Software Engineering for Web Applications

Echo,

a Twitter-like web application for knowing your districts

Authors

Román Rey Pedrero *183672*

Gary Ulloa Rodríguez *183694*

1 Our application

Echo is a Twitter-like web application which focuses on the interactions between neighbours in their favourite districts. Echo-s users can check the shared voices in their districts and share their own voice, they can follow those people which want to hear about.

2 Functionalities

In the following schema it's shown the mandatory and extra requirements.

Type of user	Use case	
	Mandatory Requirements	
Anonymous	Register ✔	Global Timeline ✔
	Login ✔	Profiles (Personal Pages) ✔
Registered	*Same as Anonymous (except Register & Login)*	
	Logout ✔	Delete voice ✔
	Personalized timeline ✔	Edit voice ✔
	Edit Profile ✔	Follow user ✔
	Publish voice ✔	Unfollow user ✔
Admin	*Same as Registered*	
	Delete voices ✔	Delete user ✔
	Edit voices ✔	Edit Profiles ✔
	Extra Functionalities	
All	Search engine for users and districts ★	District Page ★
	Retweet voice ★	User Districts ★

Login

The login form is included on the left navbar always on display, so users can login straight away by giving the right username and password. A correct login redirects you to your user feedback timeline.

Register

A user can register by accessing to the register view. The user will have to specify a username and an email, both of which have to be unique. In case they were already registered in the database, an indicative message will be shown, as shown in the image. Pseudonym, country, gender and password are obligatory fields too. Password will have to have to be confirmed. Lastly, an optional user image can be added using the URL of an online image (if not supplied, a default one is assigned).

Global timeline

This is the main page. This means, even if you are not registered you can search through the voices directly. It displays the latest voices (20 at most), some random users (50 at most) and the lastly voiced districts (10 at most). The districts and user profiles are accessible through their buttons.

User profile

Each user has a profile page, which displays their profile image, pseudonym, biography, their voices and the users this particular user is following. If the profile is not your own, you will be able to see a *Follow!* button (*Unfollow* if you already follow this user). If it is your own, you will not see it; nonetheless, you will be see the *Edit Profile* option. Profiles are accessible through the username or user image; in case it is your own you can also access by the *PROFILE* option.

Edit profile

This functionality allows you to alter your own profile. This includes changing the pseudonym, biography and the image URL. The admin has permissions to alter these fields from any user.

Follow user

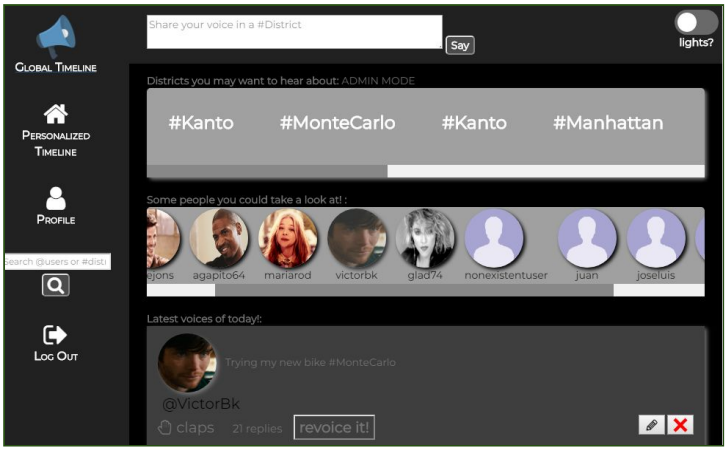
On every user profile (not your own) appears a *Follow!* (*Unfollow*, otherwise). This will append the voices of this user to your personalised timeline.

Delete user

The admin is able to delete a user entirely from the database through the user profile.

Personalised timeline

This view displays your user personalised timeline, meaning it will contain your voices and the voices of those users you follow. It also displays the districts that you use with your voices, giving direct access to their pages.



Publish voice

This functionality is show all over the web once you are logged in. You only have to include a district (using the # symbol). It will not allow to include more than one.

Delete voice

On every voice of your own appears a red cross to indicate you want it deleted. Admins can also perform this action on all voices.

Edit voice

On every voice of your own appears a blue button that opens a field to rewrite that voice. The district has to be specified. No more than one district can be written. The date will remain as the original one, otherwise it would have been the same as deleting and creating a new voice. Admins can also perform this action on all voices.

Search engine

This search bar is accessible through all the web. It is not necessary to be logged in to use it. The @ is used to search a user, the # to search a district. It will return random matching results that give access to matching users/districts.

