

BOTBLOQ: Ecosistema integral para el diseño, fabricación y programación de robots DIY

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI)

EXPEDIENTE: IDI-20150289

Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

ACRÓNIMO DEL PROYECTO: BOTBLOQ



Centro para el
Desarrollo
Tecnológico
Industrial



UNIÓN EUROPEA
Fondo Europeo de
Desarrollo Regional (FEDER)
Una manera de hacer Europa

ENTREGABLE E.2.3.1 INFORME DESCRIPTIVO DE LA APLICACIÓN

RESUMEN DEL DOCUMENTO

La aplicación desarrollada permite la programación los componentes del kit de robótica a través de bloques. En este documento se especifica técnicamente cómo se ha construido la aplicación para que cumpla los requisitos de accesibilidad.

February 3, 2016

Contents

1 Descripción de la necesidad	3
2 Visión del producto	4
3 Fases de desarrollo de la plataforma	5
3.1 Fase 1 - bitbloq v1	6
3.1.1 Objetivos	6
3.1.2 Diagramas de flujo de alto nivel	9
3.1.3 Wireframes	22
3.1.4 Diseños	23
3.1.5 Chrome App (bitbloq Serial Uploader)	26
3.1.6 Tecnologías usadas	34
3.1.7 Convenios y estilos	35
3.1.8 Features y resumen del backlog de la Fase I	36
3.1.9 Improvements tras salida a producción	38
3.1.10 Redefinición de producto tras la primera fase de Bitbloq v1 .	39
3.2 Fase 2 - Bitbloq v2	45
3.2.1 Necesidades de la nueva plataforma	45
3.2.2 Objetivos	48
3.2.3 Soluciones	49
3.2.4 Diagramas de flujo de alto nivel	50
3.2.5 Wireframes - Anexo	53
3.2.6 Nueva librería de bloques “Bloqs”	53
3.2.7 Bloques que modelan elementos de lógica y control del flujo del programa.	54
3.2.8 Librerías Arduino - BitbloqLibs	63
3.2.9 Nueva aplicación Web2Board para compilación	64
3.2.10 Plan de migración de usuarios de BitbloqV1 a BitbloqV2.0 .	66
3.2.11 v2.0 en Producción	67
3.2.12 Infraestructura para test unitarios y end-to-end.	67
3.2.13 IDE, entorno de desarrollo integrado	84
3.3 Estructura y base de datos.	87

3.3.1 Documental	87
3.3.2 Autenticación y registro	99
3.4 Integración con ecosistema DIWO	104
4 Futuros desarrollos de la plataforma	105
4.1 Profesores y alumnos	105
4.2 Detección del esquema hardware automáticamente	106
4.3 Programación inalámbrica	107
4.4 Widgets	107
4.5 Varias placas	107
4.6 Detección de errores	107
4.7 Robots	108
4.8 Integración en ROS	109
5 Anexos	110
5.1 Anexo I. Infraestructura de servicios.	110
5.2 Anexo II. CoreJS	148
5.3 Anexo III. Ejemplo de xml de un proyecto de Bitbloq 1	173
5.4 Anexo IV. Ejemplo de proyectos de bloques, código y de robots de Bitbloq 2	192
5.5 Anexo V. Wireframes bitbloq V1	213
5.6 Anexo VI. Wireframes bitbloq V2	215
5.7 Anexo VII. Diseños de la aplicación	215
5.8 Anexo VIII. Tareas realizadas por el equipo de desarrollo	244
5.9 Anexo IX. 100 proyectos de robótica con Bitbloq y Arduino	244

1 Descripción de la necesidad

Bitbloq es una aplicación que tiene como objetivo que los niños aprendan cómo funciona la tecnología y no sólo sepan utilizarla.

Esta plataforma de software se desarrolla dentro del marco de un ecosistema de robótica educativa. Este hecho supone un uso diferente de un entorno de programación visual de software, adaptado a la programación de placas micro-controladoras. Debido al trasfondo innovador DIY, la solución de software difiere de las tradicionales plataformas docentes de programación de videojuegos, aunque comparten los mismos principios y objetivos a la hora de adaptar la tecnología al conocimiento del usuario.

Como objetivo final, el entorno de programación sirve al usuario para unir programación y robótica sin apenas esfuerzo, de forma que hasta un niño pueda aprender a dar vida a una máquina.

Asimismo, uno de los requisitos principales para el entorno de programación es usar plataformas modernas que permitan ‘romper la barrera de entrada’ al uso de este tipo de tecnologías.

2 Visión del producto

Bitbloq es una plataforma que permite programar placas controladoras sin tener que instalar ningún entorno de desarrollo integrado (IDE) en el ordenador. Su primer objetivo es implantarse como la mejor herramienta educativa de programación por bloques para plataformas electrónicas.

El producto tendrá que tender a la monetización para maximizar el ROI. Para ello se prevé que los fabricantes puedan introducir sus propios bloques y además se puedan vender las piezas necesarias para fabricar el proyecto diseñado a través de la plataforma.

La aplicación se debe certificar en las últimas versiones de los navegadores cumpliendo las siguientes compatibilidades:

Navegador	Compatible
Chrome	100% Compatible
Chromium	Parcialmente
Firefox	Parcialmente
Resto de navegadores (Internet Explorer, Opera, ...)	No compatible

La aplicación se debe certificar en los siguientes sistemas operativos cumpliendo las siguientes compatibilidades:

Sistema Operativo	Compatible
Windows 7	100% Compatible
Ubuntu 14.04	100% Compatible
MacOS X	100% Compatible

3 Fases de desarrollo de la plataforma

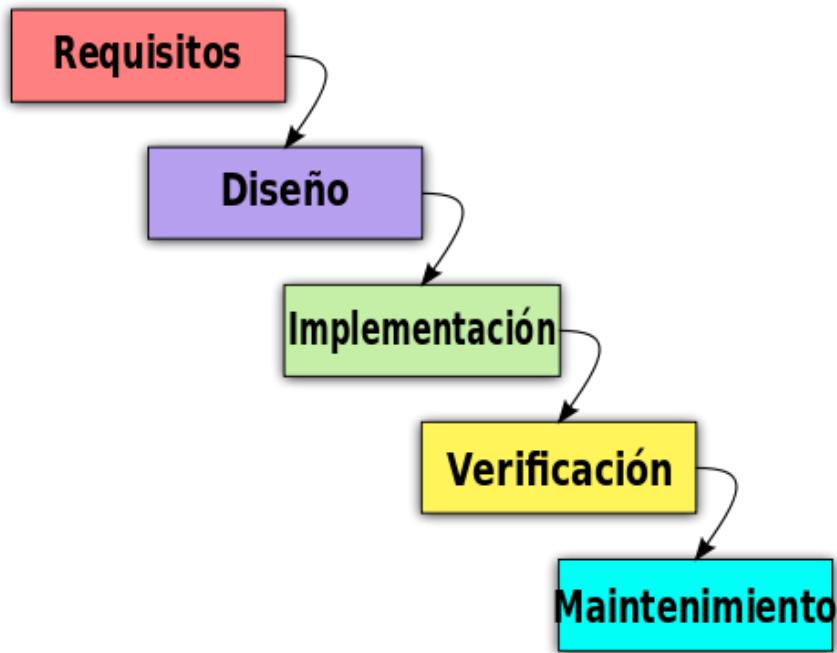
Para materializar el desarrollo de la plataforma se decidió establecer dos fases de desarrollo, definiendo en la primera fase un mínimo producto viable (MVP) en base a los requisitos mínimos establecidos en las especificaciones. La segunda fase de desarrollo está orientada a complementar y mejorar funcionalidades que surgen tras hacer pruebas en entornos con usuarios reales y seguir afinando en construir una plataforma con identidad propia centrada en el usuario.

Para comprender las fases de desarrollo dentro de un producto software, a modo resumen, se listan más abajo todas las etapas que intervienen en el proceso:

1. Definición de requisitos en base al modelo de negocio.
2. Diseño de la aplicación
 - (a) Diseño de Wireframes: El departamento de UX presenta una serie de esquemas que muestran las principales funcionalidades de la aplicación. En dicha presentación participan todos los integrantes del equipo para tratar de mejorar flujos y analizarlos desde todas las perspectivas. Dichos esquemas se dividen en función de las secciones de la aplicación para facilitar el análisis.
 - (b) Diseño de Mockups: Una vez resueltos los wireframes se realizan los correspondientes diseños de la interfaz. Estos se trasladan a los desarrolladores que se encargan de plasmarlo en el código.
 - (c) PoC & Start Coding: En función de los wireframes y los mockups de la aplicación, el equipo de desarrollo decide un stack tecnológico con el que cumplir todos los objetivos. Se analizan las necesidades y se presentan las pruebas de concepto correspondientes.
3. Implementación:

Se marcan objetivos a corto, medio y largo plazo para la creación, asignación y realización de tareas en base a su funcionalidad.

1. Fase de pruebas
2. Proceso de mantenimiento y corrección de errores



3.1 Fase 1 - bitbloq v1

El objetivo final de la fase 1 es crear una aplicación web que permita a los usuarios componer programas online usando bloques. Las placas controladoras que, como mínimo, la plataforma debe ser capaz de programar son: bq ZUM BT 328, Freaduino UNO y Arduino UNO original.

3.1.1 Objetivos

Los principales objetivos se declaran a continuación:

1. Habrá una landing o página de entrada, a modo de introducción a la plataforma, desde donde se podrá acceder al sistema de login y/o al modo de prueba

del espacio de trabajo de la plataforma. Se añadirá página estática con parte de los recursos y contenidos que ahora tenemos en el portal DIWO (diwo.bq.com). Se proporcionará en esta página información relevante para que los usuarios aprendan a manejar la herramienta correctamente.

2. Servicio de login/registro para usuarios. Desde el login acceden y se registran los usuarios. Existen tres tipos de usuarios:

- Usuarios Convencionales
- Administradores
- Invitado

El usuario no-administrador puede registrarse por Facebook y Google.

3. La aplicación web debe ser capaz de generar código arduino a partir de bloques. Se debe crear un repositorio de bloques para los componentes electrónicos y kit de robótica de bq.

4. Poder compilar el código generado y descargar en la placa controladora.

- Elegir tipo de placa: bq ZUM BT 238 o Freaduino UNO. El tipo de placa se guarda para cada usuario.
- Elegir puerto donde está conectada la placa

5. Cualquier usuario puede abrir o guardar sus proyectos tanto en local como en la nube. Se puede acceder sin logarse, pero no se podrán guardar los proyectos ni añadir descripción a los proyectos guardados

6. Un proyecto se puede salvar en cualquiera de los ámbitos:

- Privado
- Compartido con todo el mundo (El usuario puede abrir los proyectos de otros usuarios que sean públicos)

7. Cuando el usuario accede se abre el último proyecto en el que estaba trabajando.
8. Si se accede sin logarse se abre el último proyecto que había en “ese ordenador”
9. Los proyectos ya tienen asociadas etiquetas para facilitar su clasificación y búsqueda en la nube. También dispone de un autoetiquetado utilizando como labels los componentes/bloques utilizados. Existirán otro tipo de campos a través de los que se podrán filtrar proyectos:
 - Filtrado por usuario
 - Nombre de proyecto
 - Por descripción
10. Los proyectos públicos pueden tener varios ámbitos:
 - Nube privada (sólo ve sus contenidos el propio usuario; nadie puede eliminarlos salvo él)
 - Nube Pública (su contenido los puede ver cualquiera, incluso aquellos que no se han registrado)
11. Dar soporte multi-idioma y que el usuario pueda elegir el idioma. Cuando un usuario cambia el idioma, deben cambiar todos los literales (bloques, interfaz y diálogos). La opción de idioma se guarda para cada usuario.
12. Poder abrir un proyecto desde URL externa pasando por el login previamente(Una url que va a la bitbloq y carga el proyecto).
13. Bitbloq contará con un panel de seguimiento y KPIs de tráfico y conversión implementado sobre Google Analytics.

Otras funcionalidades quedan a modo de backlog para una segunda fase del proyecto:

1. Incluir el tipo de usuario Fabricante.

Los fabricantes podrán añadir su propio set de bloques.

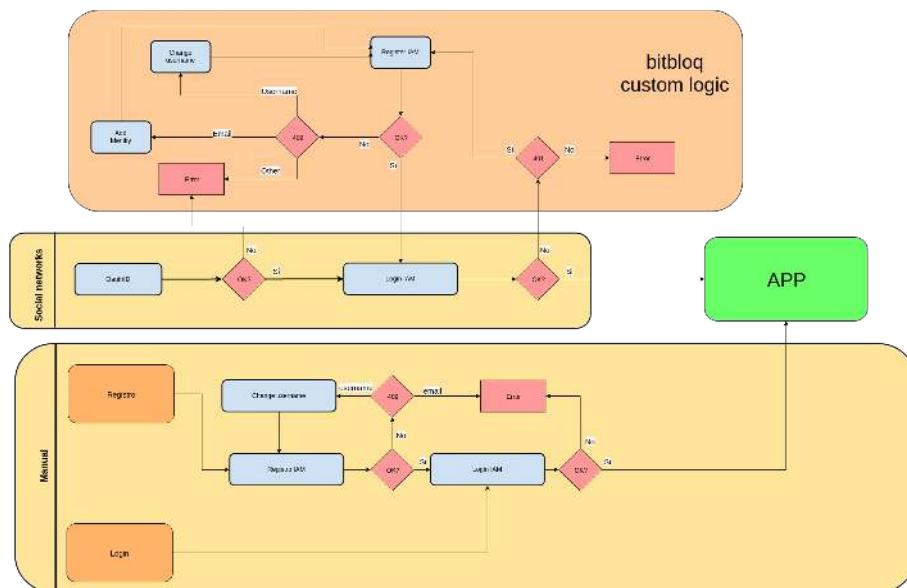
3. Monetización mediante la compra del despiece de un proyecto (pasaría el pedido a nuestra tienda Magento)
4. Crear plantillas, el usuario puede cambiar la vista de la aplicación eligiendo entre plantillas disponibles.
5. Servicio de mensajería interna: los usuarios pueden mandarse mensajes entre ellos (sin que sean públicos los datos personales)
6. Integración con Moddle para que desde portales de e-learning realizados con esta herramienta (herramienta de uso muy extendido en ámbitos académicos) se puedan incluir y valorar ejercicios por un profesor.
7. Proyecto colaborativos
8. Compartir en FB, twitter...

3.1.2 Diagramas de flujo de alto nivel

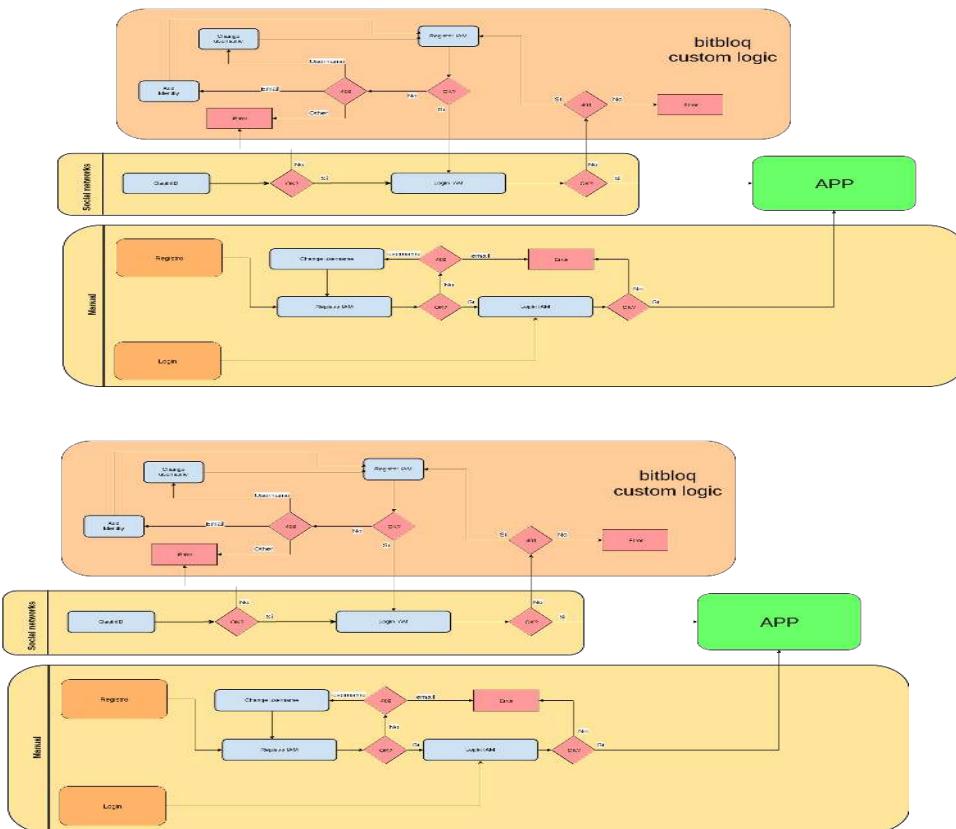
Para tener más claro el comportamiento que ha de tener ciertos caminos funcionales, que consideramos más complejos, hicimos representaciones gráficas paso a paso del proceso completo, también llamado flujo de trabajo.

Autenticación y registro al sistema El proceso de autenticación y registro de un usuario es bastante crítico ya que hay que tener en cuenta tanto el tipo de acceso como las partes de la base de datos que participan en el acceso. Se puede distinguir dos tipos de acceso:

- **Redes Sociales:** En este tipo de acceso al sistema aparecen dos tipos de entidades diferentes: iam y oauth. La primera corresponde a los datos del usuarios como son su correo electrónico o sus datos personales, mientras que la entidad oauth es la que almacena la relación con la entidad de la red social con la que el usuario desea acceder.



- **Manual:** Para acceder de forma manual, es decir, cuando un usuario escribe directamente su correo y su contraseña, no es necesario mantener ninguna relación con una entidad externa. Por esta razón, al acceder de forma manual al sistema no tenemos que tener en cuenta oauth. En este caso tan solo debemos ver si un usuario está registrado en iam y si no pedirle que registre sus datos.

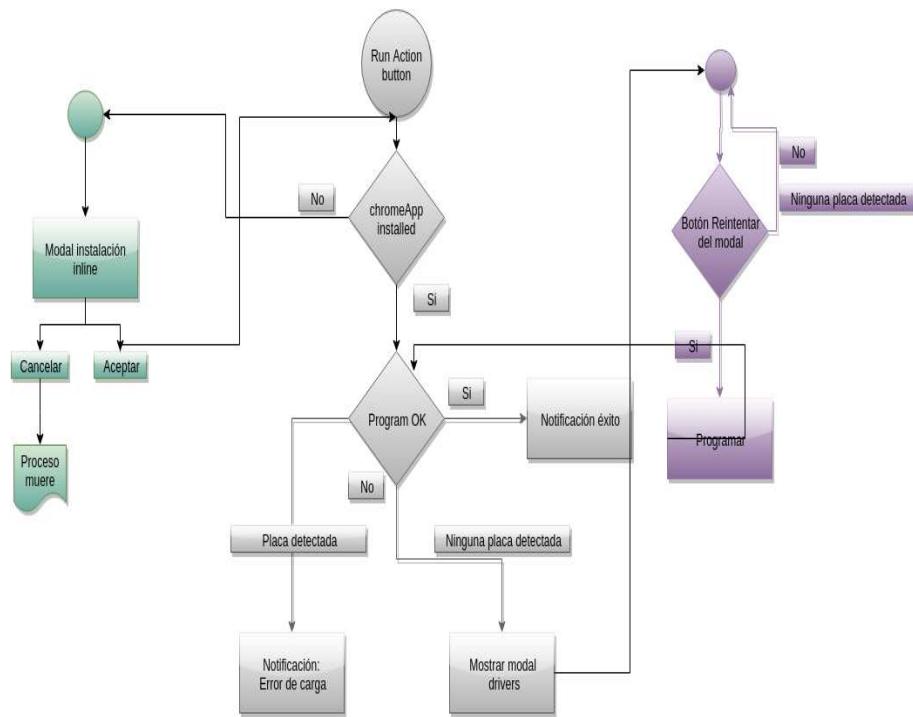


Acción programar Es una de las funcionalidades más importantes de la aplicación. Permite al usuario enviar su programa a la placa siguiendo el flujo definido en el diagrama.

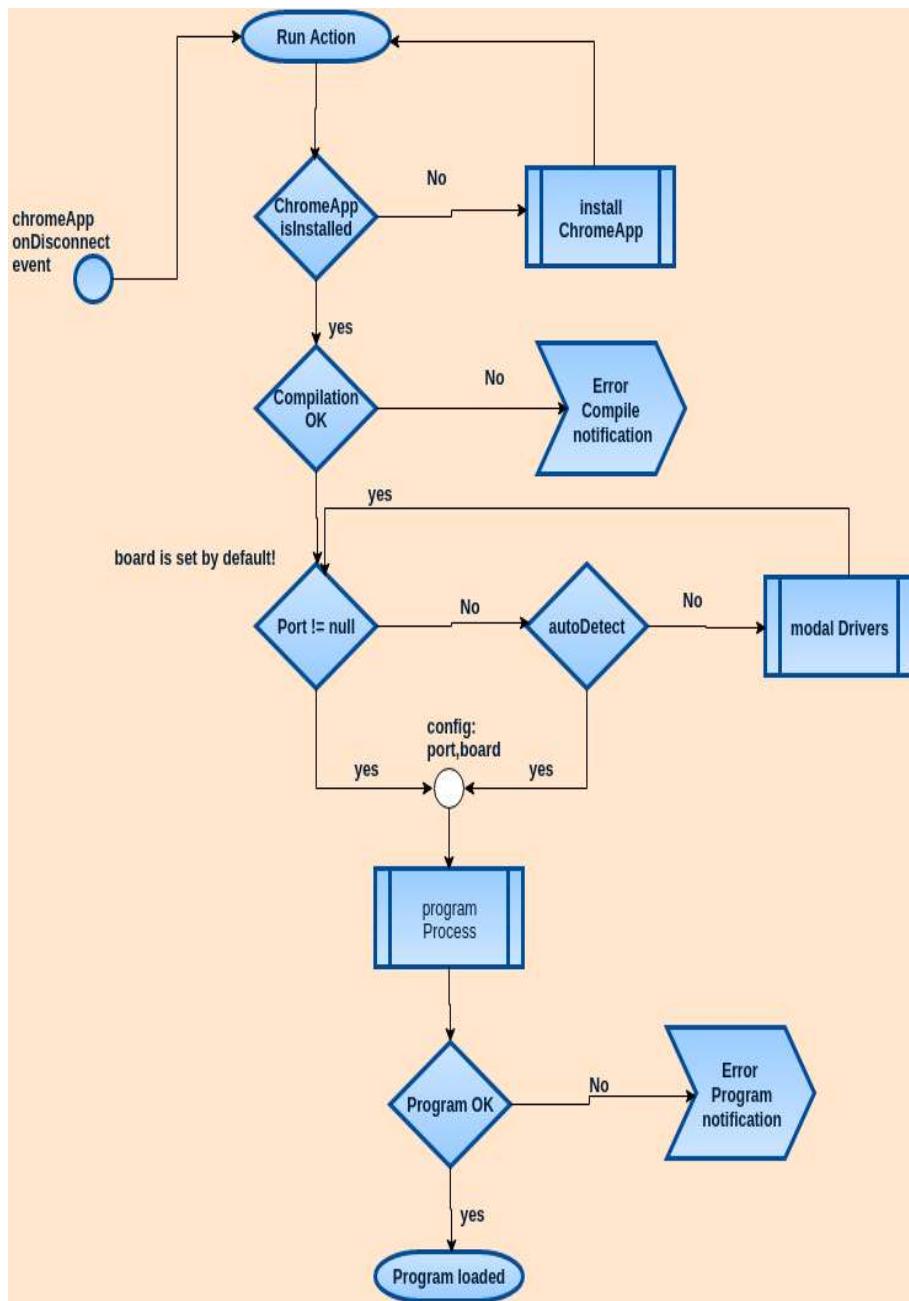


En su primera versión, el flujo de programación era el siguiente:

- Se comprueba si la chrome app está instalada.
- La chrome app comprueba la existencia de alguna placa conectada al equipo.
- La chrome app carga el programa en la placa.
- En caso de haber algún fallo se notifica al usuario. En función del punto en que se encuentre dicho fallo recibirá una notificación u otra.
- Se identifica el punto en que ocurrió el error, informando de si ocurrió durante la fase de compilación del código o de carga a la placa.



La segunda versión modificamos el flujo para incluir la autodetección de la placa, así como la selección manual de la placa y el puerto. De esta forma simplificamos el trabajo al usuario al no ser necesaria su intervención para establecer la placa y el puerto.

