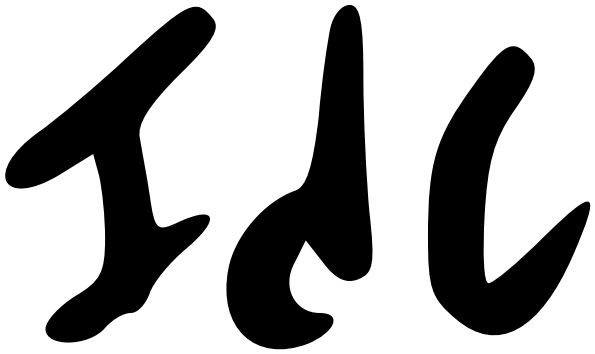


I DON’T CARE

Julián Sánchez

DESCRIPCIÓN BREVE

Trabajo de fin de grado

Fecha

10/04/2020

Contenido

Introducción

I don’t care (**IDC**) es una aplicación web que permite compartir una gran variedad de contenido, como imágenes, texto y vídeos en orden cronológico de forma dinámica.

La aplicación dispondrá de un sistema de comentarios tipo foro, por lo que se podrá discutir y debatir sobre cualquier publicación.

Todo el contenido subido a la plataforma será gestionable por el autor de dicho contenido, por lo que siempre tendrá control sobre sus propios recursos.

I don’t care se inspira en las redes sociales ‘Vero’, ‘Reddit’ y ‘9gag’ que también cuentan con un sistema similar.

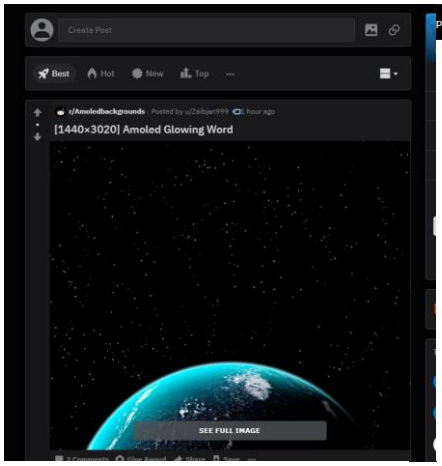


Figura I. Reddit

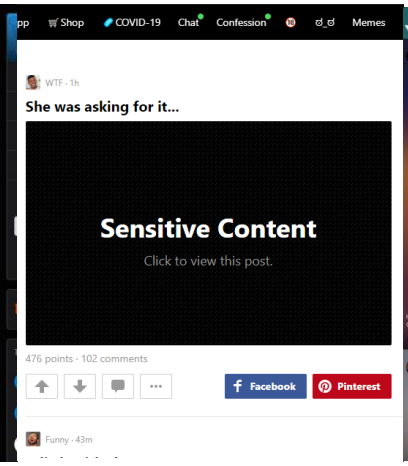


Figura II. 9GAG

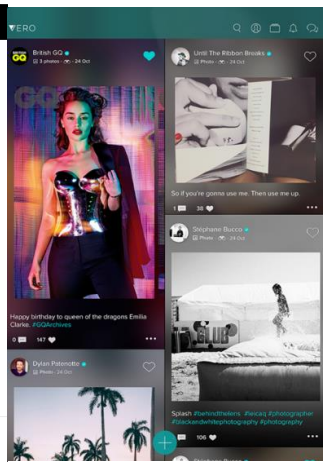


Figura III. Vero

I don’t care **se diferencia** en que no sólo pretende ser un sistema gestor de contenido, sino que pretende poner a disposición de acceso público parte de su infraestructura creando así la posibilidad de que se puedan gestar nuevas aplicaciones basadas en esta y en sus datos.

Para lograr esta meta se ha de disponer de un sistema escalable para poder afrontar futuras ampliaciones por lo que **se crearán dos subsistemas**:

- **Amaia (oAuth & API)**: Será la pasarela de autenticación, se encargará de gestionar los usuarios y aplicaciones conectadas, además será la base que proporcionará una ventana para acceder a los datos de la web permitiendo registrar tus propias aplicaciones.

Los usuarios de Amaia dispondrán de roles y permisos por lo que siempre se podrá tener un control de los recursos que se le serán concedidos.

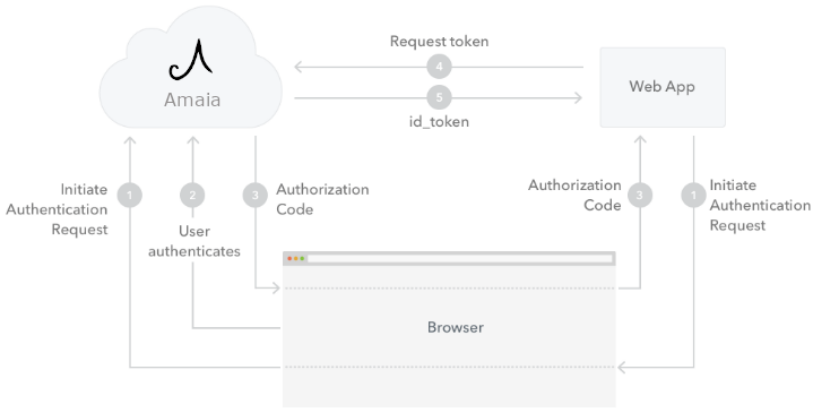


Figura IV. oAuth Workflow

Esto nos lleva a tener que crear algún tipo de aplicación para poder gestionar su configuración.