District echo

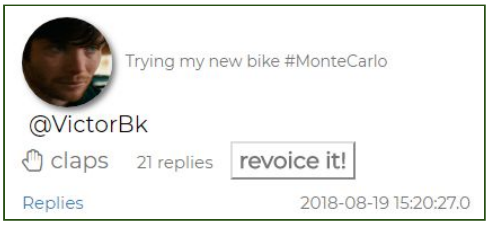
This is the page of belonging to a district. It displays all the voices posted to this district.

User follows district

This feature is done each time a user publishes a voice. It means every user can create a district

Revoice voice

Users can retweet any voice that is not their own. This generates a voice with the user data indicating is a revoice.



3. Model View Controller Schema

We used AJAX to update the following information on the website:

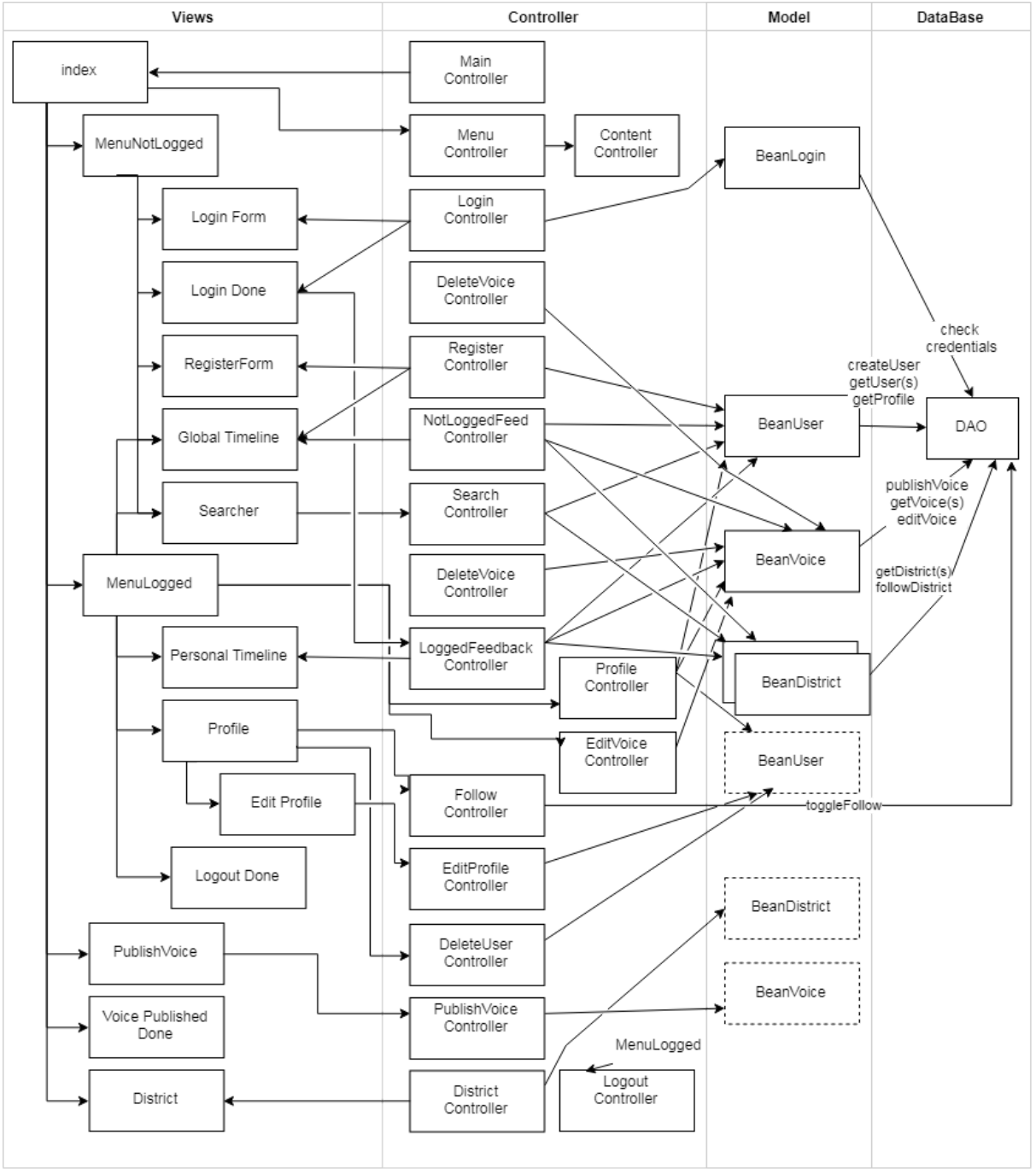
Follow, editVoice,delete voice, editProfile, revoice, delete user, search users & districts, see profile, see district.