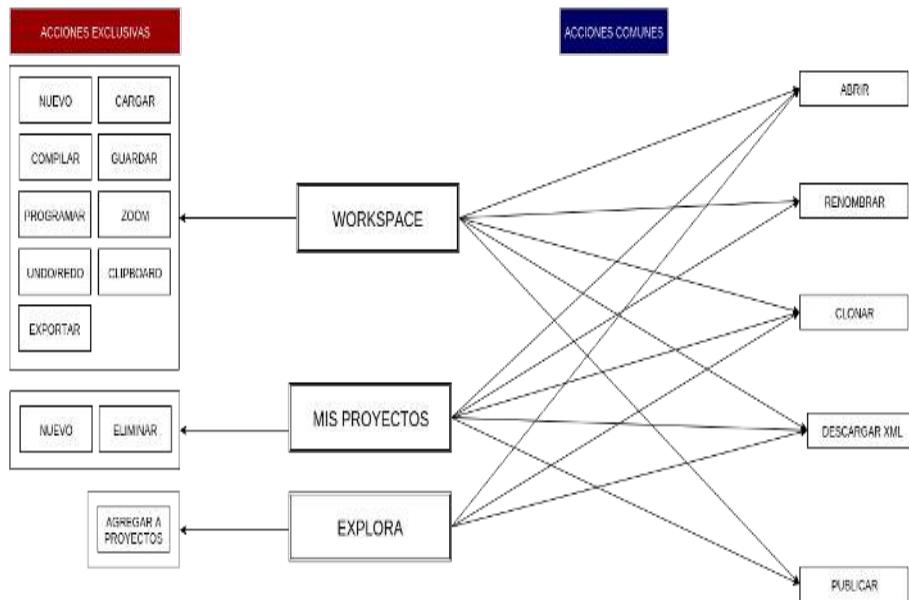


Bitbloq services A lo largo de todo el proyecto de Bitbloq, se repiten algunas acciones en diferentes contextos. La mejor forma de organizar estas acciones comunes es generando un servicio que pueda ser llamado en cualquier momento.

Al organizar estas acciones, se definen tres servicios muy importantes:

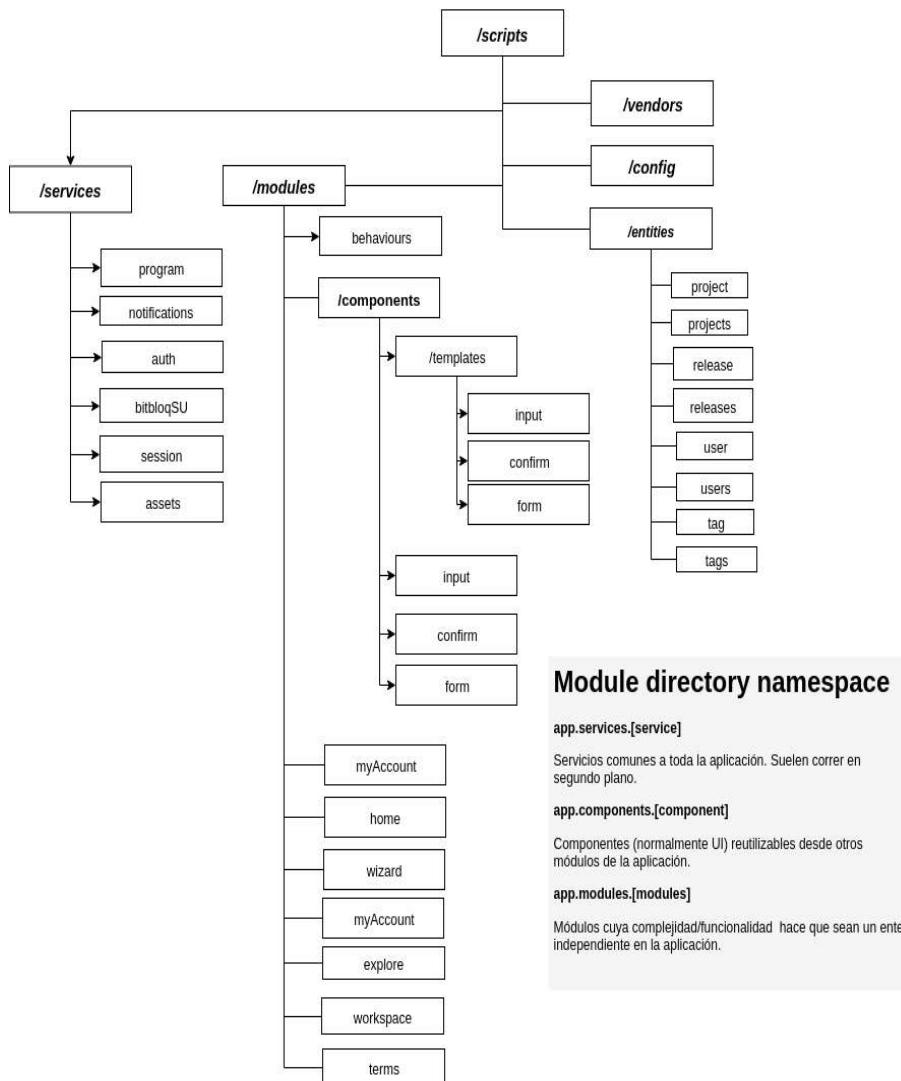
- **Workspace.** En este servicio se sitúan todas las acciones referentes a la creación y programación de proyectos, tales como cargar y exportar proyecto como compilar y programar.
- **Mis proyectos.** Corresponde con la vista que lista todos los proyectos del usuario. Las acciones que se sitúan en este servicio son referentes a los proyecto de una forma genérica, como nuevo o eliminar. En un principio estas acciones pueden confundirse con acciones dentro del servicio workspace, sin embargo, funcionalmente tienen grandes diferencias de comportamiento y, por ello, no se consideran la misma acción.
- **Explora.** En la vista con este mismo nombre se ofrecen acciones referentes a los proyectos publicados, como por ejemplo, agregar el proyectos públicos a tu lista de proyectos propios. Esta acción tan solo podrá ser efectuada desde dicha vista y, por esa razón, debe estar en un servicio propio de la vista.
- **CommonActions.** Este servicio es el correspondiente a acciones comunes a toda la aplicación. Un proyecto se podrá abrir, clonar o descargar su xml desde el workspace, desde la ventana de mis proyecto o incluso abrir un proyecto público desde explora. Las acciones de abrir, renombrar, o publicar un proyecto podrán ser ejecutadas desde el workspace como desde la ventana de mis proyectos.

A continuación, se detallan las acciones comunes y las exclusivas más importantes que pertenece a cada servicio.



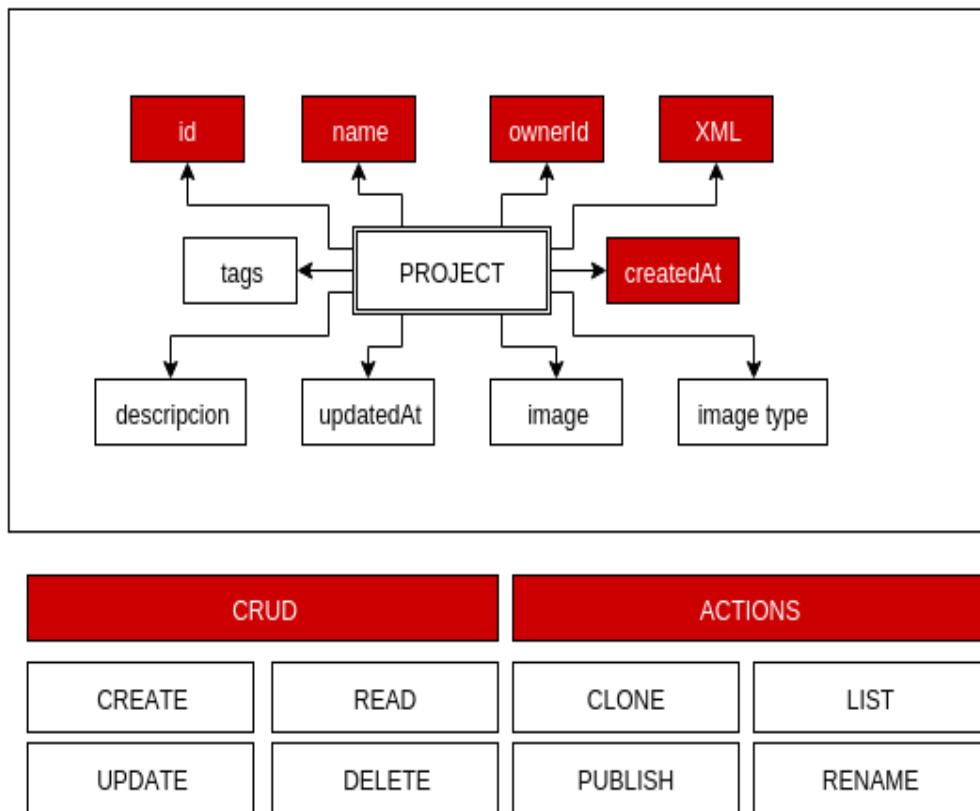
Tras detallar de una manera gráfica y sencilla las diferentes acciones que pueden alterar la estructura del proyecto, pasamos a detallar la nueva estructura del proyecto que servirá como esqueleto a lo largo de todo el desarrollo.

- Config: documentos de configuración de la aplicación y literales de traducciones.
- Entities: conjunto de los modelos que se usan y rellenan en el proyecto.
- Modules: módulos cuya complejidad y funcionalidad hace que sean un ente independiente en la aplicación.
- Services: servicios comunes a toda la aplicación.
- Vendors: servicios y bibliotecas externas que son usadas a lo largo de toda la aplicación.

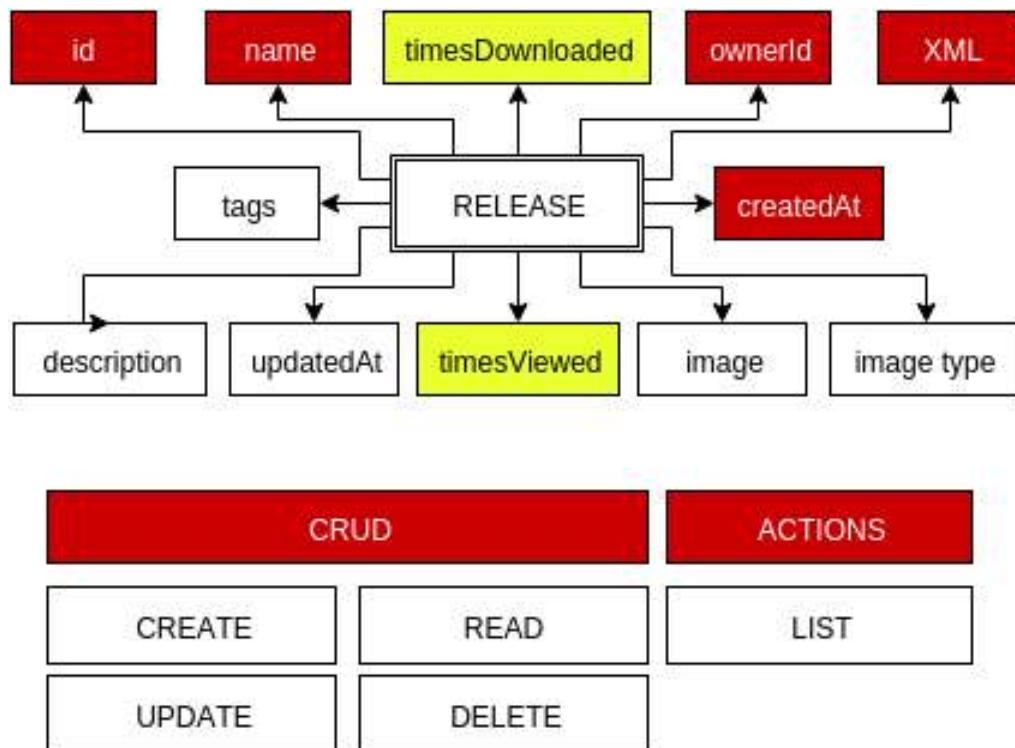


Esquemas de entidades

- Entidad proyecto.** El concepto proyecto es usado para nombrar los proyectos de los usuarios que son privados y tan solo pueden ser accedidos desde la sesión del usuario dueño del proyecto. Las acciones de clonar, publicar y renombrar son las que se pueden hacer en un proyecto privado y no en uno público. A continuación, se muestra el esquema de propiedades correspondiente a un proyecto privado.



- **Entidad release.** Se utiliza este concepto para definir una congelación de estado y código de un proyecto, el cual es publicado para que puedan verlo otros usuarios, es decir, una release es un proyecto público independiente a su correspondiente proyecto privado. A continuación, se muestra el esquema de propiedades correspondiente a un proyecto publicado.



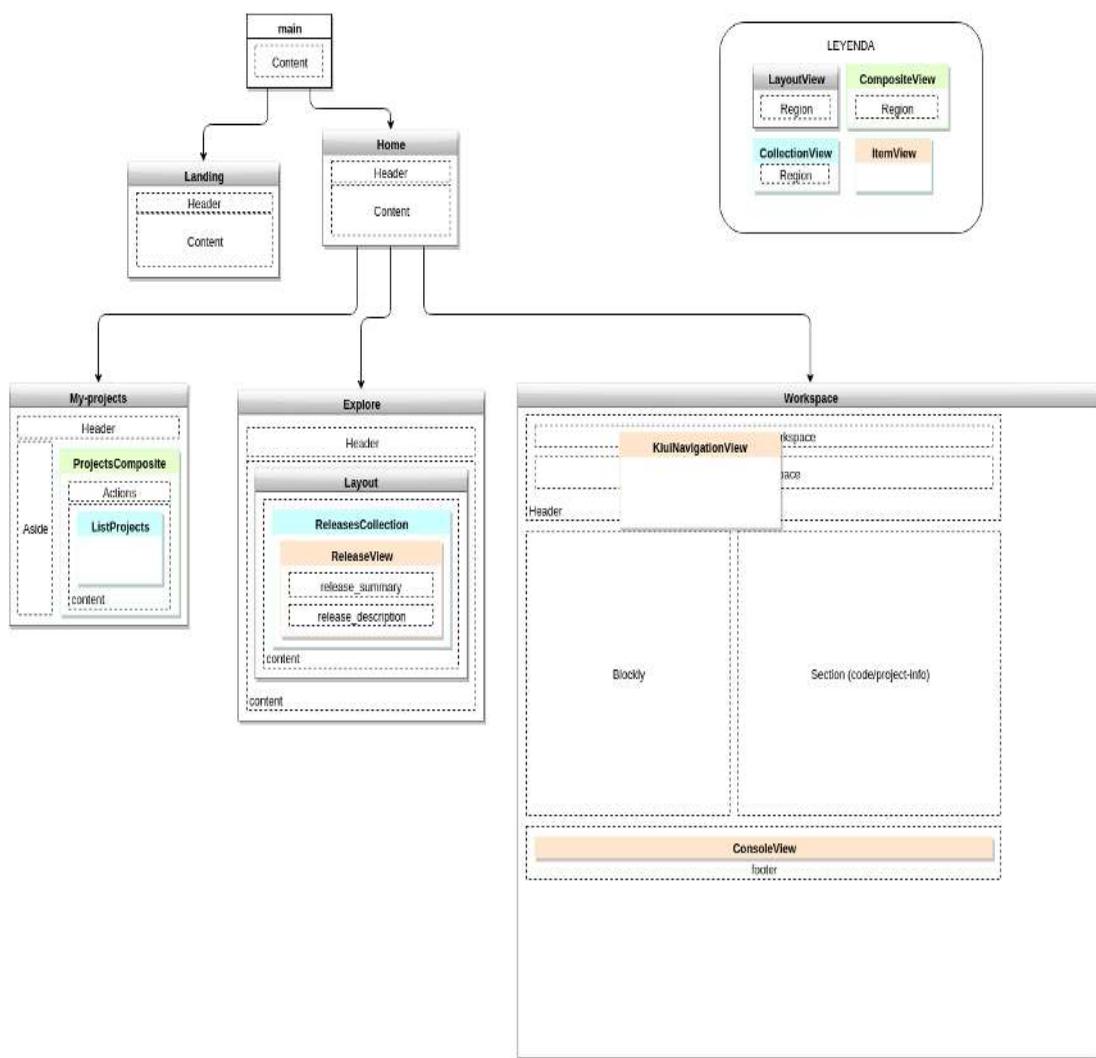
Antes de empezar el desarrollo del proyecto, hay que definir la estructuración de vistas y las relaciones que hay entre sí.

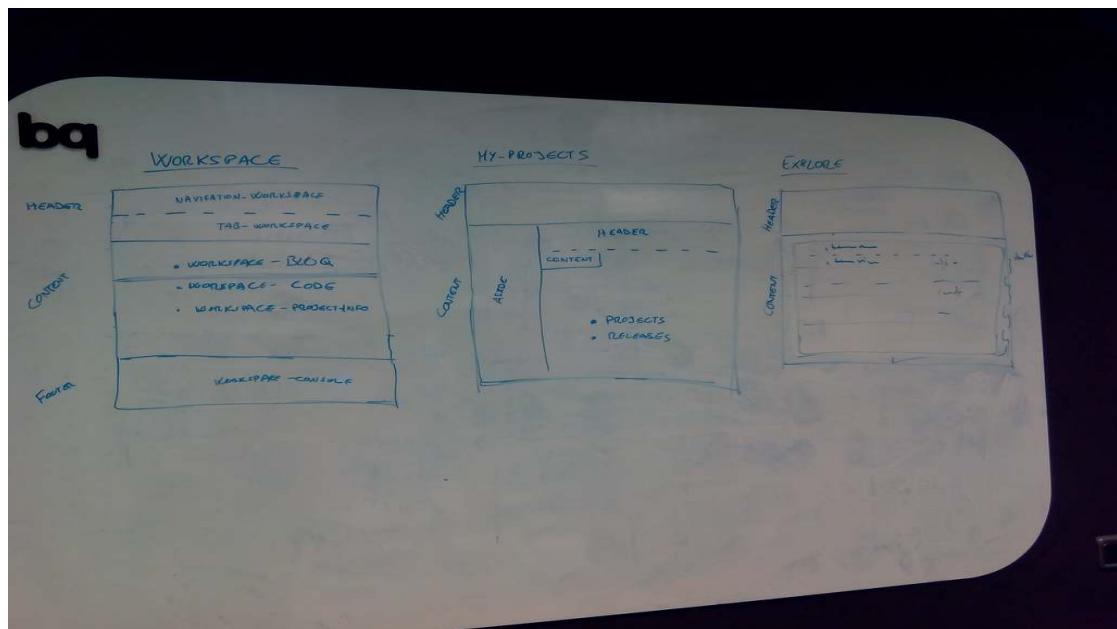
Una buena jerarquía sería especificar una estructura padre que posea una parte para el ‘header’ y otra para el contenido de la aplicación. A partir de este padre al que llamaremos ‘home’ podremos definir las layout de ‘myprojects’, ‘explore’ y ‘workspace’.

Para una mejor visualización y comprensión de la jerarquía a seguir se realiza el siguiente diagrama, donde aparecen los términos de LayoutView, CompositeView, CollectionView y ItemView.

- **LayoutView:** es una vista compuesta por múltiples vistas y componentes. Permite a la aplicación definir diferentes zonas y regiones para mantener una buena distribución.

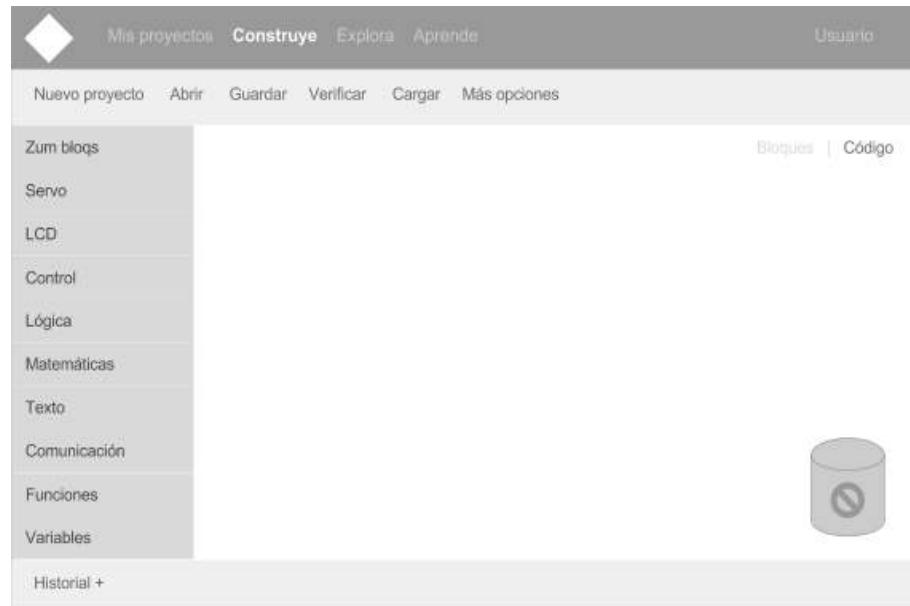
- ItemView: es una vista que representa un solo elemento. Este elemento puede ser un modelo o una colección de Backbone tratado como un solo elemento.
- CollectionView: es la manera de representar un conjunto ordenado de elementos de datos cambiantes y con un diseño flexible dentro de una vista.
- CompositeView: se extiende de CollectionView y es utilizado como vista compuesta para escenarios donde se debe representar tanto items como layouts.



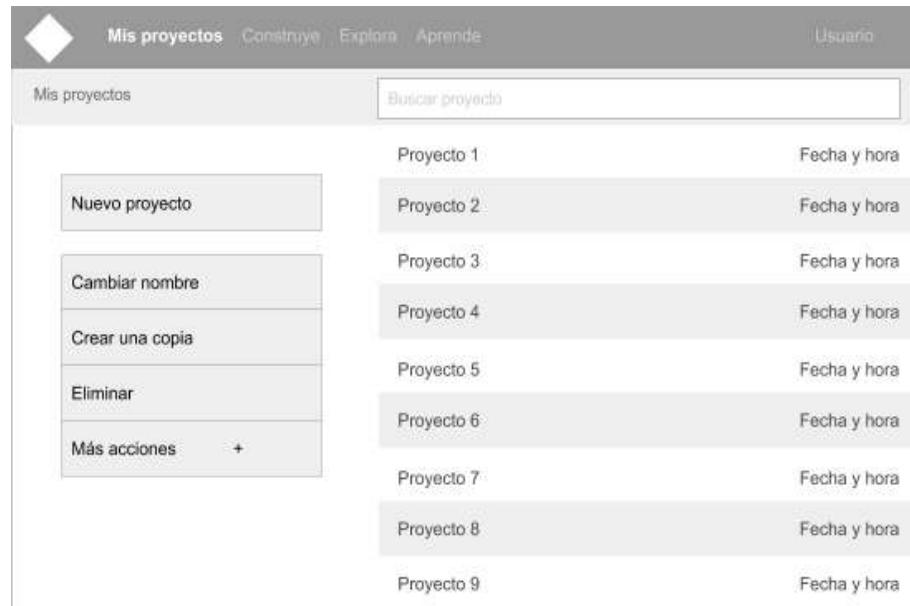


3.1.3 Wireframes

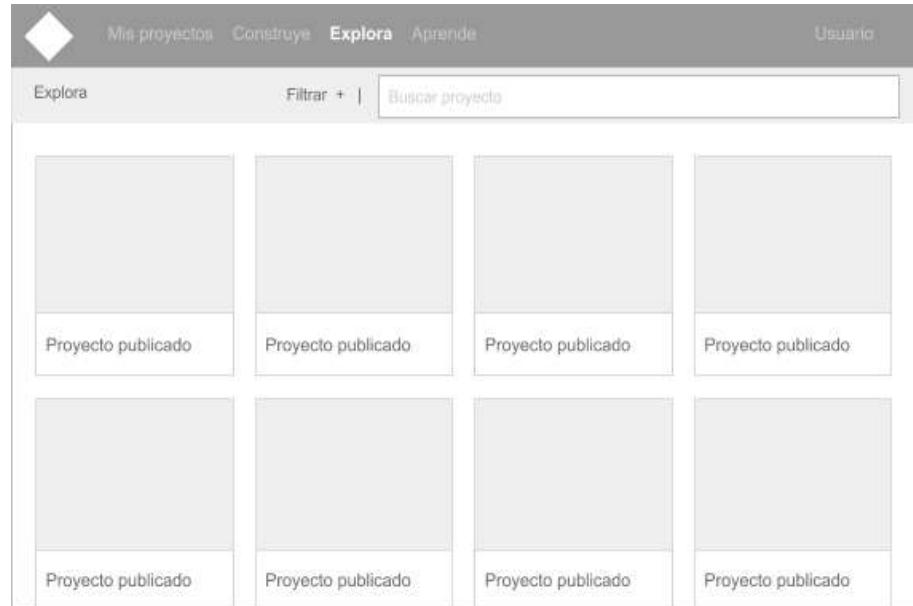
- Espacio de trabajo (Workspace)