- **Whistler:** Será el encargado de dar una interfaz de administración que permita controlar tanto Amaia como I don't care.

Objetivos

Con este proyecto se pretende crear un ecosistema de aplicaciones ampliables y de libre acceso para fomentar el libertinaje de la información.

Objetivos del programa

Ya que se cuentan con tres distintos sistemas se separarán los objetivos por cada aplicación

I don't care

- Creación de contenido, pudiendo crear uno o varios agrupados en una misma sección.
- Compatibilidad de contenido con YouTube, imágenes de cualquier formato, vídeos mp4 y texto.
- Un sistema de comentarios por publicación.

- Un sistema de perfiles de usuario que almacenen las publicaciones creadas por cada uno.
- Un buscador que se encargue de filtrar contenidos y usuarios
- Un sistema de actualización de perfil que permita personalizar tu usuario.
- Un sistema de edición que permita corregir o añadir contenido a una publicación.

Amaia

- Una pasarela de inicio de sesión.
- Una pasarela de registro.
- Un sistema para registrar aplicaciones.
- Un sistema de roles y permisos.
- Un sistema que permita gestionar las aplicaciones que como usuario has autorizado que tengan acceso a tu información.

Whistler

- Un sistema para gestionar Amaia, tanto usuario como roles, permisos y configuración.
- Un sistema para gestionar i don't care, tanto contenidos, comentarios y configuración.

Objetivos personales

Con este proyecto pretendo desarrollar habilidades para crear y mantener un ecosistema de aplicaciones escalables y dinámicas. Qué sistemas implementar, cómo y saber tomar decisiones de escalabilidad y diseño de interfaz.

Planificación

Para el desarrollo del proyecto se pretende usar GitHub como aplicación de control de versiones, por lo que todos los cambios seguirán una planificación.

Como metodologías ágiles se usará Kanban ya que viene integrado con GitHub y se puede automatizar con el versionado de la aplicación para tener un control claro de qué se ha de hacer, corregir o está terminado.

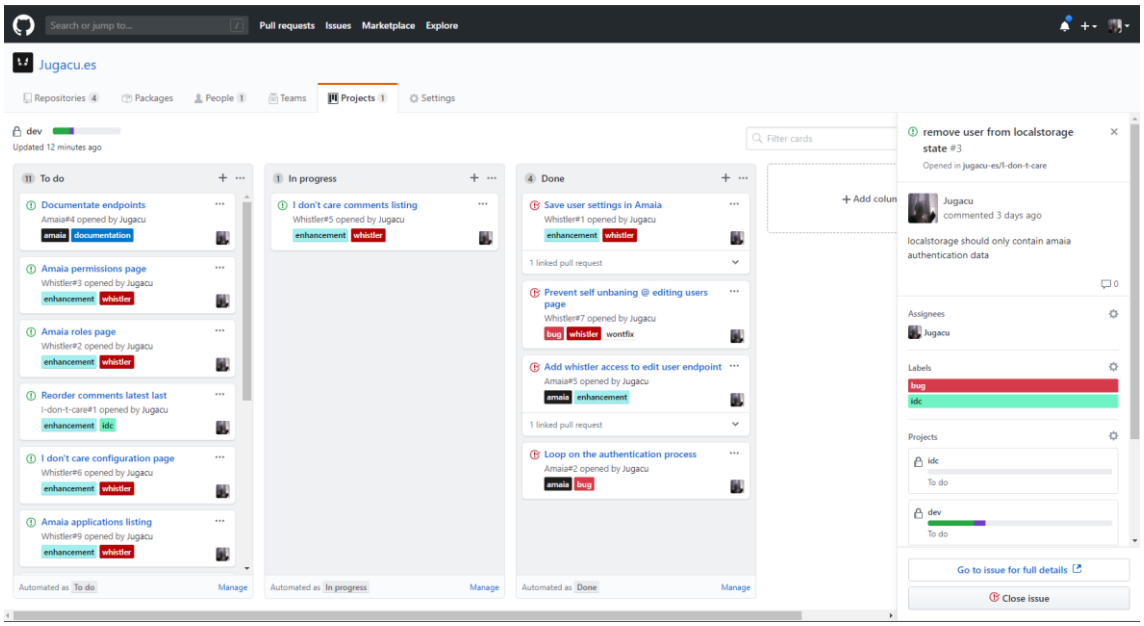


Figura V. Kanban del proyecto

Además de Kanban también se integrará un método de Scrum dividido en varios Sprints lo cual obligará a tener unas fechas límites de desarrollo y prototipos funcionales. La duración de dichos Sprint será de 1 semana de lunes a viernes durante 5 horas al día cada uno, teniendo un total de 5 Sprints.