All these features required asynchronous communications since just an element of the DOM has to be updated.

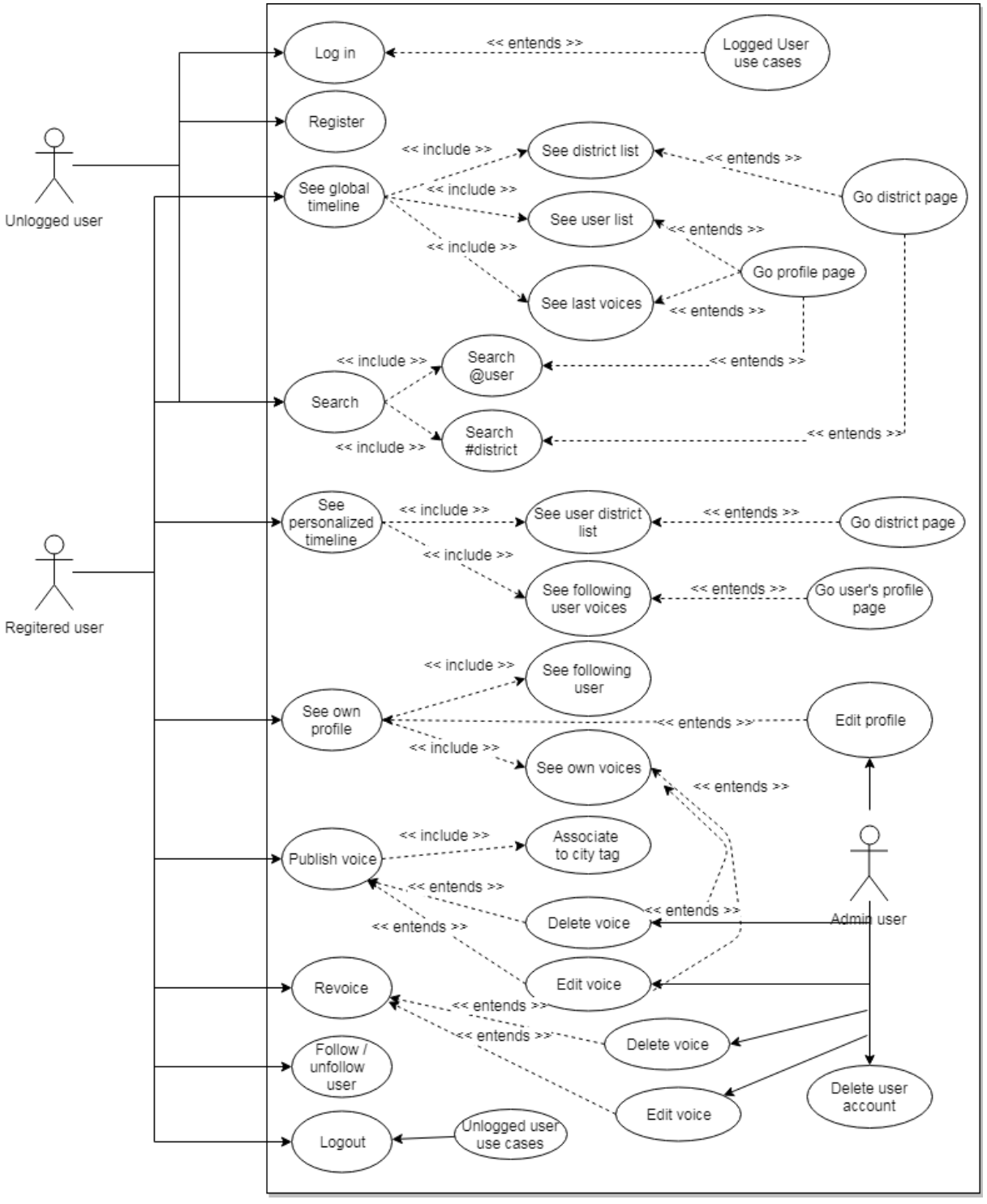
We used synchronous communication for:

Register, login, logout and global & personalized timeline.