- Mis proyectos

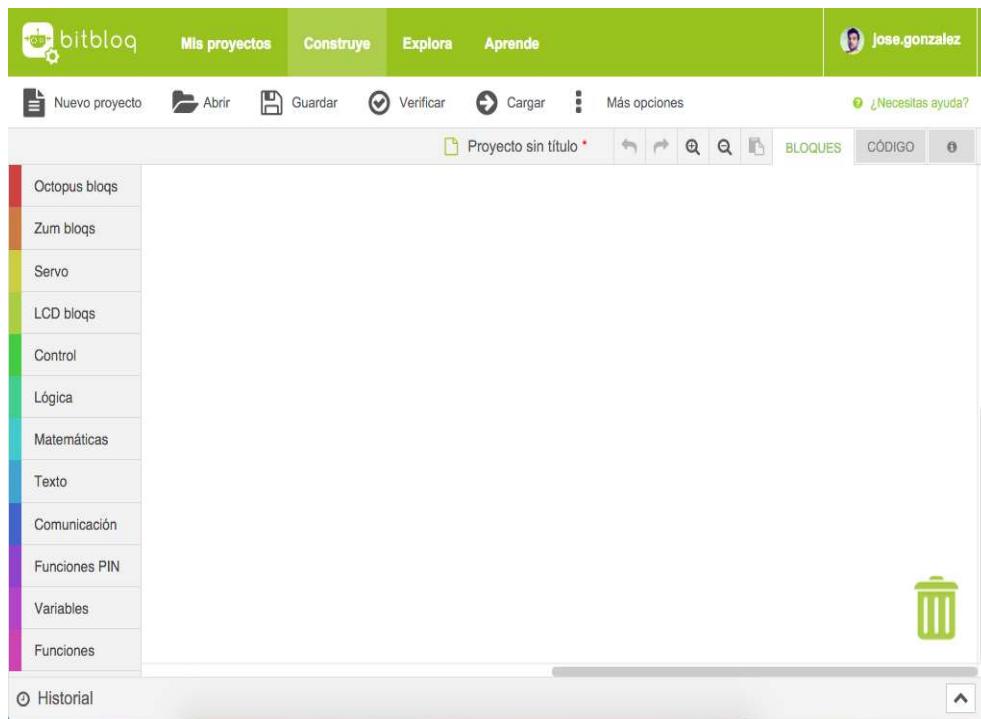


- Explora

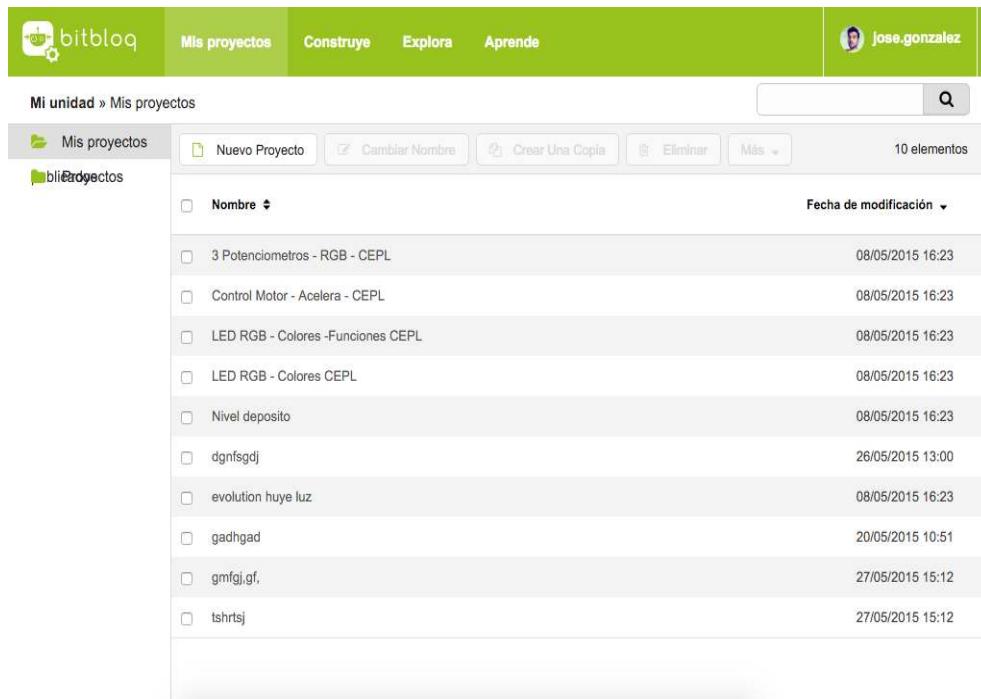


3.1.4 Diseños

- Espacio de trabajo (Workspace)



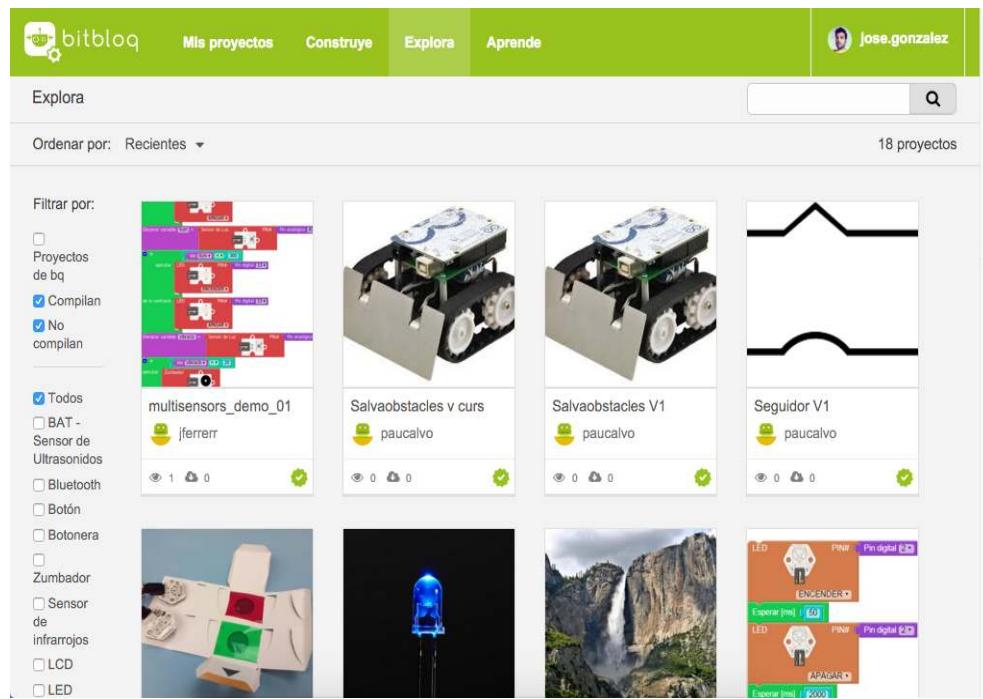
- Mis proyectos



The screenshot shows the 'Mis proyectos' (My Projects) page. At the top, there's a search bar and a button labeled 'Nuevo Proyecto'. Below the search bar, there are buttons for 'Cambiar Nombre', 'Crear Una Copia', 'Eliminar', and 'Más...'. To the right of these buttons, it says '10 elementos'. The main area displays a list of 10 recent projects, each with a checkbox, the project name, and the date of modification:

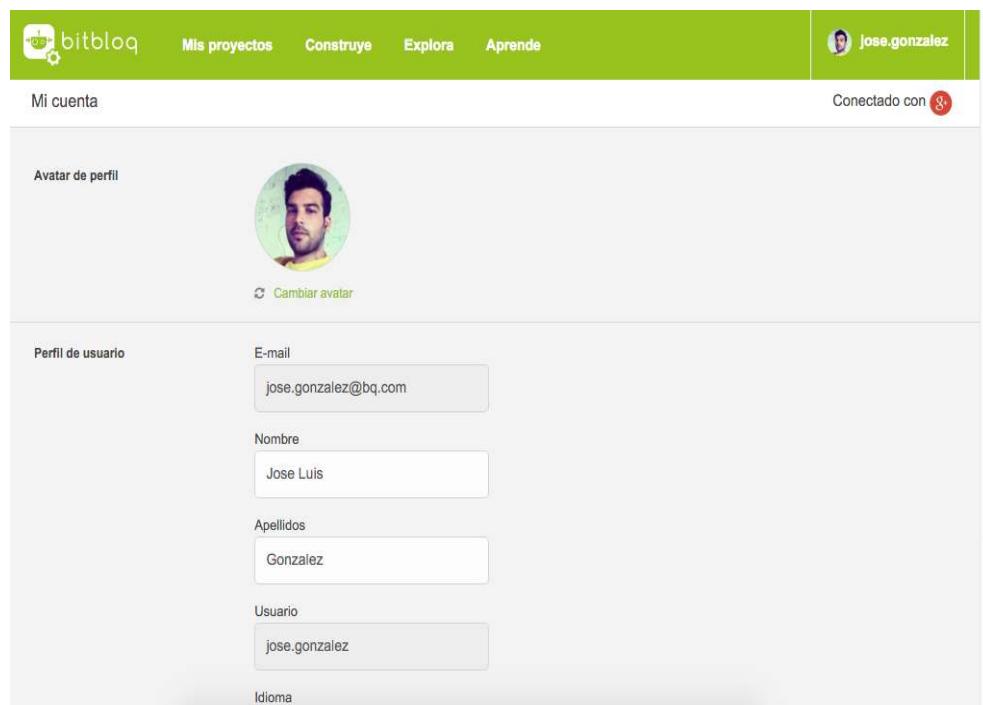
	Mis proyectos	Mis proyectos	Nombre	Fecha de modificación
<input type="checkbox"/>	3 Potenciómetros - RGB - CEPL		08/05/2015 16:23	
<input type="checkbox"/>	Control Motor - Acelera - CEPL		08/05/2015 16:23	
<input type="checkbox"/>	LED RGB - Colores -Funciones CEPL		08/05/2015 16:23	
<input type="checkbox"/>	LED RGB - Colores CEPL		08/05/2015 16:23	
<input type="checkbox"/>	Nivel deposito		08/05/2015 16:23	
<input type="checkbox"/>	dgnfsgdj		26/05/2015 13:00	
<input type="checkbox"/>	evolution huye luz		08/05/2015 16:23	
<input type="checkbox"/>	gadhgad		20/05/2015 10:51	
<input type="checkbox"/>	gmfgj,gf,		27/05/2015 15:12	
<input type="checkbox"/>	tshrtsj		27/05/2015 15:12	

- Explora



The screenshot shows the 'Explora' section of the botbloq website. At the top, there's a navigation bar with 'Mis proyectos', 'Construye', 'Explora' (which is highlighted in green), and 'Aprende'. A user profile for 'Jose.gonzalez' is on the right. Below the navigation, it says 'Explora' and 'Ordenar por: Recientes'. There are 18 projects listed. On the left, there's a sidebar for filtering projects by category: 'Proyectos de bq', 'Compilan' (which is checked), 'No compilan', 'Todos' (checked), and several other options like 'BAT - Sensor de Ultrasonidos' and 'LED' which are unchecked. The main area shows four rows of project cards. Each card includes a thumbnail image, the project name, the author's name, and some statistics (views, likes, comments). Some cards also show a small code snippet or diagram.

- Cuenta de usuario



The screenshot shows the 'Mi cuenta' (My Account) section of the botbloq website. At the top, there's a navigation bar with 'Mis proyectos', 'Construye', 'Explora', and 'Aprende'. A user profile for 'Jose.gonzalez' is on the right, with a note 'Conectado con g+'. Below the navigation, it says 'Mi cuenta'. It features a circular 'Avatar de perfil' (Profile Avatar) with a placeholder image of a man's face. Below the avatar, there's a link 'Cambiar avatar'. The main area contains several input fields for user information: 'Perfil de usuario' (User Profile) with fields for 'E-mail' (jose.gonzalez@bq.com), 'Nombre' (Jose Luis), 'Apellidos' (Gonzalez), 'Usuario' (jose.gonzalez), and 'Idioma' (Language). To the right of these fields, there's a large empty space.

3.1.5 Chrome App (bitbloq Serial Uploader)

Originalmente desde una aplicación web, por motivos de seguridad, no se tiene control directo de los puertos serie de un ordenador.

Principalmente, el objetivo de este módulo software es cargar los programas desde la aplicación web a las placas (Arduino UNO, Freaduino, ZUM), a través del USB, en varios SO (Windows, Linux, Mac), sin tener toda la lógica de selección de puertos y placas.

Tras estudiar distintas opciones, decidimos utilizar una [ChromeApp](#), la cual está alojada en [ChromeWebStore](#) (<https://goo.gl/1uKJYC>) y es necesario que el usuario la instale para poder usar Bitbloq. Se le asignó el nombre de *bitbloq Serial Uploader*.

Encontramos las siguientes ventajas:

- Fácil de instalar
- Se actualiza de forma transparente
- Podemos utilizar la API `chrome.serial` para comunicarnos con cualquier puerto serie.
- Fácil de desarrollar
- Es agnóstica al sistema operativo

Y las siguientes desventajas:

- Sólo funciona en Chrome

De cara al usuario, la primera versión de la Chrome App detecta los puertos usb del usuario y permite a éste seleccionar uno. Además, permite seleccionar una placa (Arduino UNO, Freaduino UNO, bq ZUM).

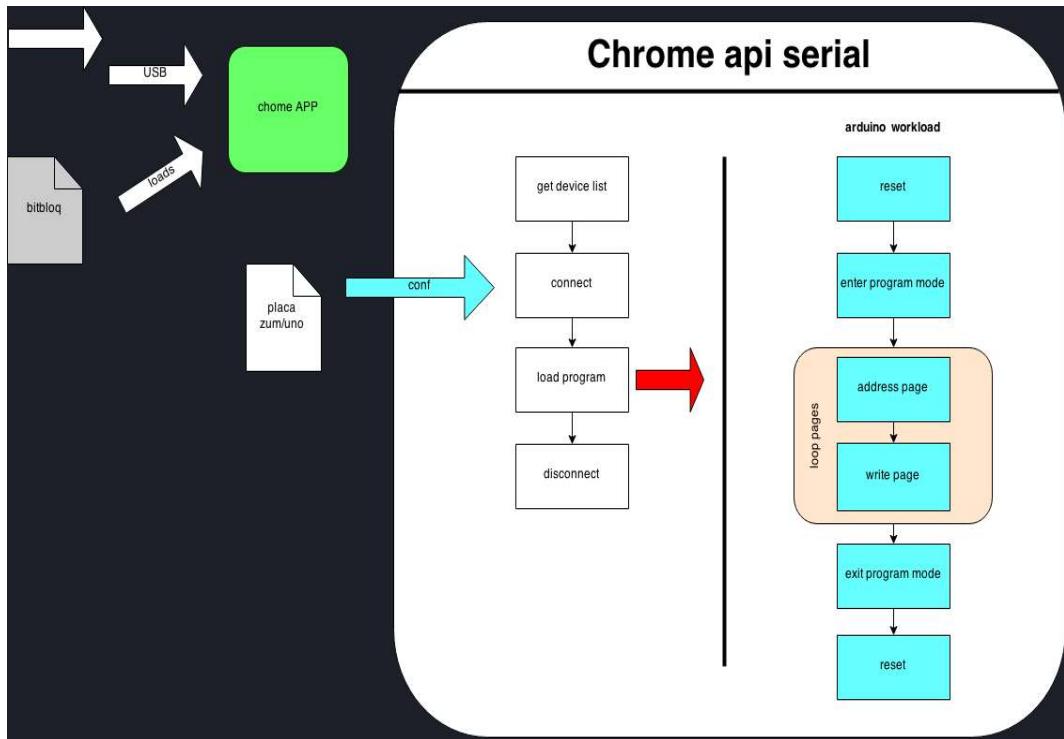
Componentes El proyecto BitbloqSU se ha dividido en los siguientes módulos:

- **Módulo Message:** Este módulo se encarga de gestionar la comunicación con bitbloq, en base a las request que éste le hace. La lista de posibles mensajes que se pueden intercambiar se representa en la siguiente tabla:

Mensajes desde bitbloq	Respuestas ChromeApp
version	<String> version
getPorts	<Array> ports
setPort	connectingport:ok
	connectingport:ko
program	program:ok
	program:error:size -> not enough space in board
	program:error:connection -> cannot connect to board (not connected or missing drivers)
	program:error:write -> problem when writing board
close	X

- **Módulo Program:** Este módulo se encarga de preparar el binario compilado para ser almacenado en la placa. Una vez preparado este binario, es también desde aquí donde se establece el flujo de carga en la placa, gracias a las API que encapsula el módulo serial.

En el siguiente diagrama se muestra como se establece el flujo de programación y que funciones intervienen.



- **Módulo Serial:** interfaz de bajo nivel que conecta con la placa y envía los datos.

Instalación del entorno

1. Clonar el repositorio de bitbloq-serial-uploader.
2. Una vez descargado el código, hay que instalar la chromeApp en modo desarrollo accediendo a **Chrome -> Configuración -> Extensiones** y haciendo click en el checkbox "**Modo desarrollador**". A continuación, se hará click en "**Cargar Extensión Descomprimida**".



1. Se selecciona el directorio donde está descargado el proyecto bitbloq-serial-uploader (debe ser la carpeta que contiene el **manifest.json**). Una vez cargada la chromeApp de desarrollo, aparecerán las siguientes opciones:
 - **Iniciar**: lanza la chromeApp.
 - **Volver a cargar**: carga la ChromeApp si se ha modificado el código.
 - **ID**: identificador de la chromeApp (se usará posteriormente en el proyecto bitbloq). Se genera según el path donde se haya descomprimido.
 - **index.html**: abre las *Developer Tools* de Chrome para debugear.

Enlazar bitbloq con ChromeApp Para poder lanzar bitbloq con la chromeApp de desarrollo local, tenemos que copiar el ID de la chromeApp (Instalación del entorno - paso 3 - ID) y escribirlo en el fichero:

bitbloq-corejs-app/src/main/webapp/res/config/config.json.

Con ello se lanzará automáticamente apuntando a la ChromeApp de desarrollo.

```
{  
...  
"chromeappId": "flihcoiblmfhmhcfdhkcadbphphcbki"  
...  
}
```

Agregar localhost en el URL Handlers Para poder lanzar la chromeApp desde bitbloq en local, en el archivo "**src/manifest.json**" de la chromeApp debemos agregar localhost de la siguiente manera:

Bloque de código

```
"url_handlers": {  
    "bitbloq": {  
        "matches": [  
            "http://localhost:9000/chromeapp.html",  
            "http://localhost:9000/favicon.ico"  
        ]  
    }  
}
```

Depuración Para poder depurar y ejecutar únicamente la ChromeApp sin necesidad de depender de Bitbloq, existen unos *mocks* creados específicamente para esto en el módulo Messages. Para ello, se conecta la placa y se abre la consola del navegador. Desde la consola ejecutaremos cualquiera de estos mocks (hay uno disponible por cada placa).

- `Messages.programMock('freaduinoUNO')`
- `Messages.programMock('ZUM')`
- `Messages.programMock('arduinoUNO')`

Despliegue La ChromeApp está alojada en Chrome Web Store de Google. Desafortunadamente no tenemos implementada la subida automática, por lo que tenemos que hacerlo manualmente.

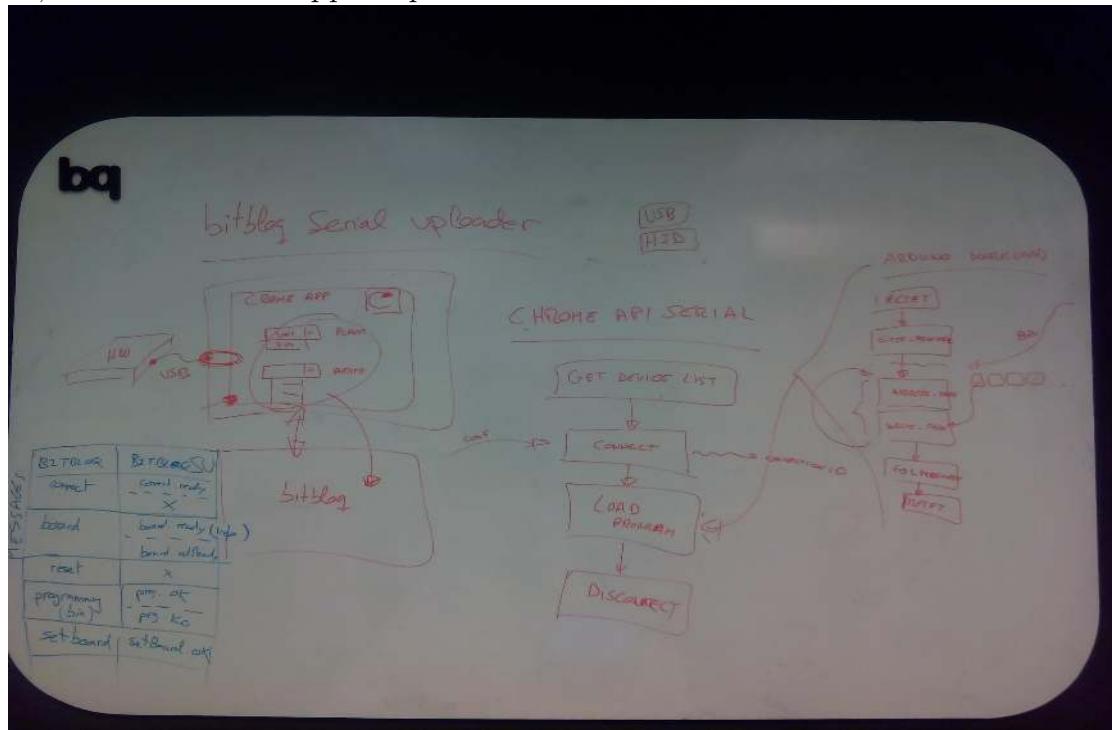
Existen 2 versiones de chromeApp:

- bitbloq Serial Uploader DEV: Aplicación de desarrollo, **no mostrada**
- bitbloq Serial Uploader: Aplicación de producción

El componente que se sube es un build de la chromeApp. Por lo que tenemos que generarla:

1. Modificamos el manifest.json (src/manifest.json) con los siguientes parámetros:
 - Versión: Incrementar la versión siguiendo la convención establecida
 - Name: Añadimos la extensión "DEV" en el caso que subamos una versión de desarrollo.
2. Generamos el zip ejecutando: *grunt*
3. Finalmente tendremos un zip en dist/app_dist_{VERSION}.zip que deberemos subir a través de la plataforma de administración de la Chrome Web Store.

Funcionamiento Si el usuario introduce la combinación correcta de puerto/placa, la chrome app se encarga de enviar el código compilado hasta la placa. De lo contrario, notificará a bitbloq para que muestre un error al usuario.



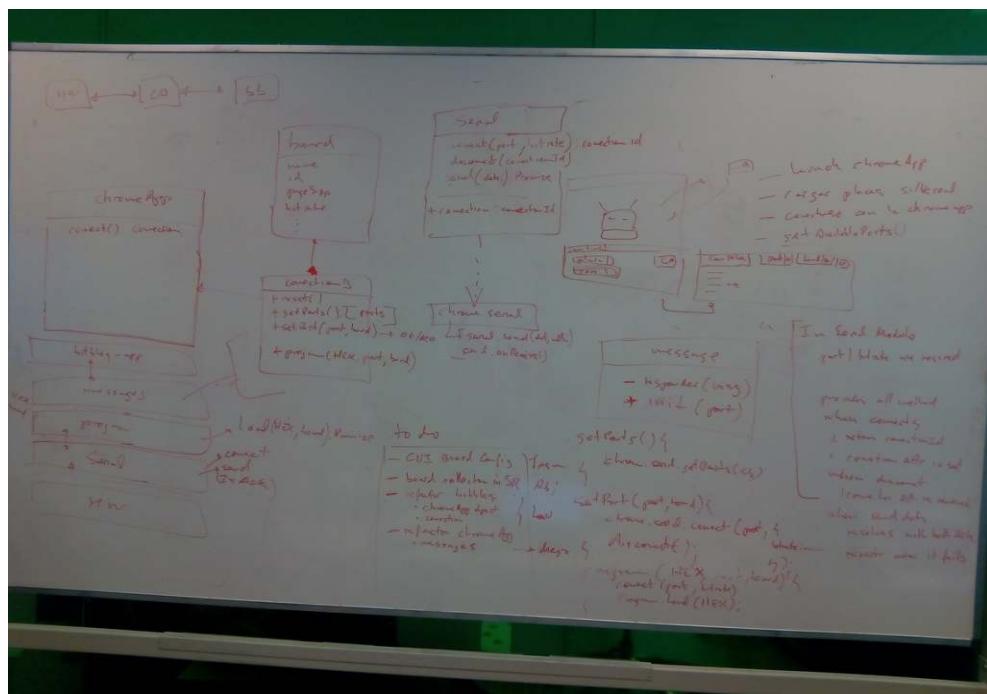
Internamente, la Chrome App se comunica por un lado con la aplicación y por otro con la placa, de forma que puedan intercambiar eventos y responder ante los mismos. Para ello, utiliza la API de chrome.serial.

Algunos de estos eventos son:

Bitbloq	Chrome App
Connect	Connect.Ready / X
Board	Board.Ready(Info) / BoardNotReady
Reset	X
Programming	Prog.OK / Prog.KO
SetBoard	SetBoard.OK / X

Nos encontramos con distintas dificultades a la hora de escribir en las placas. Por un lado había que tener en cuenta los tamaños de las páginas de la memoria, el bitrate, etc. Por otro había que tener cuidado con la velocidad a la que escribíamos en la placa, ya que se producían problemas de sincronización. Además, algunas placas son muy sensibles y a veces se quedaban colgadas, por lo que había que resetearlas de forma transparente al usuario.