- Sprint 1: En este sprint se pretende planificar toda la estructura de la base de datos y de las distintas aplicaciones (i don't care, Amaia, Whistler), crear el modelo relacional de la base de datos, investigar las posibles soluciones de escalabilidad disponibles y planificar su implementación.

- Sprint 2: El desarrollo de la infraestructura de Amaia (usuarios, permisos, pasarela y aplicaciones), esto incluye la base de datos planificada anteriormente y los sistemas seleccionados anteriormente.

- Sprint 3: Empezar el desarrollo de i don't care e implementar el proceso de autenticación que proporciona Amaia y desarrollo de la interfaz.

- Sprint 4: Continuar con el desarrollo de i don't care, crear el editor de contenidos, el sistema de perfiles, buscador y comentarios.

- Sprint 5: La creación de Whistler y su integración en el ecosistema.

Al final de cada uno de estos, se deberán crear los Test respectivos para verificar el correcto funcionamiento (esto se verá en más profundidad en el apartado de 'Pruebas').

Sprints

Sprint 1 (27 de abril al 1 de mayo)

Para el desarrollo de Amaia se han tenido en cuenta los siguientes sistemas, ambos de código abierto:

- Symfony: Ofrece un sistema de MVC ya predeterminado que permite crear tu propio Framework personalizado a partir de una base dada.

- Laravel: Basado en Symfony ampliando aún más su sistema MVC dando librerías de acceso a base de datos de manera más cómoda sin comprometer su escalabilidad.

Se ha seleccionado Laravel ya que complementa lo que nos da Symfony además de que dispone de un paquete llamado 'Laravel Passport' que ayudará con la implementación oAuth.

Para el desarrollo de i don't care y Whistler se han contemplado como Frameworks: React, VueJS y Angular.

Aquí se ha elegido Angular ya que dispone de muchos más paquetes que VueJS, es más sencillo de implementar que React e implementa Typescript por defecto.

Modelo relacional

Al plantear el siguiente esquema relacional se han tenido en cuenta los siguientes detalles:

- Un usuario puede tener uno o varios roles, que dependiendo de qué permisos tenga cada rol será capaz de acceder a un recurso de la aplicación.
- Un usuario es capaz de tener permisos indistintamente de tener un rol u otro.
- Un usuario es capaz de pedir un reseteo de contraseña al correo, por lo que se han de guardar los tokens y la fecha de creación para poder calcular su fecha de expiración.
- Un usuario es capaz de crear sus propias aplicaciones oAuth en el sistema que pedirán autorización a otro usuario para acceder a sus datos o sus permisos.
- Los clientes oAuth constarán de un nombre de aplicación para ser humanamente identificable, una id y secretos generados por el servidor y la URL de la aplicación.
- El usuario dispondrá de tokens que serán los que guardarán los permisos que el usuario ha concedido a X aplicación, estos tokens tendrán una fecha de expiración, de creación y de actualización. Los tokens, al ser creados generan otros tokens de refresco que permiten aumentar la fecha de expiración.
- El proceso oAuth requiere que se generen unos códigos temporales para presentar al usuario su autorización o denegación que estén vinculados con la aplicación.
- Cada usuario tendrá la capacidad de crear posts, cuyos contenidos estarán compuestos por descripción y tipo (IMG, URL, YT, PH).
- Los usuarios también podrán comentar en los posts de otros. Así, estos almacenarán el texto, el usuario que ha comentado y su post.


