------------------------|
| POST | /api/register | visitor |
| PUT | /api/register | user |
| POST | /api/login | visitor |
| GET | /api/login | visitor, user |
| DELETE | /api/login | user |
| POST | /api/artists | user |
| GET | /api/artists | visitor, user |
| GET | /api/artists/:id | visitor, user |
| PUT | /api/artists/:id | user |
| DELETE | /api/artists/:id | user |
| GET | /api/artistsWithoutImages | visitor, user |
| GET | /api/artistsWithoutImages/:id | visitor, user |
| POST | /api/albums | user |
| GET | /api/albums | visitor, user |
| GET | /api/albums/:id | visitor, user |
| PUT | /api/albums/:id | user |
| DELETE | /api/albums/:id | user |
| GET | /api/albumsWithoutImages | visitor, user |
| GET | /api/albumsWithoutImages/:id | visitor, user |
| GET | /artistXalbums | visitor, user |
| POST | /artistXalbums | user |
| DELETE | /artistXalbums/:id | user |

Jacob's Frontend: Quick overview

The frontend is an SPA built with "vanilla JS" (JavaScript without any frontend framework). Two external JS libraries are used: The JS part of the Bootstrap CSS library and Marked.js (*conversion of markdown to html*). The Bootstrap CSS library is used to accomplish a modern responsive layout in conjunction with my own CSS.



SPA (Single Page Application): Frontend routing

The frontend is an SPA, which takes care of the routing of a number of different GET routes, via JavaScript running in the browser, without doing any hard page reloads between route changes.

| Menu item / Link | Shown for | GET route | Description |
|---------------------------------|--------------------------|-------------------|------------------------------|
| Artists | everyone | / | List of artists |
| Albums | everyone | /albums | List of albums |
| Add an artist | logged in users | /add-artist | Form for adding an artist |
| Add an album | logged in users | /add-album | Form for adding an album |
| About | everyone | /about | Info about the application |
| Register | visitors (not logged in) | /register | Form for registering a user |
| [name of logged in user] | logged in users | /register | Form for updating user info |
| [link from each artist in list] | everyone | /artists-info/:id | Details about an artist |
| [link from each album in list] | everyone | /album-info/:id | Details about an album |
| [link from each artist in list] | logged in users | /edit-artist/:id | Form for editing artist info |
| [link from each album in list] | logged in usrs | /edit-album/:id | Form for editing album info |