Refactor bitbloqSU La segunda versión de la Chrome App surge de la necesidad de estabilizar su funcionamiento, así como facilitar al usuario la tarea de enviar su programa a la placa.



Es más transparente todavía, ya no es necesario seleccionar puerto/placa y nada más abrirse se minimiza para no invadir al usuario.

Además, mejora mucho el tiempo de espera del usuario, reduciéndolo a un máximo de 15 segundos desde que el usuario hace clic en “cargar” hasta que el programa se carga en la placa definitivamente.

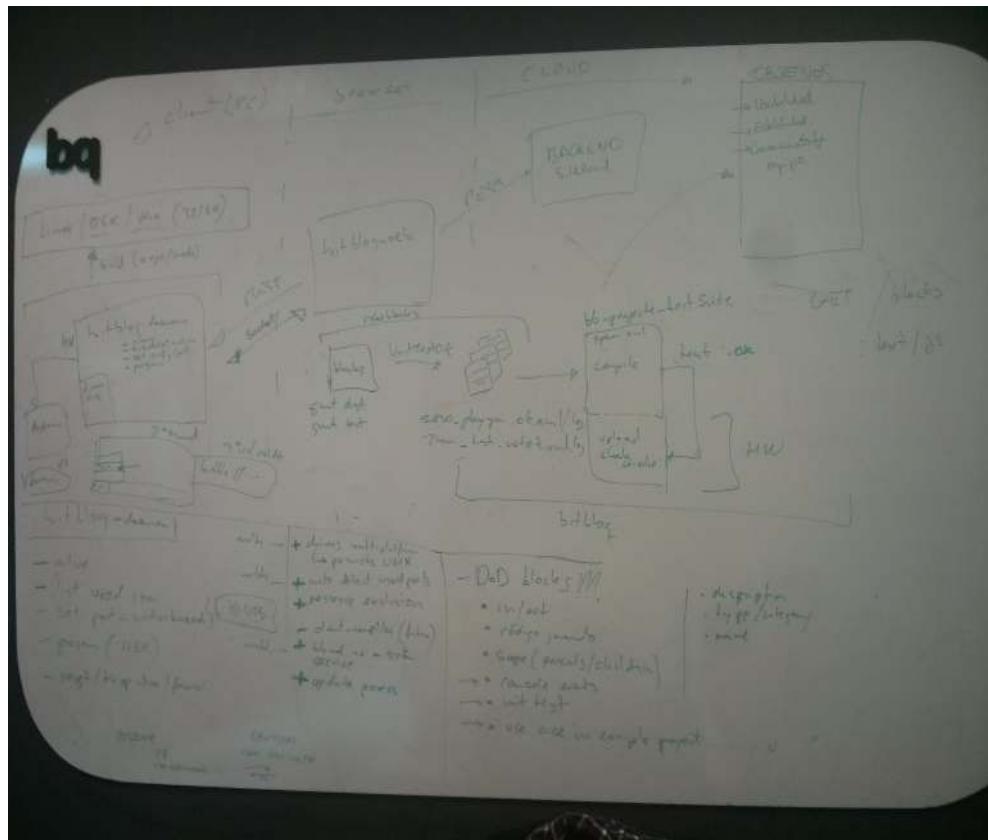
Técnicamente, pasamos de utilizar programación síncrona a programación asíncrona funcional, gracias al buen uso de las promesas de Javascript. De esta forma reducimos el número de timeouts necesarios para detectar o escribir en las placas.



Para alcanzar este objetivo tuvimos que volver a empezar, ya que la primera versión arrastraba deuda técnica debido a su nivel experimental.

Esta vez tuvimos en cuenta los problemas de sincronía con la placa desde el principio y simplificamos considerablemente el flujo de instalación. Ya sólo era necesario un clic para instalar la Chrome App.

Además, reducimos considerablemente el tiempo transcurrido desde que el usuario hace clic en el botón de “cargar” hasta que el programa queda definitivamente cargado en la placa.



3.1.6 Tecnologías usadas

En la primera fase, Bitbloq v1, se optó por el uso de Backbone.js para dar forma a la aplicación. Sin embargo, Backbone necesita un gran apoyo de otras bibliotecas y framework. Por esta razón, se decidió incorporar librerías como Marionette o handlebar y desarrollar un módulo independiente que sirviese como base y estructura al proyecto y que pudiese servir de pilar en la construcción de la aplicación, consiguiendo así una mejor velocidad de trabajo.

Este módulo de apoyo es coreJS que formó parte del núcleo de toda la aplicación haciendo más fácil la integración con el servicio de Back-end y la base de datos. Además, al ser independiente de la aplicación, podría tener una gestión y desarrollo autónomo, cosa que beneficiaría al desarrollo global de toda la aplicación. En el anexo II. CoreJS se explicará más en profundidad en qué consiste el módulo base coreJS.

Además, nos hemos apoyado en la plataforma Corbel que nos ofrece lo necesario para desarrollar Bitbloq de una forma rápida sin preocuparnos de la parte del Backend. En el anexo I. Infraestructura de servicios se explicará más en profundidad en qué consiste esta plataforma.

3.1.7 Convenios y estilos

Es importante establecer un estilo de código unificado para mantener un mínimo de calidad, mantenibilidad y legibilidad del código.

El objetivo es que el código parezca que ha sido desarrollado una única persona, sin importar el número de miembros en el equipo de desarrollo. En los proyectos web nos centraremos en los siguientes lenguajes:

Los desarrolladores que usan Sublime, pueden beneficiarse del plugin HTML-CSS-JS Prettify, que ya se encarga de formatear el código siguiendo las convenciones más extendidas.

Hojas de Estilos

- Archivos

```
_this-is-a-large-name-component.scss // componentes app  
this-is-a-large-name-app-styles.scss // estilos generales de la app
```

- Selectores

```
/*  
* BEM  
* Stands for "Block", "Element", "Modifier"  
*/
```

```
.nav           // bloque  
.nav__item {} // elemento de bloque
```

```
.nav__item--selected {} // modificador de elemento de bloque  
.nav--secondary {} // modificador de bloque  
.nav--secondary__item {} // elemento de bloque con modificador
```

Ejemplo:

```
.nav {}  
.nav__item {}  
.nav__item--selected {}  
.nav--secondary {}  
.nav--vertical__item {}
```

- Imágenes

```
this-is-a-large-name-folder // folder img  
this-is-a-large-name-image.png // img
```

- JavaScript

La convención de código que seguimos para el desarrollo de JavaScript será la definida en IdiomaticJS. A grandes rasgos:

```
// fileNameLikeThis.js
```

```
var CONSTANTS_LIKE_THIS = 3.14;  
var variableNamesLikeThis = true;  
var functionNamesLikeThis = function() { };  
var ConstructorNamesLikeThis = function() { };  
var objectNameLikeThis = new ConstructorNameLikeThis();  
var objectNameLikethis.methodNamesLikeThis = function() { };
```

3.1.8 Features y resumen del backlog de la Fase I

Feature	Estado	Comentarios
Fase I		
Login FB	Closed	
Login Google	Closed	
Login OAuth	Closed	
Registro FB	Closed	
Registro Google	Closed	
Registro OAuth	Closed	
Recordar	Closed	
Recuperar pwd	Closed	Bug al recuperar pwd con redes sociales -> Edge case
Salir de mi cuenta	Closed	
AutoSave en local	Closed	
Guardar Cloud	Closed	
Abrir Cloud	Closed	
Cargar un proyecto(local)	Closed	
Descargar proyecto	Closed	
Exportar código arduino	Closed	
Crear nuevo proyecto	Closed	
Integración de blocky	Closed	
Eliminar proyecto	Closed	
Bloques Nativos	Closed	
Bloques Compuestos	Closed	
Compilar proyecto	Closed	
Sistema de carga de bloques	Closed	
Zoom	Closed	
Deshacer	Closed	
Cambiar idioma	Closed	
Añadir/Eliminar tags	Closed	
Buscar proyecto por tag, user y descripción	Parcial	Buscamos sólo por nombre

Convertir bloques en tags	Closed	
User profile	Closed	
Puerto Serie	Closed	
Añadir rol invitado	Closed	
Help	Closed	
Home	Closed	

3.1.9 Improvements tras salida a producción

Feature	Comentarios	Prioridad
Permitir en usar bitbloq en firefox excepto programar	Al pulsar programar, saldrá modal informando de que sólo se puede programar la placa desde Chrome.	P0
Permitir ver la landing en resto de navegadores navegadores	Al pulsar en "Entrar" o "Empezar a jugar" saldrá un modal de que sólo se puede acceder desde Chrome o Firefox.	P0
Traducir al portugués		P1
Al pulsar botón derecho sobre un bloque el menú aparece en inglés (mientras bitbloq está en español). Mi OS está en inglés		
Al introducir la info del proyecto y guardar cambios no vuelve a la página "construye". Ni siquiera pulsando sobre el menú "construye"		

He cambiado mi avatar (alias bq) pero sigue apareciendo el robot por defecto en los proyectos públicos (ver blink)		
La descripción de los proyectos no mantiene los retornos de carro		

3.1.10 Redefinición de producto tras la primera fase de Bitbloq v1

Bitbloq ha conseguido establecerse como una herramienta para programar electrónica de una forma más visual e intuitiva, uniendo mediante bloques visuales de código y teniendo un ecosistema de proyectos en una comunidad. Aún así el proyecto necesita crecer e introducir nuevas funcionalidades para seguir avanzando e ir hacia un modelo de creación más interactivo, sencillo y con muchas más posibilidades.

Una herramienta escalable que pueda ser desde el inicio del aprendizaje de un niño de 8 a 12 años hasta una herramienta de programación para profesionales ilimitada mezclando bloques visuales y código propio.

Problemas encontrados en la V1 En la versión 1 de Bitbloq hemos encontrado mediante casos de uso, observatorios y talleres que hay algunas cualidades de la herramienta que dificultan el conseguir los objetivos indicados como principales para ser una herramienta fácil, intuitiva y escalable.

Algunos de esos problemas se describen a continuación:

- **La curva de aprendizaje de los usuarios es elevada** y tiene una gran dependencia de ayuda externa de un tutor o adulto. Aunque la forma de programar mediante bloques visuales sea más fácil que programar, el entender el concepto es complicado debido a muchos tecnicismos y conocimientos previos necesarios para conseguir programar algo sencillo.

- La herramienta no une el **mundo de la electrónica creada por BQ y la herramienta** en sí, por lo que no potencia las posibilidades de creación de inventos o robots. Además los componentes que aparecen en la herramienta son difícilmente identificables con su versión real, dificultando la asociación entre lo que ven en pantalla y lo que tienen en sus manos.
- La **Chrome App** que consigue cargar el programa de Bitbloq en la placa conectada al ordenador tiene muchas dependencias del servidor de Google y de la conexión de internet, además de errores de interacción debido a las interferencias que se crean en los flujos de carga y actualización.
- La **comunidad** tiene muchos proyectos pero no destacables. Hay muchos proyectos repetidos y/o con poca información sobre el programa. Además no se puede hacer una busca efectiva o más filtrada de los proyectos.
- Los **bloques visuales de código** son confusos porque mezclan muchas características de programación que necesitan ser explicadas con anterioridad para llegar a comprenderse. Además de crear confusión y de no seguir una estructura de orden y lógica cómo puede llevar un programa de código.
- No existe **ayuda online** para resolver dudas o ayudar en caso de errores o de que no se consiga realizar un ejercicio de programación. Los usuarios se quedan con la duda y no pueden resolverse si no es con ayuda externa a la herramienta.

Tras la salida a producción de la primera versión de Bitbloq, se pensaron en algunas nuevas funcionalidades que podían ser integradas en la plataforma, orientando ésta a entornos docentes. En concreto, la nueva visión de producto se focaliza en crear diferentes roles de usuarios, distinguiendo entre usuarios ‘profesores’ y usuarios ‘alumnos’ (Está fase no ha sido implementada todavía)

Visión de nuevos roles de usuarios Existen cuatro tipos de usuarios en Bitbloq:

- **Usuarios Convencionales**

Bitbloq tendrá una gestión de usuarios convencionales que permitirá:

1) Registro/Login por tres métodos:

- Email+Password
- Facebook Login
- Google Login

2) Los usuarios convencionales, pueden administrar su perfil.

3) Los Perfiles de Usuario Convencional incluyen actualmente la siguiente información (completar a partir de la última versión en integración):

- Name
- Email
- Password

4) El usuario puede editar su perfil, pudiendo cambiar:

- Name
- Email
- Password

5) Asociado con un usuario está su espacio de almacenamiento en la nube (Cloud Privado), en el que puede almacenar y desde el que puede cargar ficheros de proyecto. Los proyectos tienen etiquetas que nos permiten clasificar y buscar, y además pueden tener ámbito Privado o Público. Los proyectos privados sólo los pueden ver los usuarios que los crearon. El Administrador de Bitbloq sólo puede eliminar archivos públicos subidos al cloud. Sólo pueden eliminar aquellos proyectos de los que son autores.

- **Profesores de colegios**

Bitbloq tendrá una gestión de usuarios específica para Profesores de Colegios:

1) Registro/Login por tres métodos:

- Email+Password
- Facebook Login
- Google Login

2) Se registra como usuario convencional. Hay un Call-To-Action en el UI (¿Eres Profesor?) que muestra una página informativa en la que se habla del proyecto educativo bitbloq. El usuario puede, desde aquí, enviar una solicitud a bq para tener acceso a las opciones de profesor. Para ello rellenará un formulario de solicitud con datos que permitan al personal de BQ verificar la identidad de la persona y su pertenencia a la comunidad de enseñanza (NOMBRE APELLIDOS COLEGIO DEPARTAMENTO CURSO ASIGNATURA Nº Colegiado TELÉFONO...); . Si el usuario envía el formulario, el administrador de bitbloq recibe una notificación por email con un enlace a la ficha del usuario en cuestión. Puede decidir llamarle o pedirle más información. Si está todo correcto, puede autorizarle marcando simplemente un check de que es profesor. El usuario recibe una notificación de que su solicitud ha sido aprobada. Si se desactiva esta opción en cualquier momento, el usuario también recibirá una notificación de que pierde a partir de ese momento sus privilegios por los motivos que sean.

3) Los usuarios profesores, igual que los convencionales, pueden administrar su perfil.

4) Los Perfiles de Profesor permiten realizar gestiones adicionales a las de los usuarios convencionales:

- Gestionar Clases y Alumnos

5) La gestión de Clases es una opción de menú que sólo aparece a los profesores. Desde esta parte de la herramienta se permitirá:

- Crear, Editar y Eliminar CLASES
- Crear, Editar y Eliminar ALUMNOS que se quedan asignados a esas CLASES

La creación de alumnos es automática. Dado que pueden tratarse de menores, no se recoge ningún tipo de email. Por tanto los alumnos se validan únicamente con un apodo y palabra de paso.

Una clase tiene un cloud separado del público, al que pueden acceder sus alumnos. Por tanto un alumno podría guardar proyectos en cloud público (lo ve el mundo), privado (lo ve sólo él), o el de su clase (lo ven sólamente los alumnos que pertenecen a esa clase).

La gestión de clases permite el uploading de ficheros de proyecto a los cloud de las clases.

De momento, un profesor puede crear varias clases, pero un alumno sólo puede pertenecer a una clase. En el futuro cambiaremos esto.

6) Asociado con un profesor está su espacio de almacenamiento en la nube (Cloud Privado), en el que puede almacenar y desde el que puede cargar ficheros de proyecto. Los proyectos tienen etiquetas que nos permiten clasificar y buscar, y además pueden tener ámbito Privado, Público o de Clase. Sólo pueden eliminar aquellos proyectos de los que son autores.

- **Usuarios que son Alumnos de Colegios**

Bitbloq tendrá una gestión de usuarios convencionales que permitirá:

1) No tienen registro. Sólo Login por este método:

- Apodo+Password

2) Los usuarios Alumnos no pueden administrar su perfil, sólo lo pueden ver. En el perfil de un alumno aparecen los datos del COLEGIO Y CLASE a la que pertenecen, así como el PROFESOR asignado.

3) Asociado con un usuario está su espacio de almacenamiento en la nube (Cloud Privado), en el que puede almacenar y desde el que puede cargar ficheros de proyecto. Los proyectos tienen etiquetas que nos permiten clasificar y buscar, y además pueden tener ámbito Privado, Público o de Clase. Los proyectos privados

sólo los pueden ver los usuarios que los crearon. Sólo pueden eliminar aquellos proyectos de los que son autores.

3.2 Fase 2 - Bitbloq v2

3.2.1 Necesidades de la nueva plataforma

Para conseguir los nuevos objetivos en la versión 2, es necesario hacer un estudio de las necesidades y dificultades que se encuentra un usuario al entrar en la herramienta por primera vez y en momentos puntuales del proceso de programación.

Para ello hemos estado realizando talleres y observatorios con el fin de saber más acerca de la utilización de la herramienta y cómo es el proceso de aprendizaje y abordaje de un proyecto.

Las principales dudas que nos surgen son:

- ¿Qué modelo pedagógico sigue un tutor cuando introduce a un usuario en la herramienta?
- ¿Qué problemas se encuentra un usuario al entrar en la herramienta?
- ¿Qué problemas se encuentran al abrir un kit de robótica?
- ¿Qué es lo más complicado de aprender?
- ¿Y qué es lo más sencillo que aprenden?
- ¿Qué pasos o flujos siguen a la hora de programar una placa?
- ¿Qué cosas son imprescindibles para aprender?
- ¿Cuanto tiempo tardan en comprender la herramienta?

Estas son algunas de las principales dudas que necesitamos resolver para poder marcar unas pautas y directrices a la hora de rediseñar la herramienta enfocada al usuario.

En los talleres y observatorios hemos sacado algunas conclusiones generales sobre el aprendizaje en este tipo de herramientas como son:

- Un usuario es activo cuando sabe que está aprendiendo algo nuevo, no se limita a escuchar y seguir las instrucciones, si no que tiene momentos de descubrimiento y búsqueda de los siguientes pasos por él mismo. Suele querer avanzar en el descubrimiento de la herramienta y se aventura a hacer click en zonas desconocidas.
- Cuando el aprendizaje es en un grupo de usuarios, se crea una comunidad automáticamente donde se descubren cosas en conjunto, se comunican entre ellos y se apoyan para ir avanzando en el aprendizaje y en los errores o dudas que puedan ir surgiendo en el proceso. Se crea una comunicación entre los usuarios para avanzar en común e ir descubriendo cada vez más posibilidades.
- Se ha observado que no todos los usuarios llevan el mismo ritmo de aprendizaje, pero por ello se apoyan en la comunidad para solucionar problemas o ir avanzando con su propio ritmo. Lo bueno de la comunidad es que nadie se queda atrás o desiste ya que siempre se encuentra apoyo de los demás.
- Durante los talleres realizados hemos introducido errores (problemas) en momentos puntuales para facilitar a los usuarios el desarrollo de la lógica abstracta y fomentar así su aprendizaje, favoreciendo el error o conflicto se les ayuda a buscar soluciones y aprender a utilizar y desarrollar la lógica.
- Después de las primeras explicaciones para encuadrar a los usuarios en la herramienta, el educador se llega a convertir rápidamente en un guía en el proceso de aprendizaje, ya que los usuarios crean su ritmo de aprendizaje y descubrimiento y el educador sirve de apoyo en momento puntuales, no como un educador que tiene que ir explicando todo el proceso para llegar a un objetivo.

- El usuario llega a ser cada vez más autónomo y es capaz de abordar los problemas o errores que pueden surgir en un momento clave con un pensamiento lógico y estructurado.
- Para los educadores es una buena forma de ver cómo el alumno piensa y aborda un problema de programación, teniendo rápidamente una explicación de cómo el alumno detecta el error, piensa sobre ello y elabora un plan b para solucionarlo.
- Como la herramienta no tiene limitaciones para crear programas cada vez más complejos, aquí se encuentra uno de los grandes potenciales para poder ir desarrollando programas más complejos sin necesidad de obtener más conocimientos ni de migrar a otro programa. Con la lógica y la imaginación el usuario es capaz de crear desde lo más sencillo hasta lo más complicado sin necesidades diferentes.
- La herramienta tiene un lenguaje lúdico, facilitando el juego y pudiendo probarlo con facilidad, sin temor al error ya que rápidamente se puede modificar, crear y eliminar.
- Una herramienta tiene que tener un inicio fácil, que no exija mucha complejidad o tiempo de aprendizaje. También tiene que poderse hacer proyectos para programación solamente, programación mezclada con electrónica simple y proyectos de electrónica complejos como robots. La herramienta va a servir para iniciarse en el mundo de la programación de electrónica pero tiene que servir para un usuario con conocimientos avanzados pueda programar algo complejo también.

- Los proyectos significativos aportan a la comunidad la capacidad de descubrir nuevos retos u horizontes al resto de usuarios. Además crear proyectos que sean útiles convierte lo que es solo aprendizaje y abstracto en algo palpable y usable. El tener una herramienta para crear programas abiertos y sujetos a modificaciones hace que se pueda personalizar, avanzar y explorar cada usuario con sus proyectos.

3.2.2 Objetivos

Con toda la información anterior y siguiendo unas pautas para hacer la herramienta más potente y útil vamos a definir unos objetivos en el rediseño de la herramienta:

- Unir la programación con la robótica.
- Hacer una herramienta más visual, fácil e intuitiva.
- Curva de aprendizaje más corta
- Introducción de metodología de programación
- Mejorar la experiencia de programación de la placa desde la herramienta
- Crear una herramienta escalable
- Introducción de nuevos productos
- Crear nueva librería de bloques
- Crear una comunidad con proyectos significativos
- Conseguir una ayuda rápida y efectiva para los usuarios

3.2.3 Soluciones

Con la Versión 1 de la herramienta hay unos problemas identificados y tras analizarlos detenidamente y haber hecho pruebas con usuarios para encontrar posibles soluciones, se han llegado a los acuerdos de las acciones para intentar solventarlos:

Problema:

- Entrada a un entorno poco intuitivo y complejo. Necesidad de explicación para entender la herramienta y empezar a programar.

Solución:

- Entrada a una parte de la herramienta más personal. Procesos e interfaz simplificados, con las acciones importantes y necesarias más visibles.

Problema:

- El usuario no consigue comprender ni relacionar el hardware físico con el que el figurativo que aparece en la herramienta.

Solución:

- Creación de dos partes diferenciadas en la herramienta, una primera parte Hardware con la que identificar tu esquema hardware físico y crearlo virtualmente en la herramienta y una segunda parte de Software para añadir la programación mediante bloques visuales.

Problema:

- La Chrome App tiene muchos errores de conexión y es complicado programar una placa.

Solución:

- Creación de una aplicación Web2Board para compilar, programar y comunicar la herramienta con la placa conectada al ordenador. Una única instalación y sin dependencias de terceros.

Problema:

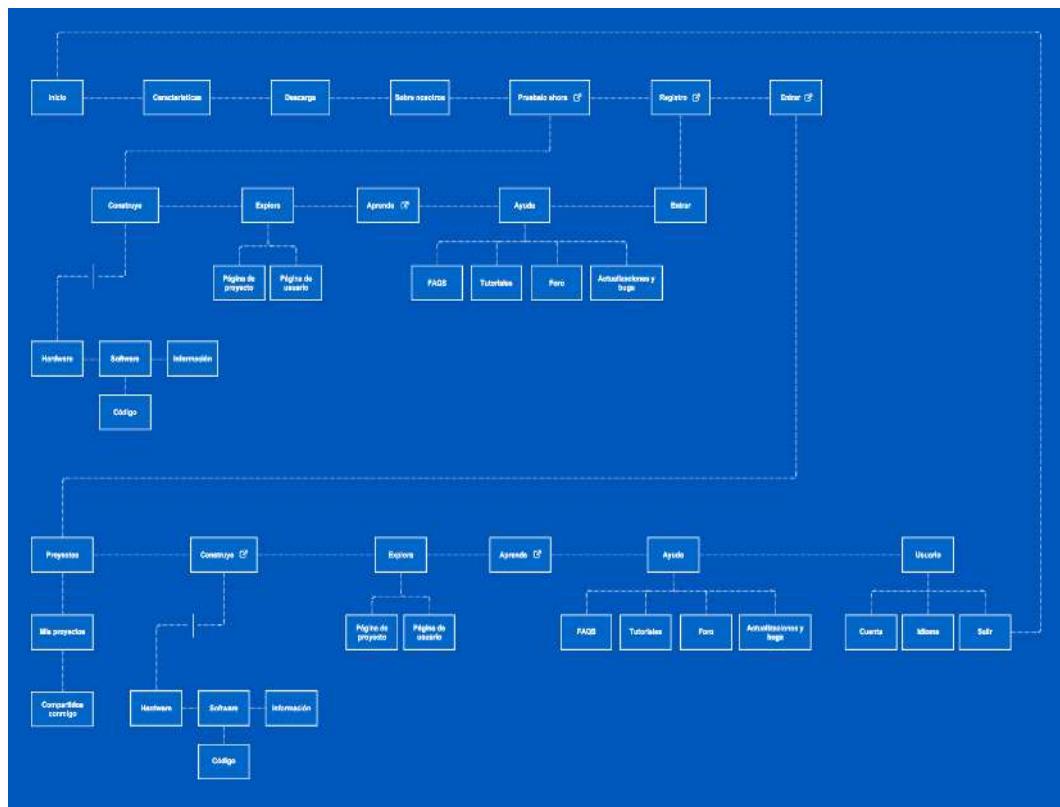
- La comunidad de proyectos no destaca por la calidad de éstos. Es una sección monótona y con muchos proyectos duplicados y vacíos.