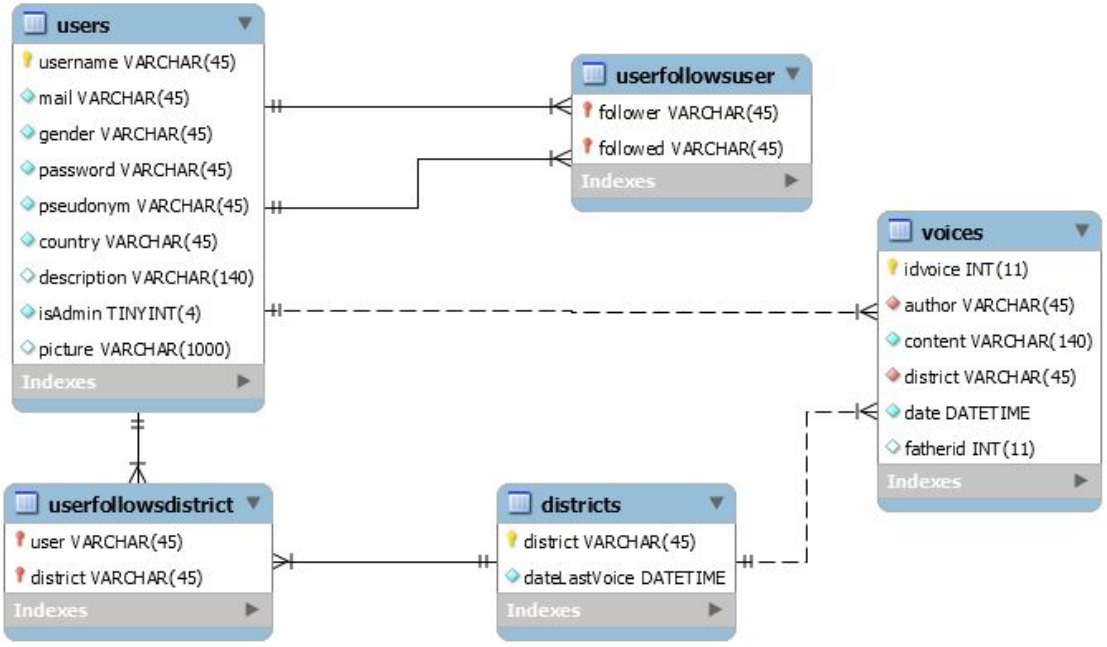
This feature need all the content of the page to be updated



4. UML use case diagram



5. DataBase Schema



User follows user

This relations defines how a user can follow multiple users and can be followed by various users. Both attributes of the table are foreign keys of username from users table

User publishes voices

A voice is published by one and only one user, identified by *author* attribute, which is foreign key of username in users.

Voices belong to a district

Each voice belongs to a district, which is foreign key of districts. Since it is the user who decides publishing in some district, we have modeled this functionality in the controllers (if a user publishes in a nonexistent district the controller will add the new district).

User follows district

When a user publishes in a district, we also add a tuple in *userFollowsDistrict* to make the user follow the district.

5. Page navigation structure and Styling (CSS)

We have designed the structure of the page based on its **accessibility**. Our main objectives were, **reducing the number of clicks** to access the content, showing all the desired content in a visual and friendly way, and **avoid**ing **unnecessary complexity**.

Because most existing pages force you to register or get in their webs in an unintuitive way to show its public content, we considered this aspect could discourage potential users to register. For this reason, we decided to show public content on main page. Registering provides the full functionalities.

Another aspect that takes into account user comfort on the web is the feature ***turn of the lights*** to change the general use of colors. There are two options, a lighter compositions of composition of colors, and a darker one. This is not only so users have a choice about style, but also because if they felt the colors are too bright, they could choose an option not so visually exhausting.
Many CSS features are implemented using a :root global variables declaration, many classes depend on this, then when it changes they update its values

```
/* GLOBAL VARS   */
:root {
    --main-bg-color: #e4f4f0;
    --nav-bg-color: #90ac99;
    --nav-active-color: #819888;
    --titles-color: #633833;
    --titles-size: 48px;
    --block-color: none;
    --error-color: red;
    --font-color: grey;
    --voice-bg-color: white;
}
```

The **web is responsive**, meaning it will adapt the organization of html elements according to the screen size of the **device**. So it is not only apt for desktop, but also for tablet and phone. Even though the design changes, options (such as login, register, timeline, etc.).

6. Security

This way we store passwords in the database is secure against dictionary attacks. The process a password follows when it is sent from client is the following: First we combine the plaintext password with a general salt of the web plus a pepper (which is the username, which is unique). A SHA-256 hash is generated from this combination. This means, if a dictionary attack is performed, common passwords will not match the cracking because of the salt, and the fact that we use usernames as pepper it is not known to external agents.

Note: During the test phase, we tried to have a common password for all users by changing it directly in the DB, thus, this didn't work since the password was hashed with salt and pepper !

Further notes

Test it	User	Password
*You can **login as admin** with the following credentials:*	*juanDoc*	*1234QWERasdf*
*To **log in as common user**: agapito64 1234QWERasdf*	*agapito64*	*1234QWERasdf*
*You can also **register** to test the web.*		