configs
id bigint(20)
optionName varchar(191)
optionValue varchar(191)
created_at timestamp
updated_at timestamp
comments
id bigint(20)
post_id bigint(20)
user_id bigint(20)
comment text
created_at timestamp
updated_at timestamp
oauth_clients
id char(36)
user_id bigint(20)
name varchar(191)
secret varchar(100)
redirect text
personal_access_client tinyint(1)
password_client tinyint(1)
revoked tinyint(1)
created_at timestamp
updated_at timestamp
oauth_refresh_tokens
id varchar(100)
access_token_id varchar(100)
revoked tinyint(1)
expires_at datetime
postcontents
id bigint(20)
post_id bigint(20)
desc text
type postcontents_type_enum
outstanding varchar(191)
created_at timestamp
updated_at timestamp
oauth_access_tokens
id varchar(100)
user_id bigint(20)
client_id char(36)
name varchar(191)
scopes text
revoked tinyint(1)
created_at timestamp
updated_at timestamp
expires_at datetime
oauth_auth_codes
id varchar(100)
user_id bigint(20)
client_id char(36)
scopes text
revoked tinyint(1)
expires_at datetime
posts
id bigint(20)
user_id bigint(20)
title varchar(191)
nsfw tinyint(1)
created_at timestamp
updated_at timestamp
roles
id bigint(20)
name varchar(191)
desc varchar(191)
color varchar(191)
badge varchar(191)
guard_name varchar(191)
created_at timestamp
updated_at timestamp
model_has_roles
role_id bigint(20)
model_type varchar(191)
model_id bigint(20)
users
id bigint(20)
name varchar(191)
email varchar(191)
email_verified_at timestamp
password varchar(191)
avatar varchar(191)
desc varchar(191)
nsfw tinyint(1)
remember_token varchar(100)
created_at timestamp
updated_at timestamp
model_has_permissions
permission_id bigint(20)
model_type varchar(191)
model_id bigint(20)
permissions
id bigint(20)
name varchar(191)
guard_name varchar(191)
created_at timestamp
updated_at timestamp
role_has_permissions
permission_id bigint(20)
role_id bigint(20)
password_resets
email varchar(191)
token varchar(191)
created_at timestamp

Sprint 2 (4 de mayo al 8 de mayo)

En este sprint se pretende implementar Laravel y hacer el sistema básico de autenticación, permisos y roles de usuarios.

Laravel usa Composer, así que para instalarlo se han tenido que cumplir los siguientes requisitos locales:

- PHP 5.3.2+
- PHP CURL

Con Composer ya listo Laravel se instalará con los dos siguientes comandos:

- composer global require laravel/installer
- laravel new amaia

Nota: Actualmente (04/05/2020) la última versión de Laravel es la 7.x

Antes de continuar con la creación de la base de datos, las migraciones y la configuración se instalarán los siguientes paquetes:

- Para el sistema de la pasarela (oAuth2) se planteó en el Sprint 1 usar laravel passport que se instalará siguiendo los pasos de la documentación.
- Para el sistema de roles y permisos se usará el paquete laravel-permissions, que es de código abierto y nos ayudará a la hora de establecer la lógica de control de los recursos.

Con Laravel y los permisos ya listos, podemos empezar a crear las migraciones, que no son más que una forma de indicarle al Framework como debería ser la base de datos y las tablas que contiene.

Una migración se crea usando el comando:

- php artisan make:migration

Y en ella deberá estar definido el modelo relacional que se creó en el Sprint 1

```php
/**
 * Run the migrations.
 *
 * @return void
 */
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->bigIncrements( column: 'id');
        $table->string( column: 'name')->unique();
        $table->string( column: 'email')->unique();
        $table->timestamp( column: 'email_verified_at')->nullable();
        $table->string( column: 'password');
        $table->string( column: 'avatar')->default( value: 'https://i.imgur.com/oTOMHFn.gif');
        $table->string( column: 'desc')->default( value: 'silence is golden');
        $table->boolean( column: 'nsfw')->default( value: 0);
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Figura VI. Migración de la tabla usuarios

Una vez fueron creadas las tablas, se podrá empezar a poblar con datos iniciales como los distintos roles que habrá en la aplicación y sus permisos. Para ello, Laravel dispone de un sistema llamado ‘Seeds’ o ‘Seeder’.

```php
// Idc post permissions
Permission::create(['name' => 'create posts']);
Permission::create(['name' => 'edit own posts']);
Permission::create(['name' => 'delete own posts']);

Permission::create(['name' => 'edit any post']);
Permission::create(['name' => 'delete any post']);
```

Figura VII. Seeds de los permisos

```php
Role::create(['name' => 'user'])->givePermissionTo(
    [
        'create posts',
        'edit own posts',
        'delete own posts',
        'create comments',
        'edit own comments',
        'delete own comments',
        'register applications',
        'delete own applications'
    ]
);
```

Figura VIII. Seed del rol usuario

Para el diseño de Amaia se aprovechará que Laravel utiliza NodeJS para instalar el paquete Tailwind que es un sistema de CSS parecido a Bootstrap, pero con soporte para colores.