Solución:

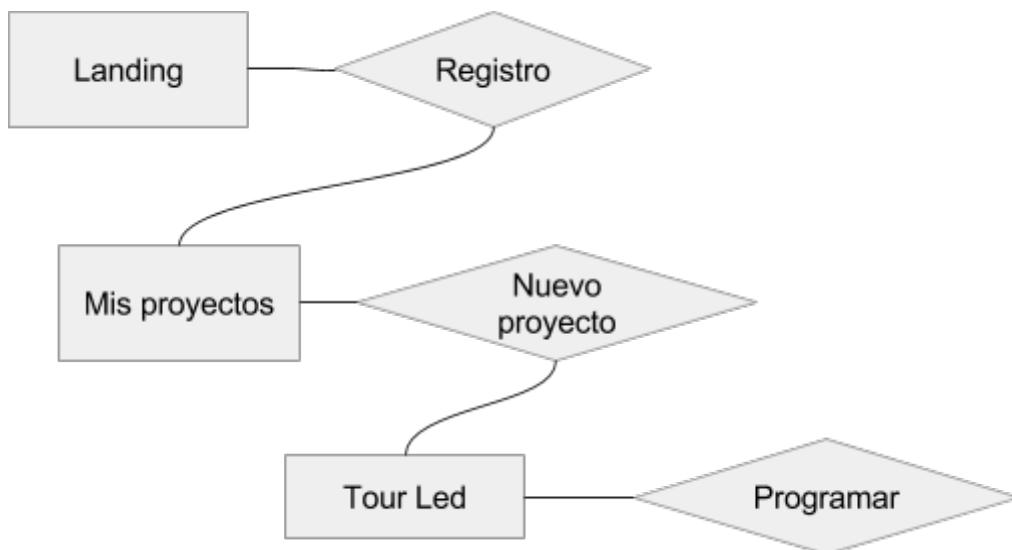
- Potenciar visualmente la comunidad. Dar mayor proyección a la ficha de proyecto. Crear un filtro en la herramienta para que haya proyectos publicados con programación.

3.2.4 Diagramas de flujo de alto nivel

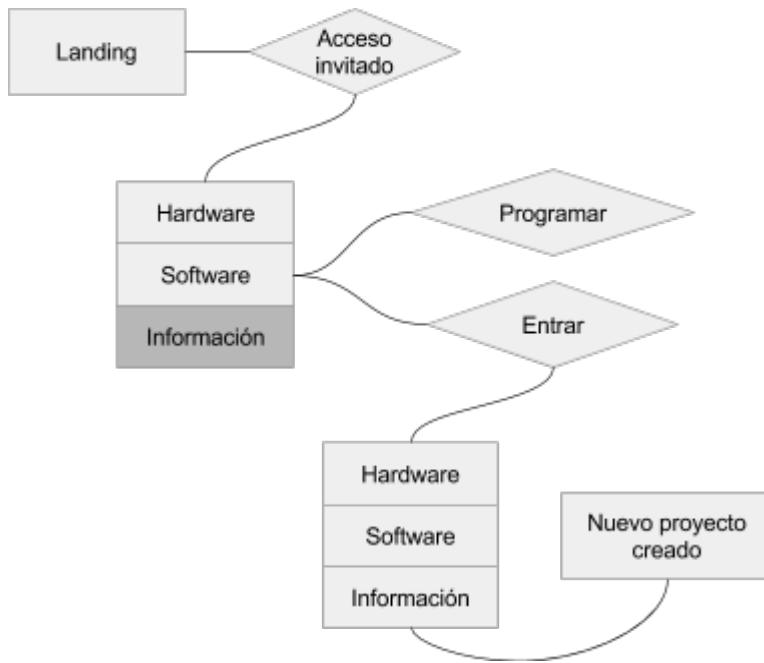
Mapa completo de la herramienta, con landing page, versión, usuario invitado y usuario logado.



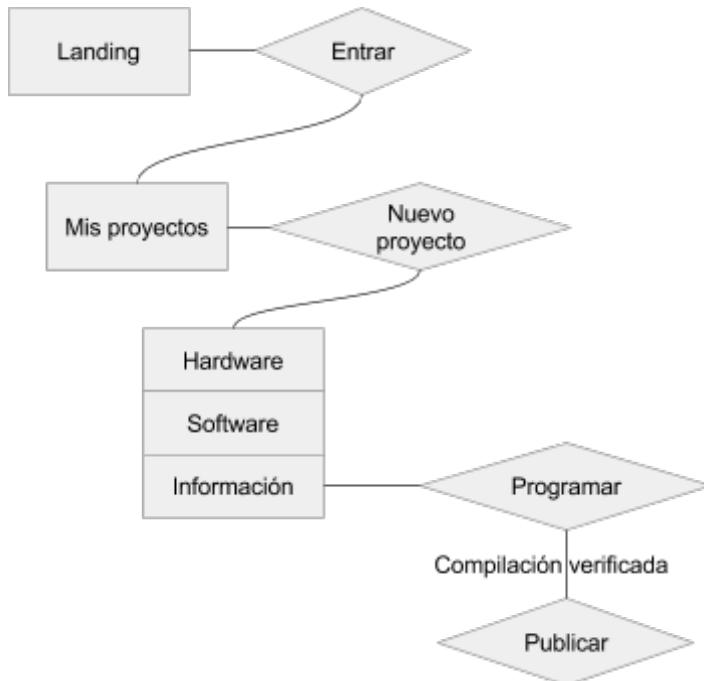
Flujo del primer acceso de un usuario a la herramienta.



Flujo de un usuario que accede de forma invitado y se registra en la herramienta.



Flujo general de la creación de un proyecto desde el inicio con un usuario que no ha iniciado sesión.



3.2.5 Wireframes - Anexo

Los wireframes diseñados para la segunda versión de Bitbloq son adjuntos en el Anexo V llamado Wireframes bitbloq V1.

3.2.6 Nueva librería de bloques “Bloqs”

A principios de Marzo de 2015 se empieza a analizar el coste de desarrollar una nueva librería de bloques visuales, para evitar todos los problemas de la actual Blockly (poca configuración de colores, formas, poco control de las features que se introducían, falta de documentación del código, preprocesado con python que causaría problemas a la hora de hacer herramientas online para que cualquier usuario pudiese hacer sus bloques desde una herramienta web).

El 6 de Marzo de 2015 se comienza el desarrollo de la librería:

<https://github.com/bq/bloqs>.

Esta librería se desarrolla para facilitar la creación de bloques por cualquier persona y está dividida en 4 secciones:

- Definición de bloques: Todos y cada uno de los bloques definidos.
- Bloqs: Generación de bloques en html/css a partir de la definición, interacción entre ellos, generación de código arduino.
- BloqsUtils: Funciones de utilidad encapsuladas y usadas en bloqs.
- BloqsLanguages: Textos de los bloques por idioma.

En la primera prueba de concepto se empieza a desarrollar utilizando svg para la construcción de bloques, sin embargo nos damos cuenta que la construcción de los bloques nos limita bastante, a la hora de integrarlo por el resto del interfaz web, por lo que se decide empezar a utilizar para la construcción de bloques exclusivamente HTML5 CSS3 y JS.

Se empieza la definición de los bloques, y su estructura en el formato JSON.

3.2.7 Bloques que modelan elementos de lógica y control del flujo del programa.

Se especifican 3 tipos de bloques básicos y sus conexiones:

Bloques básicos Statement

Este es el bloque para sentencias, su estructura consta de dos conectores, uno en la parte superior que acepta solo conectores de tipo inferior, y otro conector en la parte inferior que solo acepta elementos de tipo superior:

```
bloq = {  
  
    type: 'statement',  
    name: 'statement',  
    connectors: [{  
        type: 'connector--top',  
        accept: 'connector--bottom'  
  
    }, {  
        type: 'connector--bottom',  
        accept: 'connector--top'  
  
    }]  
};
```



Output

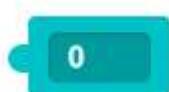
Este bloque es aquel cuya sentencia devuelve un valor. Tiene un solo conector de tipo output, y acepta únicamente conectores de tipo input.

```
bloq = {
```

```
type: 'output',
name: 'output',
connectors: [
  type: 'connector--output',
  accept: 'connector--input'
]
};
```

Como particularidad este bloque tiene un campo returnType, que indica el tipo de dato que devuelve, depende de cada bloque. , como por ejemplo el bloque simple:

simple:



Statement-Input

Estos son los conectores que al igual que los de tipo statement, tienen conectores top y bottom, pero además tiene un conector de tipo root, que solo permite elementos de tipo top.

```
var bloq = {

  type: 'statement-input',
  name: 'statement-input',
  connectors: [
    type: 'connector--top',
    accept: 'connector--bottom'
  ], {
    type: 'connector--bottom',
    accept: 'connector--top'
  }, {
    type: 'connector--root',
    accept: 'connector--top'
  ]
}
```

};



Componentes de los bloques Los bloques aparte de los conectores tienen un campo “content” que permite especificar el contenido de los bloques, se rellena un array con cada contenido con el que luego se generará el bloque.

Dependiendo del contenido tendrá unos campos u otros. Siempre tienen un campo “alias” que contiene el tipo de elemento, y un “id” que nos permite identificar inequívocamente el elemento en caso de tener dos del mismo tipo.

A continuación se describen estos componentes:

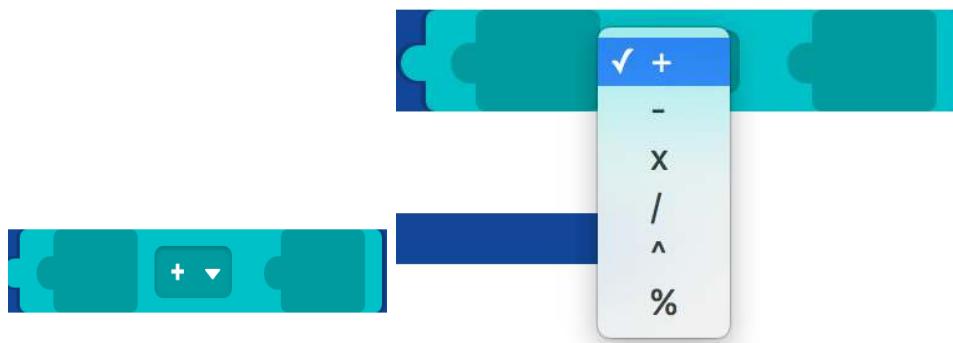
- staticDropdown: Este elemento es un desplegable de opciones fijas, por ejemplo son las notas musicales del bloque del zumbador o las operaciones que puede realizar el bloque de operaciones matemáticas. Tiene como atributo extra el campo options, que es un array de objetos, que indican el valor de la selección del desplegable y luego un label que será la representación de ese valor en el desplegable.

```
content: [
  [
    {
      id: 'OPERATOR',
      alias: 'staticDropdown',
      options: [
        {
          label: '+',
          value: '+'
        }
      ]
    }
]
```

```

        label: '-',
        value: '-'
    }, {
        label: 'x',
        value: '*'
    }, {
        label: '/',
        value: '/'
    }, {
        label: '^',
        value: '^'
    }, {
        label: '%',
        value: '%'
    }
}]
],

```



- **dynamicDropdown** : Este elemento es un desplegable de opciones cuyo contenido proviene de otros elementos externos al bloque, como por ejemplo las variables que genera otro bloque. Se utiliza poniendo un content con alias “dynamicDropdown” y en options indicando alguno de los elementos dinámicos, en el caso de las funciones, lo rellenaremos con las voidFunctions (el listado de funciones que no devuelven ningún valor).

```
"content": [
  [
    {
      "id": "FUNCTION",
      "alias": "dynamicDropdown",
      "options": "voidFunctions"
    }
  ]
],
```

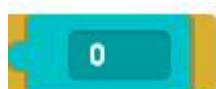


- text: Este elemento genera un campo de texto y lo rellena. Se usa el campo value para ponerle el texto o una clave que luego será traducida.

```
"content": [
  [
    {
      "alias": "text",
      "value": "bloq-invoke-function-exec"
    },
    [
    ]
],
```

- **numberInput** : Este es un campo de texto que admite números.

```
"content": [{"  
    "id": "VALUE",  
    "alias": "numberInput",  
    "value": 0  
}]]
```



- **stringInput**: Este es un campo de texto que admite todo tipo de texto.
También admite un campo de texto para añadirle un placeholder.

```
content: [  
    [ {  
        "id": "TEXT",  
        "alias": "stringInput",  
        "placeholder": "bloq-string-string"  
    }]  
,  
    "
```

- **charInput**: Igual que el bloque de texto, pero solo admite un carácter.

```
content: [  
    [ {  
        "id": "TEXT",  
        "alias": "charInput",  
        "placeholder": "bloq-char"  
    }]  
,
```

- **multilineCodeInput**: Este es un textarea que permite la entrada de código.

```
"content": [
  [
    {
      "id": "CODE",
      "alias": "multilineCodeInput",
      "value": "",
      "placeholder": "bloq-code-writeYourCode"
    }
]
```

```
if (Led == 1) {
  servo_1.write(90);
  digitalWrite(led_0, HIGH);
  digitalWrite(led_1, LOW);
  Led = 2;
} else if (Led == 2) {
  servo_1.write(0);
  digitalWrite(led_0, LOW);
  digitalWrite(led_1, HIGH);
  Led = 1;
}
```

- multilineCommentInput: Este bloque permite la entrada de texto.

```
content: [
  [
    {
      id: 'COMMENT',
      alias: 'multilineCommentInput',
      placeholder:'bloq-comment-default'
    }
  ],
]
```

Comentario // Global variables and function definition

- varInput : Campo de texto para variables.



- bloqInput: este es un hueco para un bloque, permite establecer que hay se puede poner un bloque y que tipo de bloque se permite introducir.

```
"content": [ {  
    "bloqInputId": "ARG2",  
    "alias": "bloqInput",  
    "acceptType": "float"  
},  
],  
= 
```

- headerText y descriptionText: Estos son campos especiales de bloques de tipo “group”, que son los bloques principales en los que dividimos un programa de arduino.

```
var loopBloq = __.merge(__.clone(GroupBloq, true), {  
  
    name: 'loopBloq',  
    bloqClass: 'bloq-loop',  
    headerText: 'bloq-loop-header',  
    descriptionText: 'bloq-loop-description',  
    content: [],  
    code: 'void loop(){STATEMENTS}'  
  
});
```

— Bucle principal (Loop)

Crea el programa que se va a ejecutar continuamente después del Setup.

Arrastra un bloque aquí para empezar tu programa

Código generado El código se genera desde los bloques, recorriendo el listado de bloques y usando su campo “code”, se sustituyen los ids entre llaves “{{ARG2}}” por el valor correspondiente al contenido dentro del bloque, y luego se ejecuta la función de code, dando como resultado el código generado por ese bloque.

Bloques avanzados Para facilitar la usabilidad se decidió hacer bloques sencillos y luego otros avanzados. La idea de los bloques sencillos es la de permitir programar de forma más sencilla las operaciones más habituales, evitando tener que sacar muchos bloques continuamente. Sin embargo para no restarle potencia a los programadores más exigentes, decidimos hacer versiones más avanzadas de muchos bloques que le permiten modificar mucho más el código.

Como ejemplo el bloque For, este bloque de forma sencilla te permite repetir sentencias indicándole un número inicial, otro final y el incremento modificando únicamente los campos numéricos. Mientras que el bloque de For-Advanced te permite introducir bloques que pueden contener números o variables y hacer con ellos la sentencia de for:

For:



For-Advanced:



Bloques Vars, Loop, Setup Estos 3 bloques son los 3 grupos en los que dividimos un programa de arduino.

- **Vars:** Define los valores que vas a utilizar en Setup y Loop, también puedes hacer funciones para ejecutarla en alguno de los otros grupos.
- **Setup:** En este bloque pones los bloques que quieras que se ejecute una única vez al inicio del programa.
- **Loop:** En este bloque colocas aquellos bloques que quieras que se ejecuten continuamente.

3.2.8 Librerías Arduino - BitbloqLibs

Una librería es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca. Para que el código no sea muy complejo y se tengan ficheros de 1000 líneas, en Bitbloq sacamos librerías con muchos de nuestros componentes (ultrasonidos, ledrgb, rtc,...).

Las librerías de Bitbloq están en: <https://github.com/bq/bitbloqLibs>, lo que hace el programa de compilación es descargarse las librerías que le solicita Bitbloq a la hora de compilar (si no las tiene ya descargadas) y compila con ellas.

Las librerías se almacenan siempre en la carpeta del usuario, en una subcarpeta llamada Arduino y luego en otra subcarpeta llamada libraries. Es la estructura de carpetas que utiliza el propio IDE de Arduino, la idea era sobre todo, que el código que exportas de Bitbloq luego te funcione en el IDE de Arduino sin necesidad de migrar librerías.

3.2.9 Nueva aplicación Web2Board para compilación

Para evitar todos los problemas que tenían los usuarios con las conexiones a internet, decidimos pasar de un modelo, en el que la compilación se hacía en servidor y la carga a la placa a través de una aplicación de navegador (usando una chromeapp), a un modelo de aplicación nativa que se comunica con la web de forma local por webSockets, que nos permitiría a la larga desarrollar plugins para sacarle mayor partido.

La aplicación está como opensource en <https://github.com/bq/web2board>, esta desarrollada en python, y funciona levantando un servidor que escucha peticiones en un socket, y acepta comandos predefinidos desde cualquier web, de esta forma aumentamos las posibilidades de la web a las opciones que nos da realizar una aplicación nativa. Los comandos actuales son:

- version
- setBoard
- setPort
- compile
- setBitbloqLibsVersion
- getBitbloqLibsVersion
- upload
- SerialMonitor
- exit

Decidimos utilizar python por la gran cantidad de librerías ya desarrolladas para compilar y cargar a las placas y por ser multiplataforma.

Para el empaquetado de cada sistema operativo usamos:

- Windows: Utilizamos NSIS http://nsis.sourceforge.net/Main_Page

- Darwin: Para generar un dmg, utilizamos py2app, y luego para empaquetarlo y hacer un instalador que instale también los instaladores de los drivers de las placas de ZUM y ZOWI utilizamos la aplicación Packages de Stéphane Sudre (<http://s.sudre.free.fr/Software/Packages/about.html>)
- Ubuntu: Utilizamos el comando dpkg-buildpackage para construir el paquete de debian

Alpha en el entorno de MVP Para disponer de un feedback sobre el funcionamiento y usabilidad de bitbloq V2 se decide la salida de una versión alpha a un público reducido de la empresa.

El público es elegido concienzudamente cumpliendo las siguientes características:

- Son profesores o usuarios conocedores del funcionamiento de la versión anterior.
- Disponen de conocimientos en el área de la enseñanza, en concreto en programación a niños.
- Son usuarios interesados en el producto que desde un principio han tenido una vinculación con el proyecto.

A todo el público anteriormente mencionado se le indica que nos envíe todas sus incidencias, dudas y sugerencias a mejorar. Además de ello, el equipo de UX/UI establece una comunicación más directa para mejorar la interfaz y usabilidad lo máximo posible.

Beta en el entorno de BETA Una vez que el público seleccionado de la versión Alpha nos ha expresado su feedback, todas las mejoras posibles de incorporar se implementan, las incidencias detectadas son resueltas y además de ello se implementan nuevas funcionalidades.

Con todo ello se realiza una nueva versión BETA la cual ya es pública para todos los usuarios que deseen acceder. Al igual que la versión alpha, el grupo de usuarios que fue seleccionado sigue realizando feedback sobre su usabilidad.

Con al versión BETA se producen demostraciones con un número determinado de usuarios, estos no son seleccionados, cualquier empleado puede asistir a la demostración y posteriormente interactuar con la aplicación, aún sin seleccionar a los usuarios se pueden encontrar los siguientes perfiles:

- Usuarios con conocimientos de programación
- Usuarios sin conocimientos de programación
- Usuarios relacionados con la enseñanza.
- Los propios desarrolladores.

Una vez se realiza la demostración y los usuarios comienzan a interactuar con la aplicación, los responsables de UX realizan mediante su observación una investigación contextual con la cual llegan a diferentes conclusiones según el tipo de usuario, las conclusiones son anotadas y analizadas para mejorar la usabilidad en las partes en la que los usuarios han tenido más inconvenientes, a parte los propios usuarios van sugiriendo mejoras que son anotadas para siguientes versiones.

3.2.10 Plan de migración de usuarios de BitbloqV1 a BitbloqV2.0

Una vez que la aplicación ha pasado por las fases de alpha y Beta ya está lista para salir a producción con la url principal de bitbloq.

Antes de que salga a producción, en la url principal de bitbloq se encuentra la versión 1 de producción, por ello es necesario realizar las siguientes modificaciones para que los usuarios que decidan seguir utilizando la versión antigua les sea posible:

- Desplegar bitbloq1 en una nueva url

- Indicar en la versión de bitbloq2 de forma clara mediante una modal en la pantalla la dirección de bitbloq1 para quien decida utilizarlo.

Una vez que bitbloq1 está alojado en una nueva dirección se realiza el despliegue de bitbloq2 en la url principal, de esta forma durante un tiempo conviven ambas versiones para que los usuarios traspasen sus proyectos y comiencen a utilizar la nueva versión.

3.2.11 v2.0 en Producción

Una vez que se ha realizado el plan de migración de usuarios de bitbloq1 a bitbloq2, bitbloq ya se encuentra en producción, a partir de entonces ya está plenamente operativo.

3.2.12 Infraestructura para test unitarios y end-to-end.

Un software siempre tiene algún fallo o bug, el objetivo del proceso de calidad es reducir el número de fallos al mínimo, con lo cual es necesario elegir una infraestructura adecuada y esta elección, puede influir en muchas cosas, por ejemplo, los tiempos de ejecución de pruebas.

El proceso de calidad empieza por la definición de test sobre los requisitos que van surgiendo. En este proyecto se utiliza la herramienta TestLink, que permite el almacenamiento y la gestión de los test.

Cuando se tienen los test definidos, el siguiente paso es automatizar el mayor número de test, teniendo en cuenta que algunos es inviable poder automatizarlos, ya que es necesaria la intervención humana. Para la automatización se ha utilizado javascript y un framework llamado protractor basado en selenium.

Se busca que todo el código referente a la automatización de pruebas se vaya integrando continuamente, debido a que el código debe mantenerse estable para su ejecución en cualquier momento. Para llevar a cabo esto, se utilizará la herramienta Gerrit, realizará funciones de intermediario entre el repositorio y el desarrollador.

Cuando una versión del software esté lista para subir a producción, será necesario certificar dicha versión, es decir ver lo estable y la calidad que tiene. En este punto se vuelve a utilizar la herramienta TestLink, que permite también una gestión de un historial de qué pruebas han pasado, cuales no han pasado y en qué plataformas se han realizado las pruebas (p.e. Windows, MacOS). En la herramienta esta funcionalidad recibe el nombre de Build. Si se detecta algún fallo crítico, se intentará arreglar y volver a pasar la certificación.

En los siguientes puntos se explicará el framework que se eligió para desarrollar las pruebas, la integración con Gerrit y la organización de las test, test suites y builds.

Prueba de concepto nightwatch vs protractor En este punto nos encontramos con la elección de qué framework utilizar para desarrollar nuestras pruebas. Esta elección puede ser decisiva, ya que influirá en la rapidez de desarrollo de las pruebas y en los tiempos de ejecución. Para entender mejor los dos framework, primero se describirán brevemente, después se aportaran las diferentes ventajas que nos aporta cada uno y por último la decisión tomada.

Nightwatch.js es un framework que se utiliza para automatizar pruebas que certificarán la calidad de aplicaciones orientadas a la Web. Está escrito en node.js y utiliza la API de selenium WebDriver.

Protractor es un framework que se utiliza para automatizar pruebas que certificarán la calidad de aplicaciones orientadas a la Web. Estas aplicaciones deben estar escritas en AngularJS, como es el caso de Bitbloq.

VENTAJAS DE UTILIZAR PROTRACTOR EN VEZ DE NIGHTWATCH

- En primer lugar, se busca por la Web cuánta información hay sobre los dos frameworks que pueda ser útil para el desarrollo de las pruebas. Con este fin se ha recurrido a Stackoverflow, una página donde se pueden realizar consultas sobre diferentes temas técnicos y otros usuarios pueden responder. Se ha buscado, a través de su filtro, las palabras clave Nightwatch, Protractor y Nightwatch angular (ya que no es un framework basado en AngularJS y hay que buscar compatibilidad con él). El resultado ha sido el siguiente:

- La búsqueda por la palabra clave Protractor tiene 7471 resultados.
- La búsqueda por la palabra clave Nightwatch tiene 196 resultados.
- La búsqueda por las palabras clave Nightwatch y angular solo tiene 1 resultado.

En base a estos resultados parece que se puede encontrar más información de Protractor que de Nightwatch. Además el tercer resultado aporta que poca gente ha intentado compatibilizar Angular y Nightwatch.js.

- Otro punto a comparar son las APIs de los dos frameworks. Si se entra en sus respectivas páginas (Protractor <https://angular.github.io/protractor/#/api> y Nightwatch <http://nightwatchjs.org/api>) entonces se observa que la API de Protractor es más extensa y esta mejor documentada.
- Se puede integrar con otros frameworks como Mocha (<https://mochajs.org/>), que permite haces pruebas inclusivas o exclusivas, y además ya viene configurado selenium webdriver, sin tener que descargarlo e integrarlo con el proyecto de pruebas.
- Como se ha dicho en la descripción de Protractor, es un framework orientado a aplicaciones escritas en AngularJS, por lo tanto protractor está preparado para que no haya timeouts cuando se realizan los test utilizando elementos internos como \$http, \$scope, \$timeout,... Esto agiliza las pruebas dado que la mayoría de las veces cuando fallan es por timeouts provocados por el navegador.
- Por último, se ha desarrollado la misma prueba para los dos frameworks y de ese modo poder valorar la eficiencia de cada uno. En este caso, Protractor ha sido dos segundos más rápido, siendo esto determinante en el caso de que se ejecuten una pila grande de test.

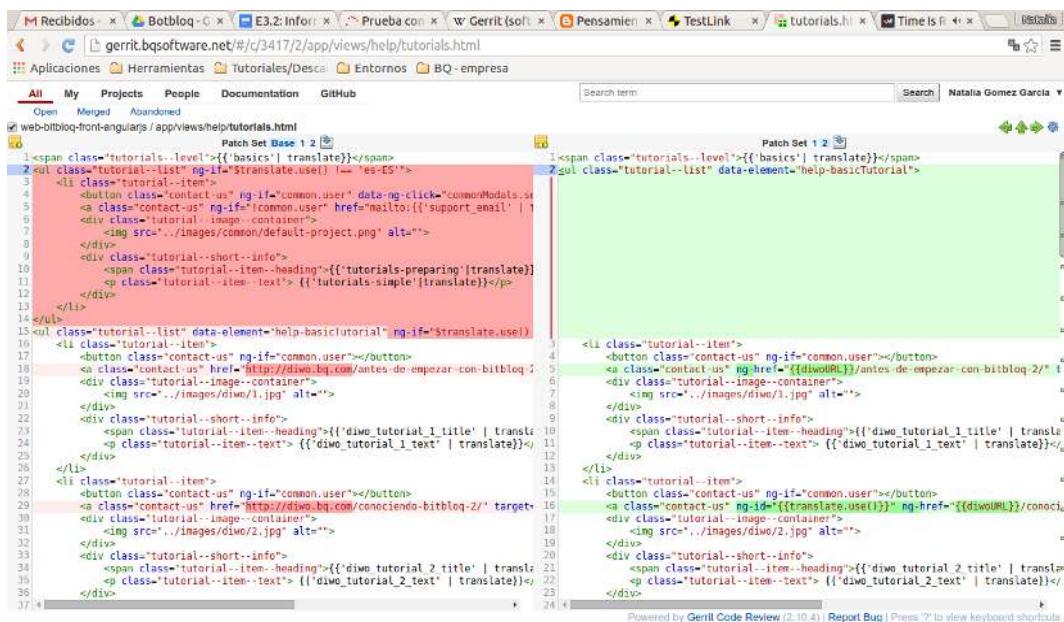
VENTAJAS DE UTILIZAR NIGHTWATCH EN VEZ DE PROTRACTOR.

- La principal ventaja de utilizar Nightwatch es que posee una sintaxis sencilla y clara y eso puede aportar más rapidez a la hora de escribir la prueba.
- Puede integrarse con sistemas Hudson y TeamCity.

Después de explicar los diferentes ventajas de cada framework, se llega a la conclusión de que las ventajas que aporta Protractor son más beneficiosas frente a las que aporta Nightwatch.js, sobre todo porque está integrado con Angular.

Integración con gerrit Gerrit es una sistema de revisión de código para sistemas que utilicen un control de versiones Git. Lo interesante de esta herramienta es que Gerrit actúa como un intermediario entre el desarrollador y el repositorio. A continuación se explicará el proceso de utilización de la herramienta y las ventajas que aporta.

Cuando un desarrollador sube sus cambios, estos quedan almacenados en Gerrit para que sean revisados por otro desarrollador. En primer lugar Gerrit pasará diferentes pruebas y revisiones a los cambios subidos para comprobar que dicho código no estropea lo que hay en ese momento en el repositorio subido. Este proceso de pruebas se realiza gracias a que se puede integrar jenkins en gerrit. Jenkins lanza una tarea de jshint que verifica el código subido. En el caso de que la tarea termine correctamente, Gerrit aprobará la fusión del código y será el turno de un desarrollador del equipo, que revisará el commit. Con el visualizador de gerrit, se puede ver el código añadido en color verde y el código suprimido en color rojo (se muestra en la imagen inferior), y así facilita la visualización de los cambios al desarrollador que esté revisando el código.



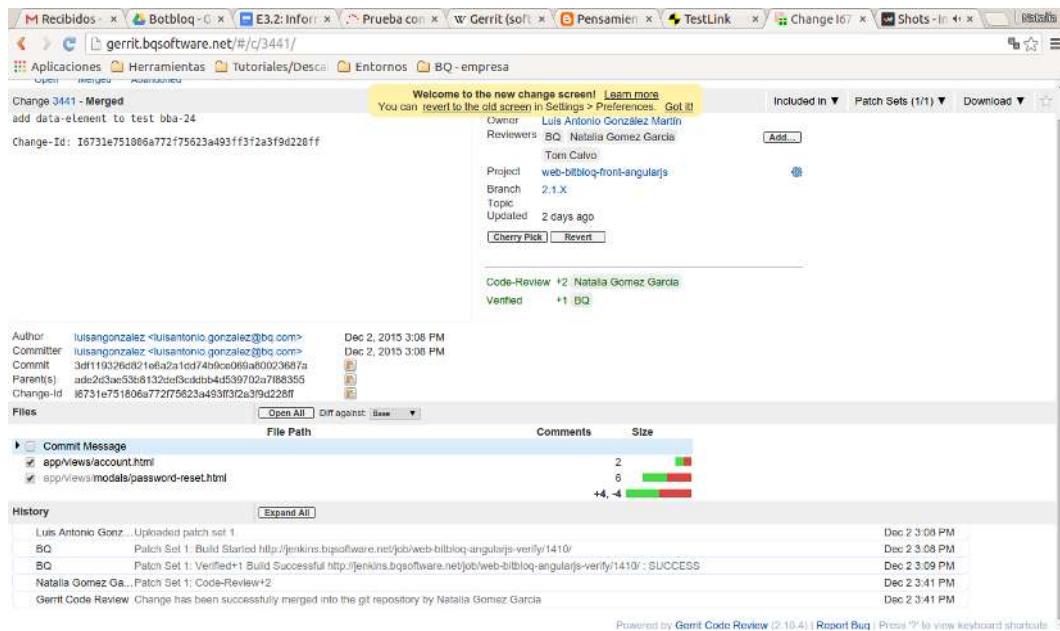
```

Patch Set 1 2
1<span class="tutorials--level">>{{'basics'| translate}}</span>
2<ul class="tutorial-list" ng-if="translate.use() || 'es-ES'>
3  <li class="tutorial-item">
4    <button class="contact-us" ng-if="common.user" data-ng-click="commonModals.show('support_email')">
5      <a class="contact-us" ng-if="!common.user" href="mailto:{{ support_email }}">
6        <div class="tutorial-image-container">
7          
8        </div>
9      <div class="tutorial-short-info">
10        <span class="tutorial-item-heading">{{'tutorials-preparing'|translate}}
11        <p class="tutorial-item-text">{{'tutorials-simple'|translate}}</p>
12      </div>
13    </li>
14  </ul>
15<ul class="tutorial-list" data-element="help-basic/tutorial" ng-if="!translate.use()">
16  <li class="tutorial-item">
17    <button class="contact-us" ng-if="common.user"></button>
18    <a class="contact-us" href="http://diwo.bq.com/antes-de-empezar-con-bitbloq-1/" target="_blank">
19      <div class="tutorial-image-container">
20        <img src='../images/diwo/1.jpg' alt="">
21      </div>
22      <div class="tutorial-short-info">
23        <span class="tutorial-item-heading">{{'diwo_tutorial_1_title' | translate}}
24        <p class="tutorial-item-text">{{'diwo_tutorial_1_text' | translate}}</p>
25      </div>
26    </li>
27  <li class="tutorial-item">
28    <button class="contact-us" ng-if="common.user"></button>
29    <a class="contact-us" href="http://diwo.bq.com/conociendo-bitbloq-2/" target="_blank">
30      <div class="tutorial-image-container">
31        <img src='../images/diwo/2.jpg' alt="">
32      </div>
33      <div class="tutorial-short-info">
34        <span class="tutorial-item-heading">{{'diwo_tutorial_2_title' | translate}}
35        <p class="tutorial-item-text">{{'diwo_tutorial_2_text' | translate}}</p>
36      </div>
37
Patch Set 1 2
1<span class="tutorials--level">>{{'basics'| translate}}</span>
2<ul class="tutorial-list" data-element="help-basicTutorial">
3  <li class="tutorial-item">
4    <button class="contact-us" ng-if="common.user"></button>
5    <a class="contact-us" ng-href="{{diwoURL}}/antes-de-empezar-con-bitbloq-2/" target="_blank">
6      <div class="tutorial-image-container">
7        <img src='../images/diwo/1.jpg' alt="">
8      </div>
9      <div class="tutorial-short-info">
10        <span class="tutorial-item-heading">{{'diwo_tutorial_1_title' | translate}}
11        <p class="tutorial-item-text">{{'diwo_tutorial_1_text' | translate}}</p>
12      </div>
13    </li>
14  <li class="tutorial-item">
15    <button class="contact-us" ng-if="common.user"></button>
16    <a class="contact-us" ng-id="{{translate.use()}}" ng-href="{{diwoURL}}/conociendo-bitbloq-2/" target="_blank">
17      <div class="tutorial-image-container">
18        <img src='../images/diwo/2.jpg' alt="">
19      </div>
20      <div class="tutorial-short-info">
21        <span class="tutorial-item-heading">{{'diwo_tutorial_2_title' | translate}}
22        <p class="tutorial-item-text">{{'diwo_tutorial_2_text' | translate}}</p>
23      </div>
24

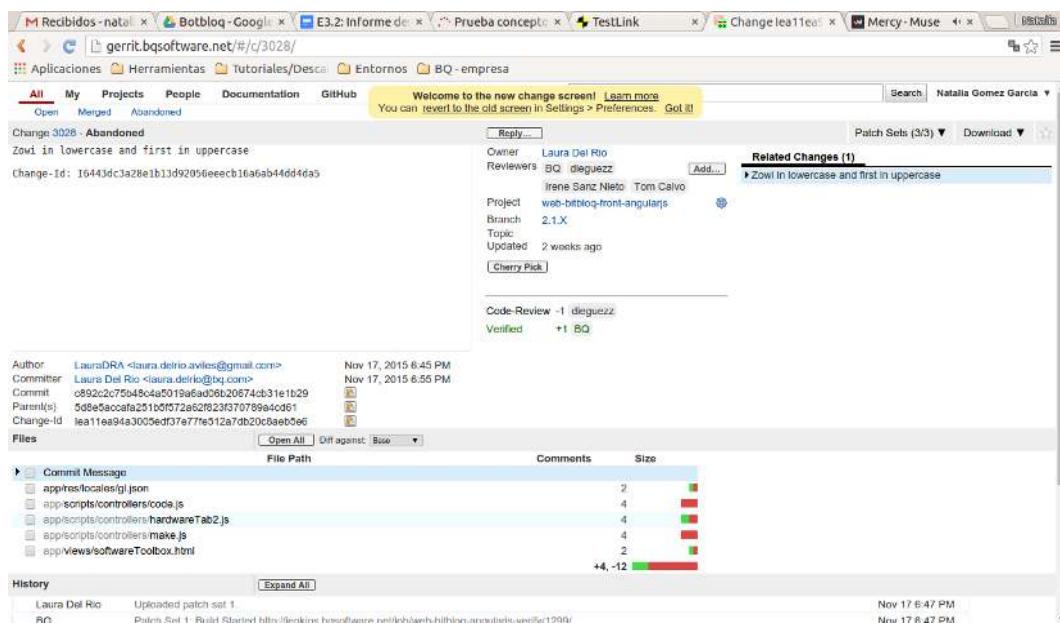
```

Powered by Gerrit Code Review (2.10.4) | Report Bug | Press '?' to view keyboard shortcuts

Después de la revisión, el desarrollador valorará si acepta o no los cambios. En caso de ser aceptados, Gerrit fusionará los cambios con la información del repositorio, en caso contrario, el desarrollador que ha subido los cambios deberá realizar las variaciones que le ha comentado el revisor y volver a pasar por el mismo proceso hasta que se acepten sus cambios o decida abandonarlos y empezar desde el principio. A continuación la imagen muestra un commit que ha sido aprobado por gerrit (+1) y por el desarrollador (+2).

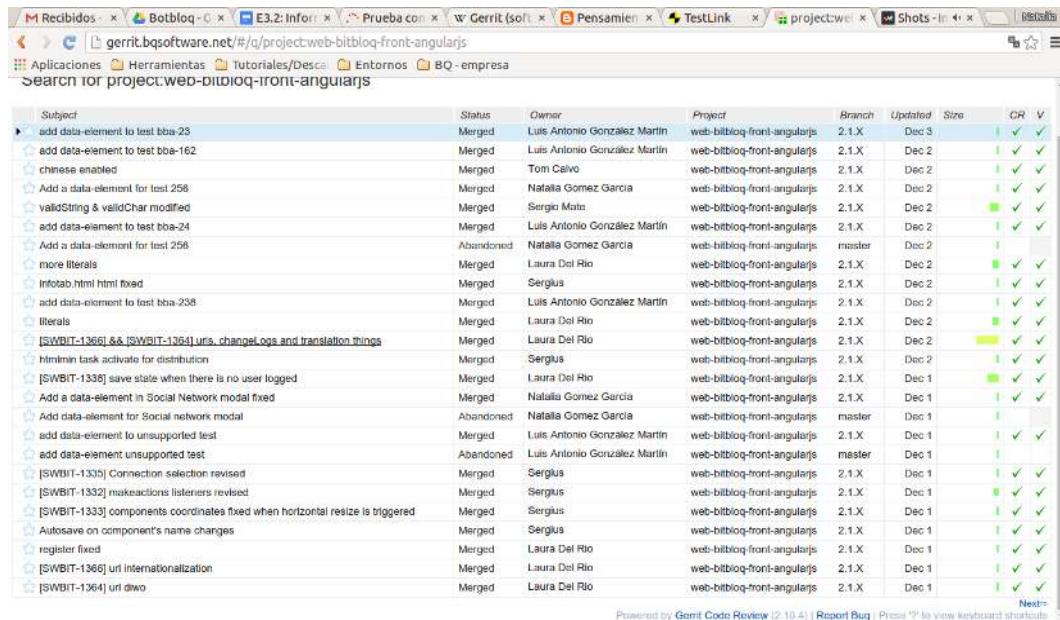


En la imagen de abajo se puede que este commit ha sido aprobado por Gerrit (+1) pero no ha sido aprobado por el desarrollador (-1).



Si en algún momento se fusiona antes un commit que otro provocando un conflicto, Gerrit informará de ese hecho y no dejará mergear el commit que provoca el conflicto. Otro punto a favor de esta herramienta es que guarda una lista de los

commit que se van fusionando a lo largo de la vida del software permitiendo ver qué cambios ha hecho quién y en qué momento se hicieron (en la imagen de abajo aparece un listado de parte de los commits que se han hecho en un día).



The screenshot shows a Gerrit code review interface with a list of commits. The commits are listed in a table with columns: Subject, Status, Owner, Project, Branch, Updated, Size, CR, and V. The commits are mostly merged and show various changes like adding data-elements, fixing bugs, and updating AngularJS components. The interface includes navigation buttons like 'Next >' and 'Powered by Gerrit Code Review (2.10.4) | Report Bug | Press '?' to view keyboard shortcuts'.

Subject	Status	Owner	Project	Branch	Updated	Size	CR	V
add data-element to test bba-23	Merged	Luis Antonio González Martín	web-bitbloq-front-angularjs	2.1.X	Dec 3		✓	✓
add data-element to test bba-162	Merged	Luis Antonio González Martín	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
chinese enabled	Merged	Tom Calvo	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
Add a data-element for test 258	Merged	Natalia Gomez Garcia	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
validString & validChar modified	Merged	Sergio Mata	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
add data-element to test bba-24	Merged	Luis Antonio González Martín	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
Add a data-element for test 258	Abandoned	Natalia Gomez Garcia	web-bitbloq-front-angularjs	master	Dec 2			
more literals	Merged	Laura Del Rio	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
infotab.html html fixed	Merged	Sergius	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
add data-element to test bba-238	Merged	Luis Antonio González Martín	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
literals	Merged	Laura Del Rio	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
[SWBIT-1360] && [SWBIT-1364] urls, change logs and translation things	Merged	Laura Del Rio	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
htmlmin task activate for distribution	Merged	Sergius	web-bitbloq-front-angularjs	2.1.X	Dec 2		✓	✓
[SWBIT-1338] save state when there is no user logged	Merged	Laura Del Rio	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
Add a data-element in Social Network modal fixed	Merged	Natalia Gomez Garcia	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
Add data-element for Social network modal	Abandoned	Natalia Gomez Garcia	web-bitbloq-front-angularjs	master	Dec 1			
add data-element to unsupported test	Merged	Luis Antonio González Martín	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
add data-element unsupported test	Abandoned	Luis Antonio González Martín	web-bitbloq-front-angularjs	master	Dec 1			
[SWBIT-1335] Connection selection revised	Merged	Sergius	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
[SWBIT-1332] make reconnections listeners revised	Merged	Sergius	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
[SWBIT-1332] components coordinates fixed when horizontal resize is triggered	Merged	Sergius	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
Autosave on component's name changes	Merged	Sergius	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
register fixed	Merged	Laura Del Rio	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
[SWBIT-1368] url internationalization	Merged	Laura Del Rio	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓
[SWBIT-1364] url dwto	Merged	Laura Del Rio	web-bitbloq-front-angularjs	2.1.X	Dec 1		✓	✓

Los beneficios que nos aporta Gerrit son los siguientes:

- *Localización temprana de errores.* Como el código debe ser aprobado por Gerrit y por otro desarrollador que revise los cambios, se pueden localizar los problemas antes de fusionar el código y/o falle la aplicación. Además gracias al historial de Gerrit, se podrá localizar un commit que provoque el mal funcionamiento de la aplicación y volver a una versión estable.
- *Intercambio de conocimiento.* El proceso de revisión de código permite a los recién llegados ver el código de otros desarrolladores más experimentados. También permite que el revisor del commit deje comentarios en el código, aportando su visión y posibles mejoras.
- *Código limpio y fácil de mantener.* Gracias a las revisiones se podrá mantener unos estándares para que los sigan los componentes del equipo.



Test suites

La organización de las test suites se ha realizado mediante páginas y secciones. Cada página y cada sección tiene su propia test suite.

Para el manejo de las test suites y los los casos de prueba correspondientes a cada una de estas test suites, se ha usado la herramienta TestLink.

Los encargados de la sección de Quality Assessment se encargan de buscar posibles casos de prueba recorriendo los caminos funcionales de la herramienta Bitbloq y los añaden a la herramienta Testlink en la suite de pruebas que le corresponda.

A día de hoy hay 245 casos de prueba presentes para la herramienta Bitbloq. De estos, 129 son manuales y 116 automatizados.

Las test suites que se han creado son:

- Web2Board: Contiene los casos de prueba relacionados con la instalación y actualización de la herramienta Web2Board, utilizada para transferir los proyectos creados de Bitbloq a las placas o robots.

- BitbloqsLibs: Se encuentra incluida en la test suite de Web2Board. Contiene los casos de pruebas relacionados con la descarga, actualización y borrado de las librerías de Bitbloq.
- Registro: Contiene los casos de prueba relacionados con el registro de usuario, tanto usuarios correctos como incorrectos y por correo electrónico o por redes sociales.
- Social: Contiene los casos de prueba relacionados con la asociación de una cuenta de Bitbloq a una red social.
- Global: Contiene los casos de prueba asociados a los posibles problemas del equipo en el que se ejecuta Bitbloq, como puede ser la hora incorrecta.
- Compile: Contiene los casos de prueba relacionados con la compilación de un proyecto de Bitbloq mediante Web2Board
- To board: Contiene los casos de prueba relacionados con la carga de proyectos a una placa o robot mediante Web2Board.
- Projects: Contiene los casos de prueba relacionados con la visualización y control de los proyectos de un usuario.
- My projects: Se encuentra incluida en la en la test suite de projects. Contiene los casos de prueba relacionados con la visualización de los proyectos creados por el usuario.
- Share: Se encuentra incluida en la test suite de projects. Contiene los casos de prueba relacionados con la visualización de los proyectos compartidos con el usuario.
- Explore: Contiene los casos de prueba relacionados con la visualización de los proyectos publicados por cualquier usuario.
- Filters: Se encuentra incluida en la test suite de Explore. Contiene los casos de prueba relacionados con el filtrado de proyectos mediante placas y componentes.

- Bloqsproject: Contiene los casos de prueba relacionados con la creación y el manejo de un proyecto que se va a programar usando componentes y bloques.
- Walkthrough: Se encuentra incluida en la test suite de Bloqsproject. Contiene los casos de prueba relacionados con el tutorial de como usar Bitbloq que aparece la primera vez que un usuario entra en la aplicación.
- Makeactions: Se encuentra incluida en la test suite de Bloqsproject. Contiene los casos de prueba relacionados con las acciones que se pueden realizar dentro de un proyecto.
- Show: Se encuentra incluida en la test suite de Makeactions. Contiene los casos de prueba relacionados con las acciones que se pueden realizar con la pestaña ver.
- Share: Se encuentra incluida en la test suite de Makeactions. Contiene los casos de prueba relacionados con las acciones que se pueden realizar con la pestaña compartir.
- File: Se encuentra incluida en la test suite de Makeactions. Contiene los casos de prueba relacionados con las acciones que se pueden realizar con la pestaña archivo.
- Help: Se encuentra incluida en la test suite de Makeactions. Contiene los casos de prueba relacionados con las acciones que se pueden realizar con la pestaña ayuda.
- Edit: Se encuentra incluida en la test suite de Makeactions. Contiene los casos de prueba relacionados con las acciones de deshacer y rehacer.
- Hardware: Se encuentra incluida en la test suite de Bloqsproject. Contiene los casos de prueba relacionados con añadir, eliminar, duplicar, conectar y desconectar componentes, placas y robots.
- Software: Se encuentra incluida en la test suite de Bloqsproject. Contiene los casos de prueba relacionados con el control de bloques de código.