```
<h1 class="text-black font-semibold tracking-tight text-xl mb-5">{{ __('Sign In') }}</h1>
```

Figura IX. Ejemplo de CSS con Tailwind

Para el login y registro se ha realizado un diseño sencillo de usuario y contraseña.

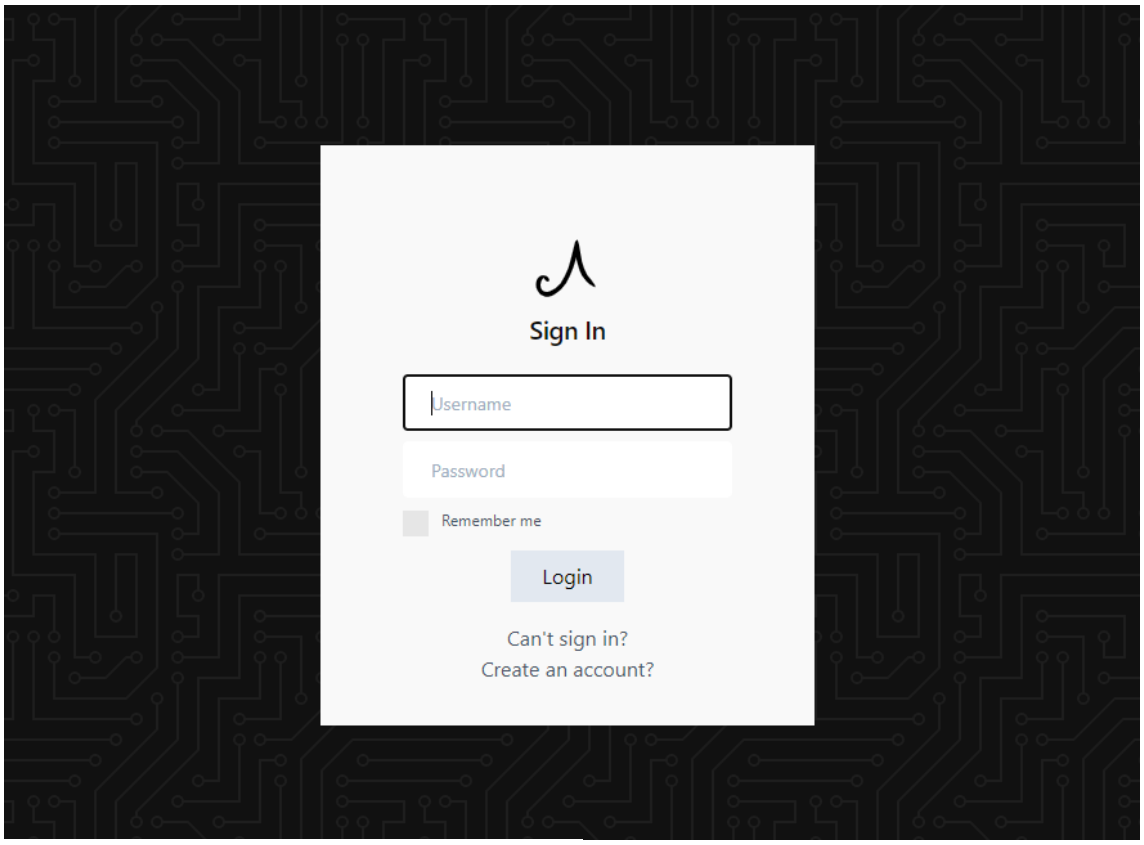


Figura X. Página de inicio de sesión de Amaia

El Dashboard necesitará varias páginas para funcionar (todas estas tendrán más sentido y se complementarán en el Sprint 3)

- Una página con la información de la cuenta.
- Una página que liste las aplicaciones creadas por cada usuario.
- Una página para crear sus propias aplicaciones.
- Una página que muestre a las aplicaciones que has autorizado el acceso a tu cuenta.

El diseño también ha sido minimalista y sólo se muestra la información importante para el usuario.

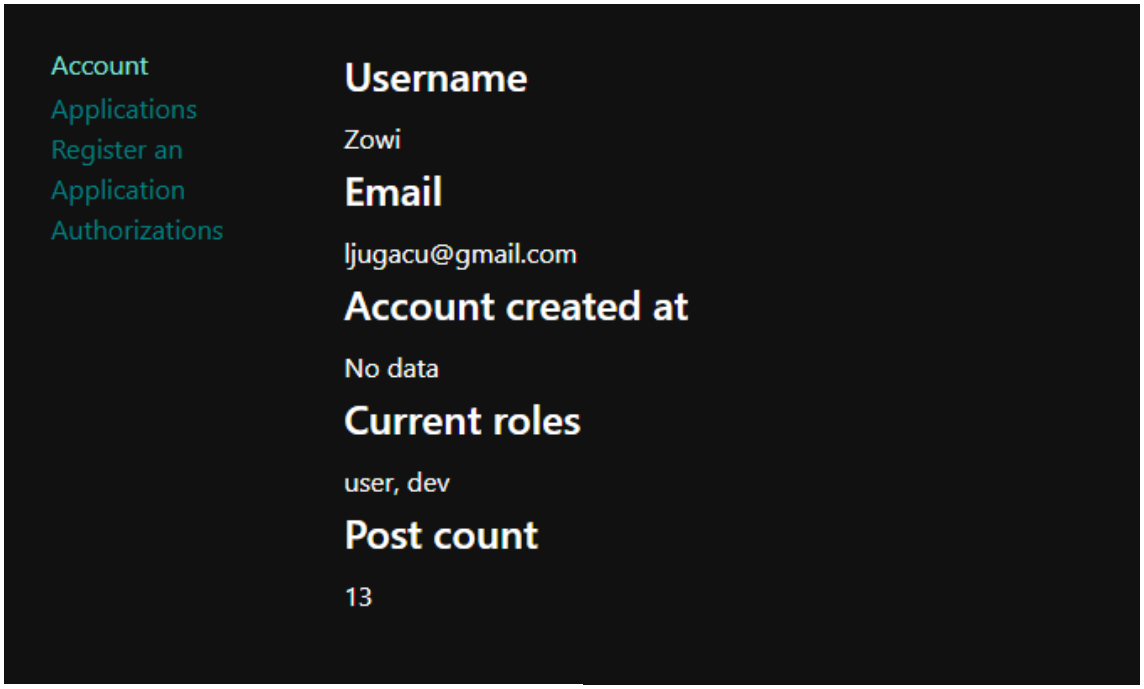


Figura XI. Dashboard de Amaia

Resultado

Sprint 3 (11 de mayo al 17 de mayo)

En este sprint se pretende implementar Angular y el acceso a la pasarela de Amaia (el tiempo de desarrollo ha sido ampliado, leer 'Problemas')

Para empezar a desarrollar i don't care primero se ha de registrar en Amaia la aplicación para que pueda acceder a sus endpoints, por lo que en el Dashboard (como se mencionó en el Sprint 2) se ha diseñado este pequeño formulario.

Figura XII. Dashboard de Amaia, registro de aplicaciones

Y un listado con las aplicaciones que el usuario ha creado.

Figura XIII. Dashboard de Amaia, listado de aplicaciones

En el código de la aplicación se ha creado un archivo JSON que contendrá toda esta información para facilitar el desarrollo y tener todo centralizado. También **se han separado las interfaces en una librería** que podrá ser usada en el desarrollo de Whistler.