- Infotab: Se encuentra incluida en la test suite de Bloqsproject. Contiene los casos de prueba relacionados con las acciones de edición de información de proyecto, como puede ser nombre, imagen o tags.
- Idioma: Contiene los casos de prueba relacionados con el cambio de idioma de la aplicación.
- Account: Contiene los casos de prueba relacionados con la configuración y modificación de una cuenta de usuario.
- Help: Contiene los casos de prueba relacionados con ayuda.
- FAQ: Se encuentra incluida en la test suite de Help. Contiene los casos de prueba relacionados con la ventana de preguntas comunes.
- Tutorial: Se encuentra incluida en la test suite de Help. Contiene los casos de prueba relacionados con los tutoriales.
- Changelog: Se encuentra incluida en la test suite de Help. Contiene los casos de prueba relacionados con el log de cambios.
- Tooltips: Contiene los casos de prueba relacionados con las anotaciones de ayuda que aparecen al pasar el ratón sobre los botones.
- Make: Se encuentra incluida en la test suite de Tooltips. Contiene los casos de prueba relacionados con las anotaciones de ayuda que aparecen al pasar el ratón sobre los botones.
- Make actions: Se encuentra incluida en la test suite de Make. Contiene los casos de prueba relacionados con las anotaciones de ayuda que aparecen al pasar el ratón sobre los botones.
- Project explore: Contiene los casos de prueba relacionados con el manejo de los proyectos en la pestaña de explora.
- Unsupported: Contiene los casos de prueba relacionados con el control de entornos sobre los cuales no está soportado Bitbloq

- Desktop: Se encuentra incluida en la test suite de Unsupported. Contiene los casos de prueba relacionados con el control de entornos de escritorio sobre los cuales no está soportado Bitbloq.
- Tablet: Se encuentra incluida en la test suite de Unsupported. Contiene los casos de prueba relacionados con el control de entornos de dispositivos tablet sobre los cuales no está soportado Bitbloq.
- Phone: Se encuentra incluida en la test suite de Unsupported. Contiene los casos de prueba relacionados con el control de entornos de dispositivos móvil sobre los cuales no está soportado Bitbloq.
- Header: Contiene los casos de prueba relacionados con la cabecera de la página.
- Login: Contiene los casos de prueba relacionados con el inicio de sesión de usuario.
- Modals: Contiene los casos de prueba relacionados con los popup que pueden aparecer en cualquier momento.
- Cookies bar: Contiene los casos de prueba relacionados con la barra de información sobre las cookies que aparece en la parte inferior de la pestaña.
- Codeproject: Contiene los casos de prueba relacionados con el control de los proyectos en los cuales sólo se codifica, no se ponen bloques o componentes.
- Makeactions: Se encuentra incluida en la test suite de Codeproject. Contiene los casos de prueba relacionados con las opciones que se pueden aplicar a un proyecto de código.
- Landing: Contiene los casos de prueba relacionados con la verificación de los links presentes en la página de landing.
- Bloqs: Contiene los casos de prueba relacionados con la verificación del funcionamiento de los bloques.
- Bitbloq help: Contiene los casos de prueba relacionados con los medios de ayuda disponibles para los usuarios.

```
spec started
  Menu file of MakeActions
    ✓ bba-64: delete a project

Executed 1 of 3 specs SUCCESS (2 SKIPPED) in 1 min 1 sec.
Session deleted: Going to shut down the Selenium server
Shutting down Selenium server: http://127.0.0.1:4444
Shut down Selenium server: http://127.0.0.1:4444 (OKOK)
[launcher] 0 instance(s) of WebDriver still running
[launcher] chrome43.0 #1 passed

Done, without errors.
```

```
Menu file of MakeActions
✗ bba-64: delete a project
  - Failed: expect(...).not is not a function

*****
*          Failures          *
*****

1) Menu file of MakeActions bba-64: delete a project
   - Failed: expect(...).not is not a function
```

- Autosave: Contiene los casos de prueba relacionados con la verificación del autoguardado de los proyectos.
- State: Contiene los casos de prueba relacionados con la conservación del estado de un proyecto en distintas situaciones.

Descripción de los test. Los tests, como se ha mencionado anteriormente, están organizados en test suites. Los tests son tanto automáticos o manuales. Para la automatización de los tests se diseña, para cada caso de prueba definido, un script con Javascript que, haciendo uso de la automatización de la navegación con Selenium, simula los pasos que da un usuario para la realización del caso de prueba en cuestión. Al lanzar una prueba, se verifica que un usuario, siguiendo el proceso normal que se simula el script es correcto.

En caso de que exista un error, o de que falle algo, aparece un mensaje de error indicando que no ha sido posible completar el test. Además, se genera un informe xml con información detallada de los logs del proceso para poder localizar el error.

Los casos de prueba se definen de forma continua. El equipo de Quality Assessment analiza Bitbloq recorriendo los caminos funcionales y definiendo tests para



cada uno de los posibles puntos de fallo que son capaces de encontrar. Existen definidos 245 casos de prueba, de los cuales 116 ya han sido automatizados y 129 esperan para ser automatizados o en estos momentos no es posible automatizarlos.

En este gráfico se puede ver el número de casos de prueba definidos mes a mes desde el inicio de la definición de casos de prueba en julio de 2015 hasta la actualidad en diciembre de 2015. Como se puede ver, en agosto de 2015 hubo un gran crecimiento en el número de casos de prueba que se stabilizó en los meses posteriores, dejando como resultado un crecimiento constante de los casos de prueba a lo largo de los meses posteriores.

Las pruebas automáticas, aunque a menor ritmo, también han ido creciendo en número, desde agosto de 2015 hasta la actualidad.

Certificaciones que aseguran la calidad del proyecto Como se ha mencionado anteriormente, los casos de prueba se definen en Testlink yes también con esta herramienta que se coordinan las pruebas que se describen a continuación.

Cada vez que se sube una nueva versión del código de la aplicación a Gerrit, antes se deben pasar todas las pruebas automáticas que se han definido. Además, otro miembro del equipo debe aprobar los cambios que se han realizado al código

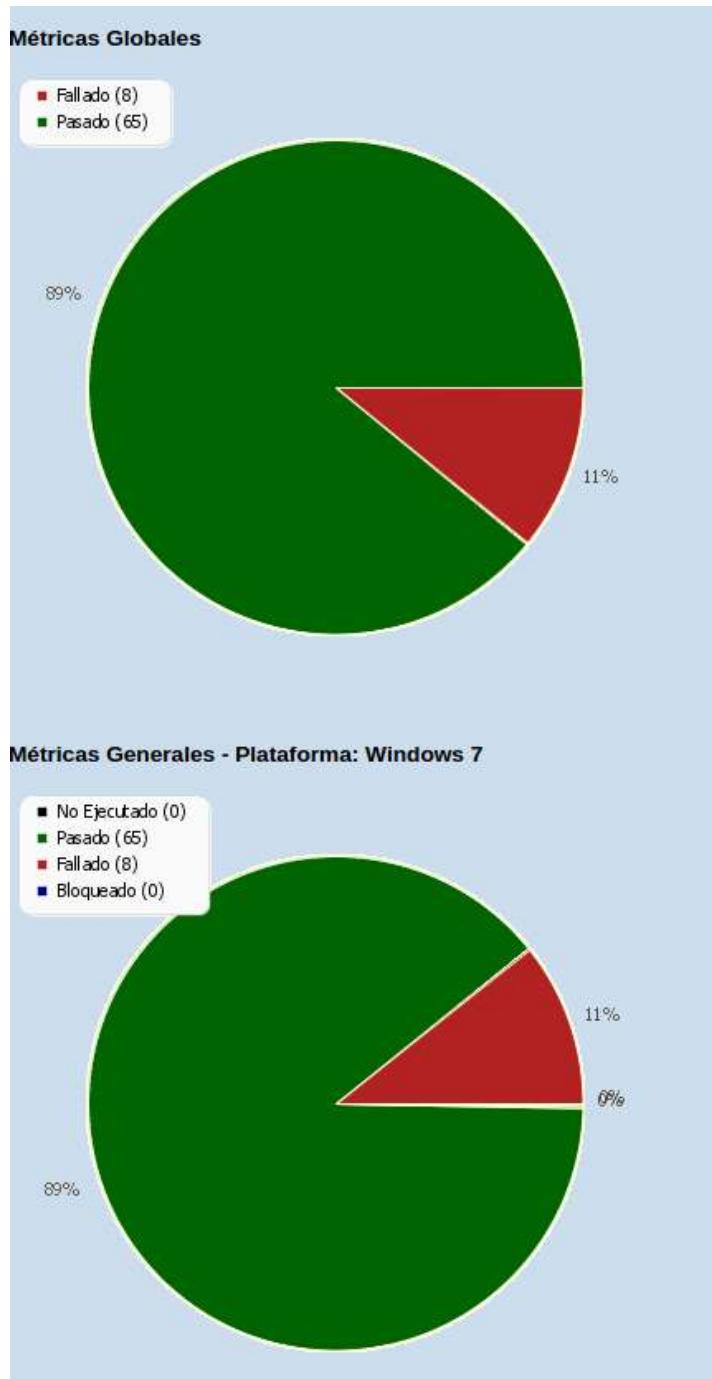
antes de que este sea mergeado con el repositorio existente. Una vez se han pasado todos estos cambios, el código está disponible para el resto del equipo. Esto ocurre tanto para el desarrollo de Bitbloq como para la automatización de pruebas.

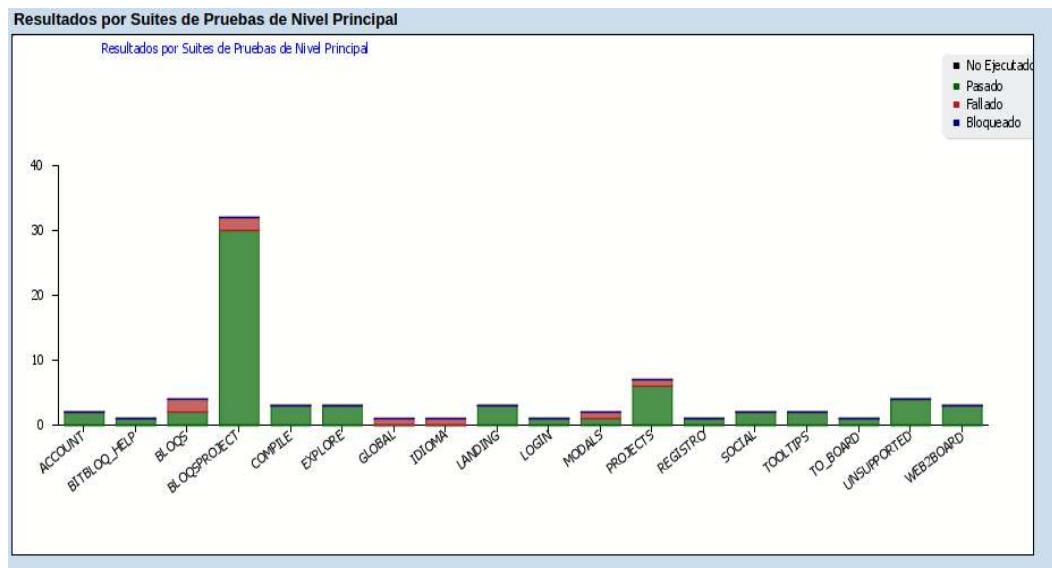
Existen cuatro entornos donde se despliega Bitbloq antes de que desplegarlo en producción que es donde tiene acceso el público. Estos entornos son:

- Next: En este entorno se prueba la nueva versión del back-end que soporta Bitbloq. Para certificar este entorno, se pasan todas las pruebas, tanto automáticas como manuales, para verificar que los cambios en la nueva versión del back-end no afectan negativamente al funcionamiento de Bitbloq.
- QA: En este entorno se prueba el funcionamiento de Bitbloq. Esta es una versión on-line del servidor en localhost que se puede generar para la validación individual de los cambios.
- Integración: En este entorno se pretende integrar los cambios diarios de Bitbloq para mantener una versión actualizada de Bitbloq.
- MVP: En este entorno se liberan los cambios antes de publicarlos en la versión de producción con la finalidad que que ciertas personas, como pueden ser profesores o personas interesadas en la evolución de Bitbloq puedan probarlo y así dar su opinión sobre los cambios en la aplicación.

Para la certificación de una nueva versión de Bitbloq, se pasan todos los tests, tanto automáticos como manuales en todas las plataformas disponibles, para asegurar que en ninguno de los entornos que se han mencionado anteriormente se han cometido errores fatales.

Una vez se pasan todos los casos de prueba, Testlink genera un informe que permite ver, de forma sencilla, los resultados de pasar los tests manuales y automáticos. A continuación se pueden ver los gráficos generados por la última puesta en producción que se hizo.





Con estos resultados, se estudia la gravedad de los casos de prueba que no se han pasado, con la finalidad de comprobar la gravedad de los errores que se cometen. En caso de que estos errores se consideren menores, se sigue adelante con el proceso de producción. En caso de que se consideren graves, es necesario tomar la decisión de realizar la puesta en producción o cancelarla y resolver el error. Finalmente, en caso de que los errores se consideren bloqueantes, se cancela la puesta en producción hasta que se soluciona el error.

Las plataformas donde se certifica son:

- Windows: en sus versiones de 32 y 64 bits para 7, 8.1 y 10
- Ubuntu
- Mac OS: para la versión Yosemite
- Móviles y tablets: Simplemente para comprobar el mensaje que dice que en estos momentos Bitbloq no es compatible con dispositivos móviles.

Cada vez que se detecta un bug, este se reporta en Jira y se asigna a un desarrollador para que se trate. Una vez el desarrollador soluciona el bug, se comprueba que el funcionamiento es adecuado. Si no se resuelve el bug, o la

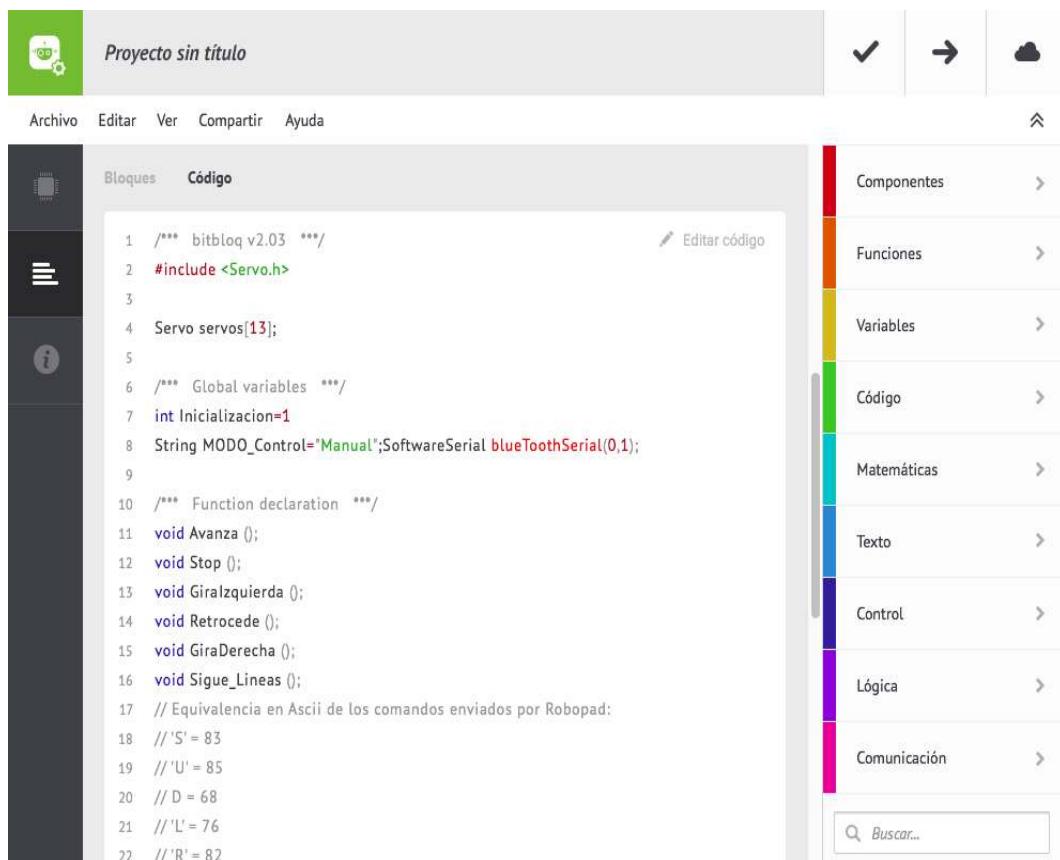
solución aportada no es completamente correcta, se devuelve el bug al desarrollador para que lo resuelva completamente. En caso de que el bug esté completamente solucionado, se aprueban los cambios y se crea un caso de prueba en Testlink para asegurarse que en futuras versiones no aparece este mismo bug.

El proceso de añadir una nueva funcionalidad es similar al de reportar un bug. Cuando se quiere añadir una nueva funcionalidad se crea una tarea en Jira y se asigna a un desarrollador. Una vez el desarrollador completa la tarea se verifica que la funcionalidad está añadida correctamente. Si no funciona de acuerdo a las especificaciones, se devuelve la tarea al desarrollador para que la complete. En caso de que sí que funcione correctamente, se aprueban los cambios y se crea un caso de prueba para comprobar la funcionalidad.

Para controlar la calidad y limpieza del código, no existe ninguna herramienta automática que compruebe que no existen bad smells como pueden ser código repetido o números utilizados directamente en funciones sin definir previamente. Sin embargo, existen revisiones que se realizan de forma manual con cada subida a Gerrit. Una de las funciones de estas revisiones es la de prevenir estos problemas.

3.2.13 IDE, entorno de desarrollo integrado

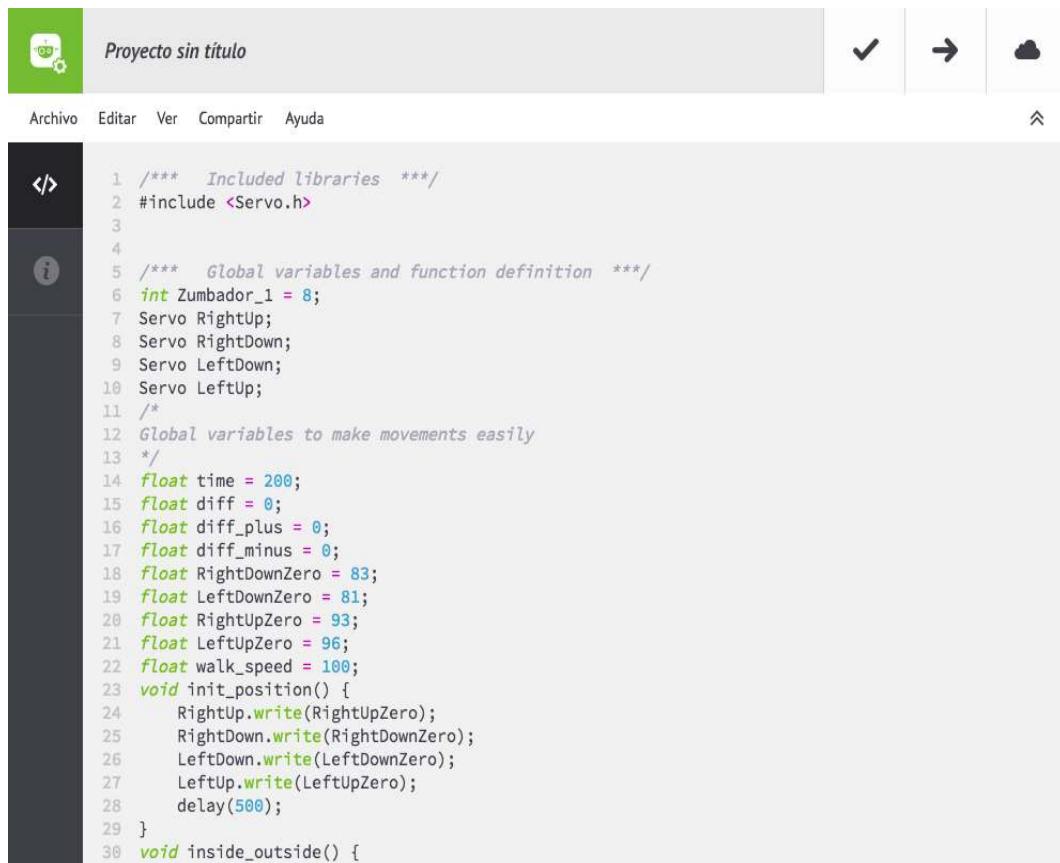
Bitbloq es una plataforma que ofrece un sistema de programación por bloques, en el cual, se va generando el código correspondiente a la estructura de bloques construida. Esta generación de código sirve para la programación de las placas controladoras, no obstante, también es mostrado al usuario para que este pueda ir aprendiendo lo que genera cada bloque. La visualización del código puede realizarse desde la pestaña llamada “Código” donde se define el modo que el usuario quiere visualizar su trabajo. A continuación, se muestra el diseño de la pestaña de código dentro del espacio de trabajo.



Con el aprendizaje del código generado por los bloques, el usuario puede iniciarse en la programación de código arduino y poder realizar su primer programa de código.

Bitbloq oferta un IDE de programación incorporado dentro del flujo propio de la plataforma, haciendo que el usuario no tenga que instalar ningún entorno de desarrollo en el ordenador para programar en código arduino. Con esto, el primer objetivo de Bitbloq se incrementa incorporando el mejor IDE, entorno de desarrollo integrado, de programación para código arduino.

En el diseño anterior, se puede apreciar un elemento que detalla “Editar código”, este hace que el usuario se redirija al IDE propio de Bitbloq para que pueda modificar su código o escribirlo desde cero. A continuación, se muestra el diseño planteado para la edición del código.



The screenshot shows the Botbloq software interface. At the top, there's a toolbar with icons for saving, running, and sharing. Below the toolbar, a menu bar includes Archivo, Editar, Ver, Compartir, and Ayuda. The main area is a code editor with the title "Proyecto sin título". The code is written in C++ and defines global variables and functions for controlling servos:

```
1 *** Included libraries ***
2 #include <Servo.h>
3
4
5 *** Global variables and function definition ***
6 int Zumbador_1 = 8;
7 Servo RightUp;
8 Servo RightDown;
9 Servo LeftDown;
10 Servo LeftUp;
11 /*
12 Global variables to make movements easily
13 */
14 float time = 200;
15 float diff = 0;
16 float diff_plus = 0;
17 float diff_minus = 0;
18 float RightDownZero = 83;
19 float LeftDownZero = 81;
20 float RightUpZero = 93;
21 float LeftUpZero = 96;
22 float walk_speed = 100;
23 void init_position() {
24     RightUp.write(RightUpZero);
25     RightDown.write(RightDownZero);
26     LeftDown.write(LeftDownZero);
27     LeftUp.write(LeftUpZero);
28     delay(500);
29 }
30 void inside_outside() {
```

3.3 Estructura y base de datos.

3.3.1 Documental

Dentro de los diferentes tipos de bases de datos NoSQL están las bases de datos documentales. Estas están constituidas por un conjunto de programas que almacenan, recuperan y gestionan datos en forma de documentos. A diferencia de las bases de datos relacionales, estas bases de datos están diseñadas alrededor de una noción abstracta de "Documento". Estos documentos se encapsulan siguiendo el formato estándar JSON y son organizados en colecciones de documentos. Los documentos de una misma colección no requieren ser ajustados a un mismo esquema estándar ni tener todos las mismas secciones, atributos, claves o cosas por el estilo.

MongoDB MongoDB es un tipo de base de datos NoSQL y más en concreto, es de tipo documental, basándose así en el almacenamiento de documentos.

Los creadores de MongoDB se definen como la base de datos NoSQL líder y la que permite a las empresas ser más ágiles y escalables. Organizaciones de todos los tamaños están usando MongoDB para crear nuevos tipos de aplicaciones, mejorar la experiencia del cliente, acelerar el tiempo de comercialización y reducir costes.

Es una base de datos ágil que permite a los esquemas cambiar rápidamente cuando las aplicaciones evolucionan, proporcionando siempre la funcionalidad que los desarrolladores esperan de las bases de datos tradicionales, tales como índices secundarios, un lenguaje completo de búsquedas y consistencia estricta.

MongoDB ha sido creado para brindar escalabilidad, rendimiento y gran disponibilidad, escalando de una implantación de servidor único a grandes arquitecturas complejas de centros multidatos. MongoDB brinda un elevado rendimiento, tanto para lectura como para escritura, potenciando la computación en memoria (in-memory). La replicación nativa de MongoDB y la tolerancia a fallos automática ofrece fiabilidad a nivel empresarial y flexibilidad operativa.

Las suscripciones de MongoDB ofrecen un servicio de asistencia técnica profesional, licencias comerciales y acceso a características de software de MongoDB

Enterprise. Las suscripciones no solo ayudan a los clientes a lograr una infraestructura de TI estable, escalable y segura, sino también a alcanzar sus objetivos empresariales más amplios, tales como reducir los costes, acelerar el tiempo de comercialización y disminuir los riesgos.

MongoDB Enterprise ofrece seguridad avanzada, monitorización on-premises, soporte SNMP, certificaciones de SO y mucho más. El servicio de gestión de MongoDB (MMS) ofrece funcionalidad de monitorización y respaldo en la nube o bien on-premises como parte de MongoDB Enterprise.

Colecciones de la base de datos

- **iam.user:** Colección en la que se describen los datos relevantes del usuario. Además, es donde se definen los scopes que posee el usuario y, por lo tanto, los permisos y accesos que pueda realizar dicho usuario.

Ejemplo de usuario:

```
{  
    "_id" : ObjectId("123456asdfg"),  
    "domain" : "bitbloq",  
    "email" : "ejemplo@botbloq.com",  
    "username" : "ejemplo",  
    "firstName" : "Ejemplo",  
    "lastName" : "Bloqbloq",  
    "scopes" : [  
        "bitbloq:web"  
    ],  
    "properties" : {  
        "birthday" : "1987-07-07T22:00:00.000Z",  
        "term" : true,  
        "newsletter" : true,  
        "language" : "es-ES",  
        "hasBeenAskedIfTeacher" : true,  
    }  
}
```

```
"tour" : true
},
"groups" : [ ],
"createdDate" : ISODate("2015-11-27T10:44:41.593Z"),
"createdBy" : "2529c929"
}
```

- **resource.bitbloq_Angularproject:** Para gestionar y diferenciar mejor los proyecto que se crean con Bitbloq 1 y los creados con Bitbloq 2, se decidió mantener dos colecciones diferentes: angularproject y project. Dentro de la colección “bitbloq_Angularproject”, correspondiente a los proyectos de los usuarios que se generan en Bitbloq 2, se puede distinguir tres tipos de proyectos: de bloques, de código y de robots. Además, también se incorpora en los proyectos un nuevo concepto: Access Control List (ACL). Los ACLs serán usados para establecer si un proyecto es privado, público o incluso, si es compartido con algún usuario.