```
{
  "AMAIA_API_URL": "http://127.0.0.1:800/api/idc",
  "AMAIA_BASE_URL": "http://127.0.0.1:800",
  "SECRET_HANDLER": "http://localhost:8080/secret",
  "SECRET_HANDLER_REFRESH": "http://localhost:8080/secret/refresh.php",
  "AMAIA_CLIENT_ID": "39a112a0-96c4-11ea-97ec-ff27dc910aba",
  "AMAIA_CALLBACK": "http://localhost:4200",
  "AMAIA_RESPONSE_TYPE": "code",
  "AMAIA_SCOPE": "manage-account manage-posts manage-comments",
  "IMGUR_AUTH": "8cbb9f54d9c6301"
}
```

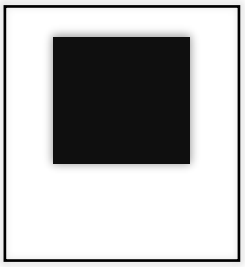
Figura XIV. Archivo JSON de configuración de i don't care

Manual de estilos

I don't care se basa bastante en el diseño de Vero por lo que contendrá bastante desenfoque de fondo y colores muy oscuros.

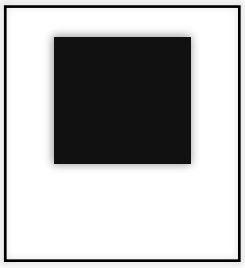
La fuente a usar será "**Roboto Regular**" y tendrá un interlineado de 1.5em

La web deberá respetar los siguientes esquemas de colores. Aunque su similitud por separado no se pueda apreciar es importante respetarlos ya que en gran medida será completamente visible la diferencia de colores.



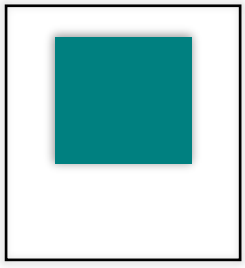
Color principal

#1d1d1d



Color secundario

#1b1b1b



Color de acentuación

#008080

RGB (0, 128, 128)

Este se usará en los hipervínculos si los hay o en cualquier elemento en el cual se tenga que llamar la atención al usuario.

Las letras serán tendrán un color blanco (#FFFFFF) y a aquellas en las que su valor sea de menor importancia (subtítulos, etc.) se les aplicará una opacidad de 0.8.

Los desenfoques están creados con una imagen principal la cual será visible de forma normal y el Blur como tal que estará detrás en el fondo.

Figura XV. Prototipo de desenfoque de i don’t care en usuarios

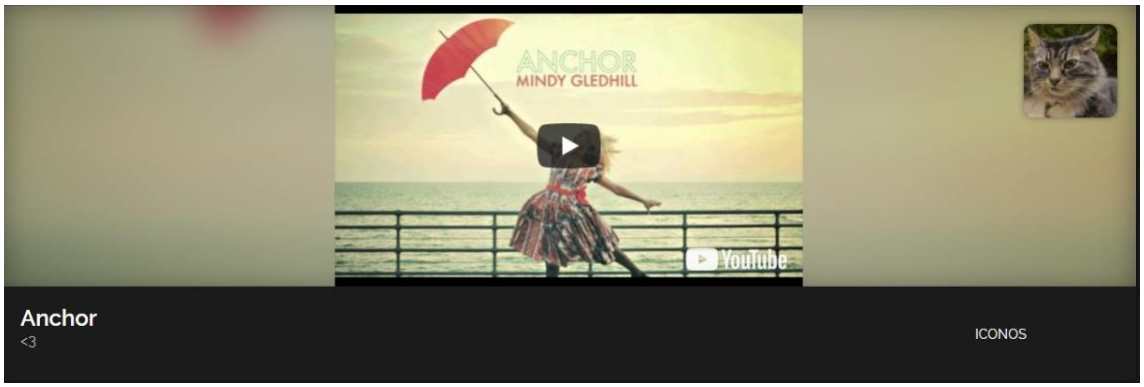


Figura XVI. Prototipo de contenido de i don’t care

La interfaz de i don’t care constará de un menú lateral de navegación a la derecha y las páginas que se vayan visitando a la izquierda.

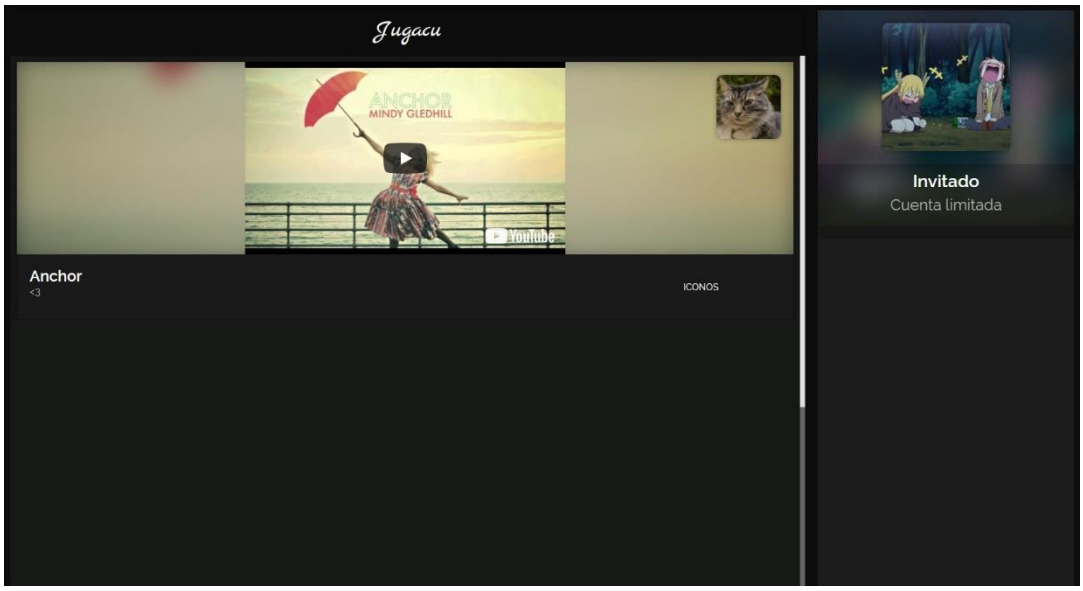


Figura XVII. Prototipo de la página de i don't care

Con los datos de Amaia el botón de inicio de sesión deberá redirigir a los usuarios para pedirles acceso a su cuenta y verificar que todo se ha realizado correctamente.

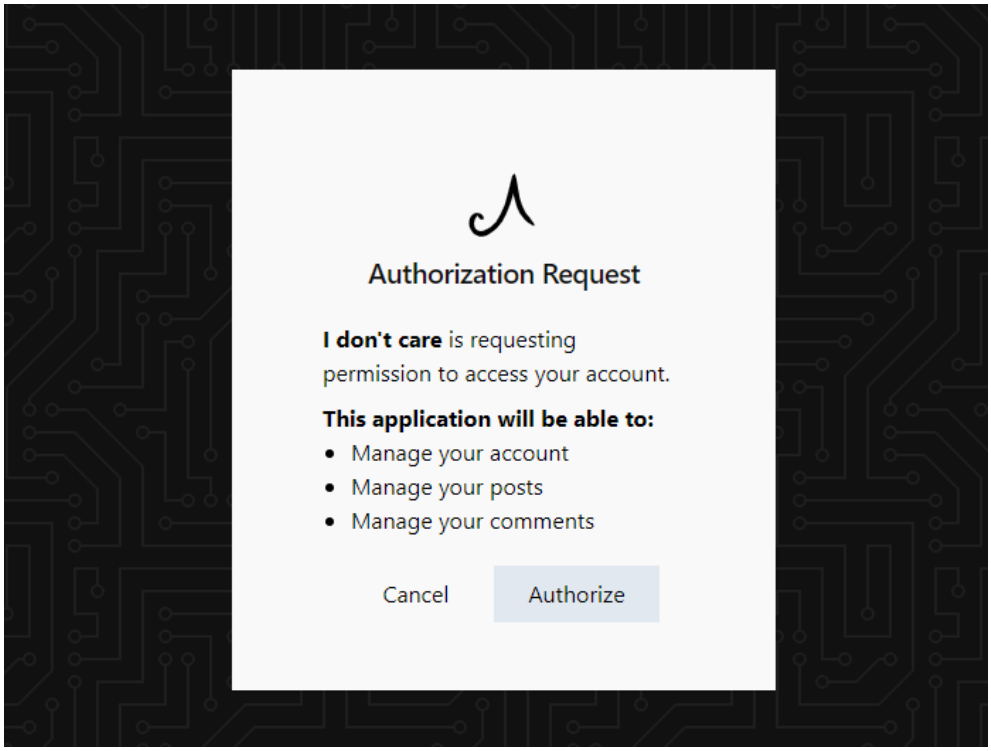


Figura XVIII. Página de autorización de Amaia

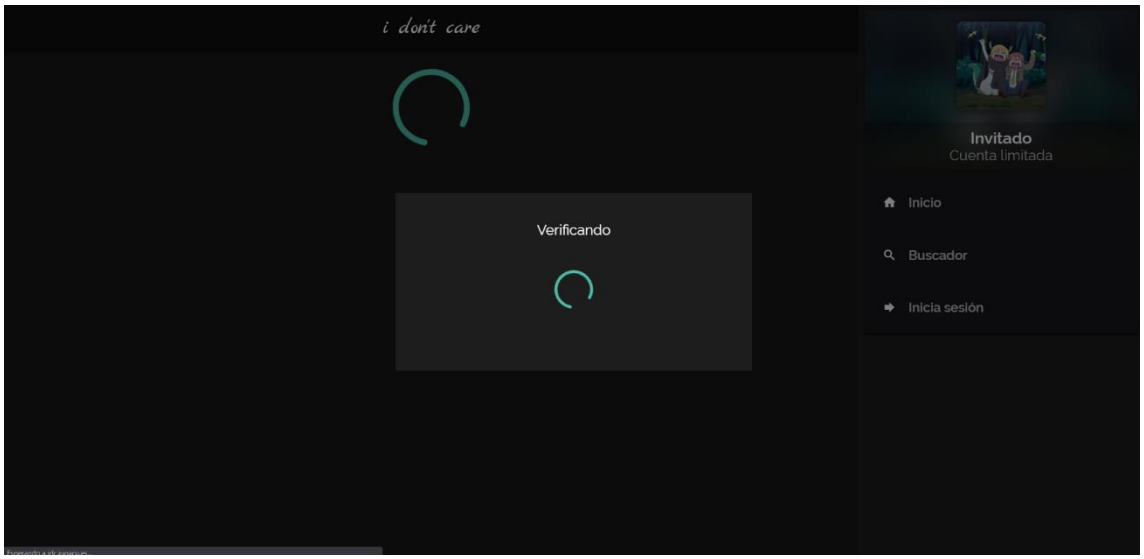


Figura XVIII. Proceso de verificación de i don't care

También se ha eliminado la tabla 'config' de la base de datos ya que no se utiliza.

Problemas

Este Sprint ha durado más de lo planeado, se tuvo que añadir dos días más (16 y 17 de mayo) para cumplir los objetivos asignados en el Sprint por causa de los siguientes problemas:

- Loop infinito de autenticación al permitir acceso a I don't care que costó encontrar y arreglar.
- Customizar Passport para que cumpla las funcionalidades especiales que requiere Amaia, como que sólo sea accesible desde el Dashboard ya que la documentación no está completa y se tuvo que hacer ingeniería inversa.
- Se intentó separar los distintos modelos de i don't care y Amaia en distintas bases de datos, cosa que Laravel permite, pero sin el potencial de crear relaciones entre ellos. Por lo que al final no se llegó a hacer.

Resultado

En este Sprint se ha logrado crear una interfaz básica en i don't care, menús en Amaia para gestionar las aplicaciones e implementar el inicio de sesión.

Requisitos de funcionamiento

Amaia utiliza Laravel como Framework para agilizar el trabajo, por lo que el servidor ha de tener instalados los siguientes requisitos para funcionar:

- Un servidor Nginx, Apache, Caddy o similar.
    - PHP >= 7.2.0
    - BCMath PHP Extension
    - Ctype PHP Extension
    - JSON PHP Extension
    - Mbstring PHP Extension
    - Extensión OpenSSL de PHP
    - Extensión PDO de PHP
    - Extensión Tokenizer de PHP
    - Extensión XML de PHP
    - Certificado SSL ya que las aplicaciones I don’t care y Whistler sólo aceptan peticiones enviadas a protocolo HTTPS
- La base de datos puede ser externa o estar incluida en el mismo servidor, ha de ser SQL como MariaDB, MYSQL o similar.