Se define un proyecto público con el siguiente acl:

```
"_acl" : {
  "user:123456asdfg" : {
    "permission" : "ADMIN",
    "properties" : {}
  },
  "ALL" : {
    "permission" : "READ",
    "properties" : {
      "date" : "2015-12-10T16:49:10.557Z"
    }
  }
}
```

Por la gran extensión de los ejemplos de este apartado, se ha añadido un ejemplo de proyecto de bloque, otro de código y por último un proyecto de robots en el anexo IV. “Ejemplo de proyectos de bloques, código y de robots de Bitbloq 2”

- **resource.bitbloq_Bloqs:** Colección donde se definen la estructura de los bloques de bitbloq 2. En el apartado 3.2.7 se detalla más información acerca de estos bloques.

Ejemplo:

```
{  
    "_id" : ObjectId("159786asdfasdf"),  
    "type" : "output",  
    "name" : "char",  
    "connectors" : [  
        {  
            "type" : "connector--output",  
            "accept" : "connector--input"  
        }  
    ],  
    "bloqClass" : "bloq-string",  
    "content" : [  
        [  
            {  
                "alias" : "text",  
                "value" : ""  
            },  
            {  
                "id" : "TEXT",  
                "alias" : "charInput",  
                "placeholder" : "bloq-char"  
            },  
            {  
                "alias" : "text",  
                "value" : "  
            }  
        ]  
    ],  
    "code" : "{TEXT}",  
}
```

```
"returnType" : {  
    "type" : "simple",  
    "value" : "char"  
}  
}
```

- **resource.bitbloq_Board:** Colección donde se describen las diferentes placas disponibles. Esta colección es usada a la hora de listar las placas para generar un proyecto tanto de bloques como de código.

Ejemplo de placa:

```
{  
    "_id" : "FT232R_USB_UART",  
    "name" : "bq ZUM",  
    "arch" : "arduino",  
    "board" : "bt328",  
    "bitrate" : NumberLong(19200),  
    "maxPageSize" : NumberLong(128),  
    "delay_reset" : NumberLong(100),  
    "max_size" : NumberLong(28672),  
    "_createdAt" : ISODate("2015-12-09T08:12:48.438Z"),  
    "_updatedAt" : ISODate("2015-12-09T08:12:48.438Z")  
}
```

- **resource.bitbloq_ChangeLogs:** En cada versión producida de Bitbloq, se recopila todos los cambios y mejoras realizadas y los errores solucionados para notificarlos a los usuarios en forma de listado. Dicha información se puede ver en el apartado de actualizaciones y bugs dentro de la pestaña de “Ayuda”.

Ejemplo de change log:

```
{  
    "_id" : ObjectId("12584sdfgsdfg"),  
    "version" : {  
        "ca-ES" : "Bitbloq beta v2.0.7",  
        "de-DE" : "Bitbloq beta v2.0.7",  
        "en-GB" : "Bitbloq beta v2.0.7",  
        "es-ES" : "Bitbloq beta v2.0.7",  
        "eu-ES" : "Bitbloq beta v2.0.7",  
        "fr-FR" : "Bitbloq beta v2.0.7",  
        "gl" : "Bitbloq beta v2.0.7",  
        "it-IT" : "Bitbloq beta v2.0.7",  
        "nl-NL" : "Bitbloq beta v2.0.7",  
        "pt-PT" : "Bitbloq beta v2.0.7",  
        "ru-RU" : "Bitbloq beta v2.0.7",  
    },  
    "content" : [  
        {  
            "es-ES" : "Arreglos de maquetación",  
            "gl" : "Arranxos de maquetación",  
        },  
        {  
            "es-ES" : "Revisar recuperación de contraseña",  
            "gl" : "Revisar recuperación do contrasinal",  
        },  
        {  
            "es-ES" : "Actualizar FAQ's",  
            "gl" : "Actualizar FAQ's ",  
        },  
        {  
            "es-ES" : "Resueltos problemas al actualizar web2board en Mac OS X",  
            "gl" : "Resoltos problemas ao actualizar web2board en Mac OS X",  
        },  
    ]  
}
```



- **resource.bitbloq_Faqs**: Colección donde se recopilan las preguntas más frecuentes de los usuarios junto con su respuesta. El usuario podrá consultar dichas preguntas en la pestaña de ayuda de bitbloq.

Ejemplo de pregunta frecuente:

```
{  
  "_id" : ObjectId("56581ad94de21e5345c4881e"),  
  "title" : {  
    "ca-ES" : "A la bq ZUM, no se m'enega l'LCD.",  
    "de-DE" : "Die LCD der BQ ZUM schaltet sich nicht ein.",  
  },  
}
```

"en-GB" : "The LCD is not showing anything in the bq ZUM board",
"es-ES" : "No se enciende el LCD cuando se conecta a la bq ZUM",
"eu-ES" : "ZUM bq-an ez zait LCD-a pizten.",
"fr-FR" : "L'écran LCD ne s'allume pas avec ma BQ ZUM",
"gl" : "Non se acende o LCD cando se conecta á bq ZUM",
"it-IT" : "Nella bq ZUM, non si accende l'LCD",
"pt-PT" : "Na bq ZUM, o LCD não acende",
},
"content" : {

"ca-ES" : "Assegura't que el botó "on/off" de la placa està connectat, perquè si no ho està, els components senzills sí que funcionen, però els que necessiten més potència, no funcionaran.",

"de-DE" : "Vergewissere dich, dass die on/off-Taste der Platine verbunden ist. Ist dies nicht der Fall, funktionieren zwar die einfacheren Komponenten, aber diejenigen, die mehr Leistung benötigen, funktionieren nicht.",

"en-GB" : "Check that the switch on the board is \"ON\". If it is \"OFF\", some components may work whereas others that need higher current will not.",

"es-ES" : "Asegúrate de que el botón on/off de la placa está en posición on, de lo contrario, los componentes sencillos sí funcionarán, pero aquellos componentes que necesiten mayor potencia, no funcionarán.",

"eu-ES" : "Egiaztatu plakaren on/off botoia konektatuta dagoela. Bestela, osagai simpleak funtzionatuko dute, baina potentzia gehiago behar duten osagaiak ez.",

"fr-FR" : "Vérifie que le bouton ON/OFF de ta carte est sur ON. Dans le cas contraire les composants de base fonctionnent mais ce n'est pas le cas des composants nécessitant plus de puissance.",

"gl" : "Asegúrache de que o botón on/off da placa está en posición on, pola contra, os compoñentes sinxelos si funcionarán, pero aqueles compoñentes que necesiten maior potencia, non funcionarán.",

"it-IT" : "Assicurati che il pulsante ON/OFF della scheda sia collegato: in caso contrario, i componenti semplici funzioneranno, ma non quelli che necessitino di una maggiore potenza.",

"pt-PT" : "Verifica que o botão on/off da placa está conectado, no caso contrário, os componentes simples funcionam, mas os componentes que precisam

de mais potência não.",

```
},  
    "order" : 17.000000000000000000000000  
}
```

- **resource.bitbloq_Feedback:** En esta colección se guardarán todos los mensajes que el usuario nos manda, tanto mensajes para reportar errores como mensajes de comentarios y sugerencias. Internamente se genera un correo correspondiente a cada uno de los mensajes que envía el usuario. Este correo es enviado a soporte de bitbloq .

Ejemplo:

```
{  
    "_id" : "123456asdfg:184asdfas4df29521b",  
    "message" : "Contenido del mensaje que escribe el usuario. Prueba para  
documentación",  
    "os" : "Ubuntu",  
    "browser" : "47",  
    "userAgent" : "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36  
(KHTML, like Gecko) Chrome/47.0.2526.27 Safari/537.36",  
    "userInfo" : {  
        "_id" : "123456asdfg",  
        "createdBy" : "15648965",  
        "domain" : "bitbloq",  
        "email" : "ejemplo@botbloq.com",  
        "username" : "ejemploUser",  
        "firstName" : "Ejemplo",  
        "lastName" : "Test",  
        "scopes" : [  
            "bitbloq:web"  
        ],  
        "properties" : {  
            "term" : true,  
        }  
    }  
}
```

```
    "newsletter" : true,  
    "language" : "es-ES",  
    "hasBeenAskedIfTeacher" : true,  
    "cookiePolicyAccepted" : true,  
    "tour" : true,  
    "code" : true  
},  
    "groups" : []  
},  
    "_createdAt" : ISODate("2015-12-10T10:30:29.915Z"),  
    "_updatedAt" : ISODate("2015-12-10T10:30:29.915Z")  
}
```

- **resource.bitbloq_Program:** Para bitbloq 1 se implementó dentro de corbel un compilador de arduino. Con esto se pretendía acelerar el proceso de compilación de código en la nube y generación de feedback al usuario sin sobrecargar demasiado su propio navegador. Internamente la plataforma implementa un mecanismo de cacheado de binarios pre-compilados que reaperovecha recursos dentro del servidor para ofrecer una mayor velocidad de respuesta en las operaciones de compilado remoto.

Ejemplo:

```
{  
    "_id" : "a1asd516fa819sdf89asd4f",  
    "_createdAt" : ISODate("2015-11-30T09:32:40.060Z"),  
    "code" : "void setup()\{} void loop()\{char x[] = \\"char9218445157506236788-  
28+1448875865056\\\";\\}",  
    "type" : "arduino",  
    "board" : "atmega328",  
    "state" : "FINISHED",  
    "_updatedAt" : ISODate("2015-11-30T09:32:46.959Z")  
}
```

- **resource.bitbloq_Project**: Diferenciar entre un proyecto puede público o privado nos permite definir quien puede visualizar el proyecto. En Bitbloq 1, se hace una gran separación entre estos dos tipos de proyecto, creando así el concepto de release (proyecto público). Las releases serán guardadas en una colección diferente que definiremos más adelante. Mientras que los proyectos privados de los usuarios serán almacenados en la colección “bitbloq_Project”.

Ejemplo de proyecto privado:

```
{  
  "id": "123456asdfa:8f15639f-727b-4e44-8de1-6d4dedc2a10a",  
  "createdAt": 1425820681934,  
  "name": "blanca",  
  "description": "Proyecto privado",  
  "userTags": [ ],  
  "userId": "123456asdfa",  
  "imageType": "image/jpg",  
  "updatedAt": 1425820681934,  
  "tags": ["lcd", "bluetooth", "servo", "button", "buzzer", "led"]  
}
```

Esta información se complementa con un xml con el código y bloques del proyecto guardado con el mismo id en s3. Por la gran extensión del archivo xml se ha añadido dicho archivo en el anexo III. “Ejemplo de xml de un proyecto de Bitbloq 1”

- **resource.bitbloq_ProjectStats**: Cada proyecto posee sus propias estadísticas de visitas, descargas y número de copias realizadas por otros usuarios.

Ejemplo de estadísticas de un proyecto:

```
{  
  "_id" : "123456asdfa:8f15639f-727b-4e44-8de1-6d4dedc2a10a",  
  "_createdAt" : ISODate("2015-11-27T10:02:44.902Z"),
```

```
"timesViewed" : NumberLong(8),  
"timesDownloaded" : NumberLong(0),  
"timesAdded" : NumberLong(0),  
"twitterCount" : NumberLong(0),  
"facebookCount" : NumberLong(0),  
"googleCount" : NumberLong(0),  
"_updatedAt" : ISODate("2015-11-27T10:02:44.902Z")  
}
```

- **resource.bitbloq_Properties:** En esta colección se definen las características que poseen cada bloque de la pestaña de software.

Ejemplo de propiedades de bloques:

```
{  
    "_id" : ObjectId("56581ada4de21e5345c48822"),  
    "web2boardVersion" : "1.0.0",  
    "bitbloqLibsVersion" : "0.0.4",  
    "bloqsSortTree" : {  
        "components" : [  
            {  
                "name" : "hts221Temperature",  
                "addClass" : "submenu__item submenu__item--component"  
            },  
            {  
                "name" : "hts221Humidity",  
                "addClass" : "submenu__item submenu__item--component"  
            },  
            ....  
            ....  
            ....
```

- **resource.bitbloq_Release:** Colección correspondiente a los proyectos públicos del usuario en Bitbloq 1. Estos proyectos son visibles a todos los usuarios, pero el dueño del proyecto es el único que puede sobreescribirlo.

Ejemplo de proyecto público:

```
{  
    "description": "Proyecto publicado",  
    "userId": "123456asdfa",  
    "tags": ["bat", "serial"],  
    "createdAt": 1447865611981,  
    "timesDownloaded": 0,  
    "compiled": true,  
    "timesViewed": 0,  
    "_createdAt": 1447865600037,  
    "name": "Salvaobstacle",  
    "userTags": [],  
    "id": "123456asdfa:85603ae3-77b0-4e63-adbb-5c1d5072c540",  
    "imageType": "image/jpeg",  
    "_updatedAt": 1447865600037,  
    "updatedAt": 1447865611981  
}
```

3.3.2 Autenticación y registro

El acceso y el registro de usuarios es un tema muy importante y, por lo tanto, requiere una mayor atención y análisis.

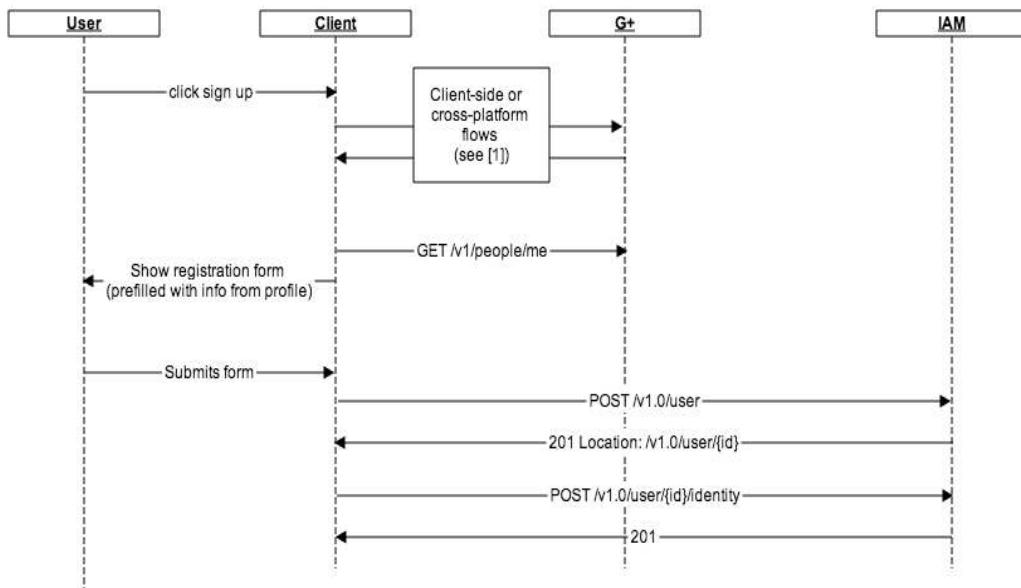
Los procesos definidos para estas acciones con redes sociales son los que se presentan en los siguientes apartados.

En todos los diagramas presentados se muestran los siguientes agentes:

- User: es el sujeto que desea entrar en el sistema.

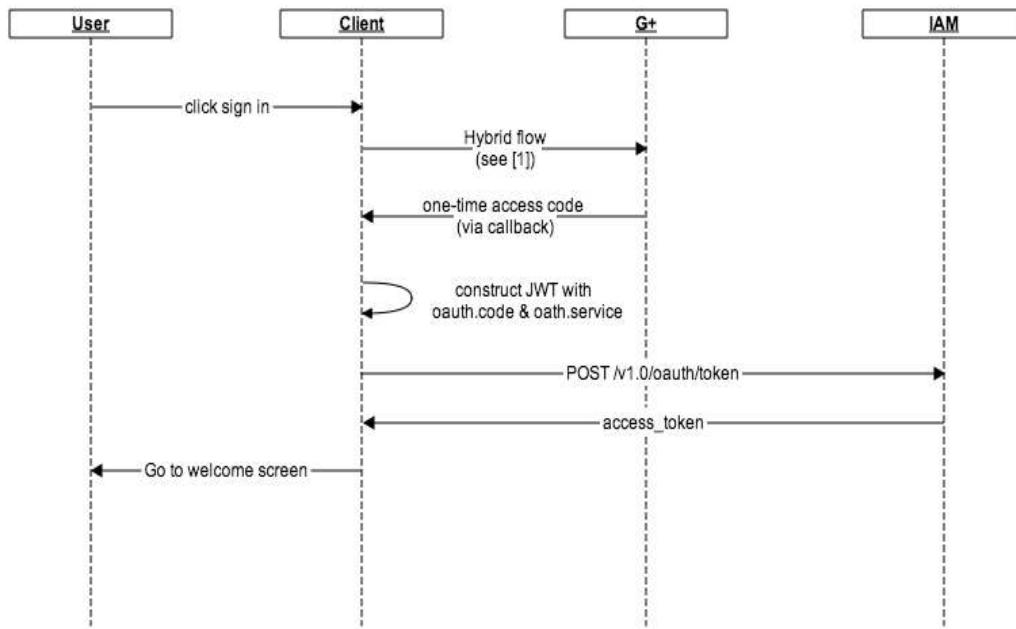
- Client: corresponde a aplicación con la que interactúa el usuario, es decir, Bitbloq.
- G+: Red social que se ha puesto como ejemplo. Esto no significa que tan sólo se pueda entrar en el sistema con google plus.
- IAM: es la parte de la base de datos correspondiente a la gestión y almacenaje de los datos del usuario.

Registro con Google plus, G+ En primer lugar el usuario realiza la acción que detalla que desea registrarse en el sistema a través de una red social. A continuación el cliente debe meter sus datos de la red social seleccionada. En el momento que tenemos las credenciales del usuario se registra la información del usuario en la colección ‘user’ y también se guarda la relación con dicha red social en la colección ‘identity’. Esta última guarda todas las relaciones con redes sociales que posee el usuario en nuestro sistema.



[1] <https://developers.google.com/+/web/signin/>

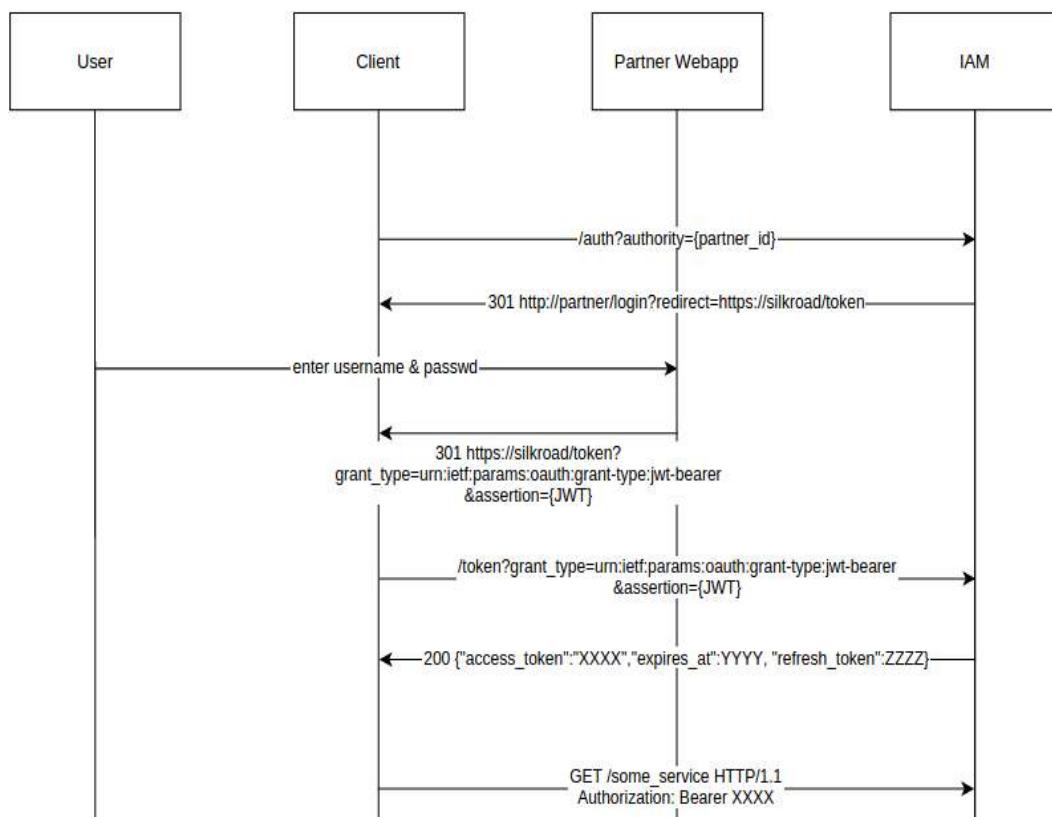
Autenticación con Google plus, G+ El siguiente diagrama indica el flujo de autenticación en G+



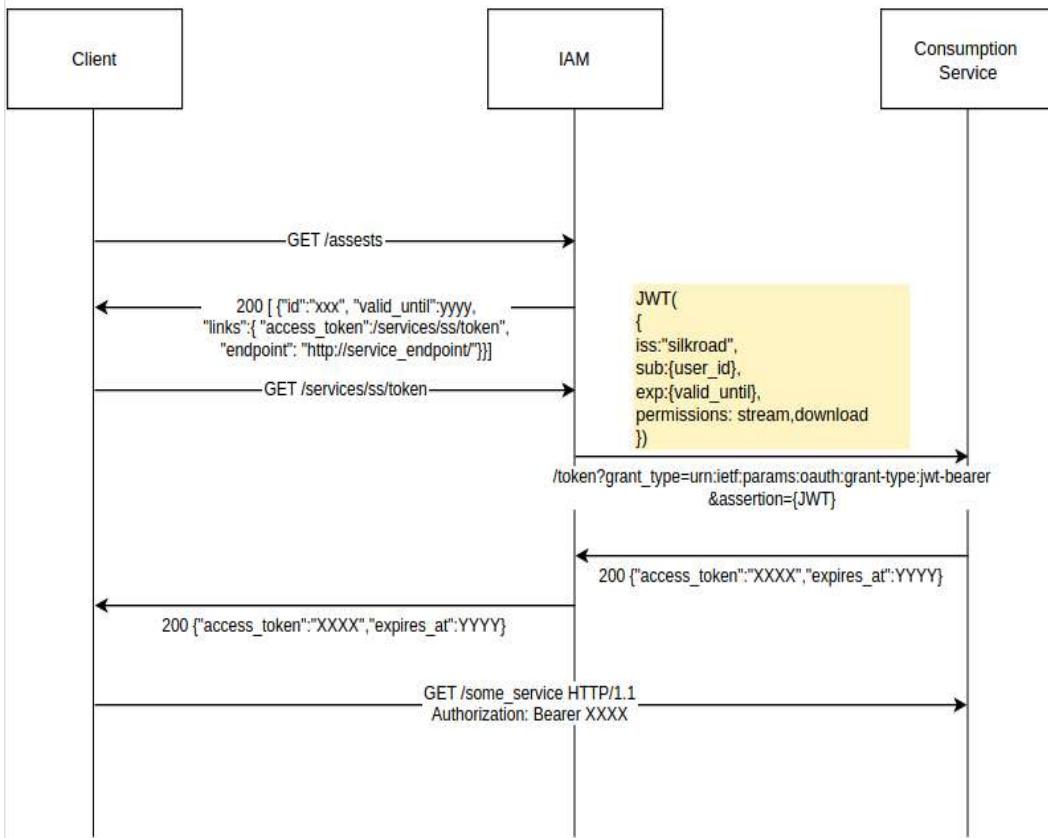
[1] https://developers.google.com/+/web/signin/server-side-flow#implementing_the_one-time-code_flow

Protocolos de autenticación El siguiente diagrama indica el flujo de protocolo de autenticación

Authenticating a partner user with SilkRoad



Transfer of authority from SilkRoad to Consumption Services



3.4 Integración con ecosistema DIWO

Como con casi todas las herramientas para el aprendizaje necesitamos tutoriales, ejemplos, ideas, en general todo un ecosistema que pueda ayudar a los alumnos a usar Bitbloq y aprender en el proceso. Por esa razón hemos incluidos enlaces a Diwo, a través de la sección de ayuda (<http://bitbloq.bq.com/#/help/tutorial>) a donde se puede ir a los tutoriales que se han preparado específicos de Bitbloq. También se ha habilitado una API Rest en DIWO para que podamos acceder a todo su contenido, eso nos permitirá desarrollar una integración mucho mayor. Está actualmente en fase de implementación, pero se quiere hacer el acceso mucho más directo, permitiendo al utilizar un componente hardware o un bloque acceder a un tutorial con un solo Click.



4 Futuros desarrollos de la plataforma

A largo plazo hay definidas unas funcionalidades que tienen por objetivo complementar y escalar la herramienta para añadir opciones y calidad al entorno. No están implementadas actualmente, pero se tiene prevista su implantación.

Algunas de las funcionalidades previstas para desarrollo son:

4.1 Profesores y alumnos

La herramienta está enfocada a la educación y como además está metida en planes de enseñanza del sistema educativo español, es necesario crear una gestión de usuarios y unas funcionalidades específicas para esta parte.

Es necesario dotar a la herramienta de una parte de creación de usuario llamado “profesor” que pueda registrar usuarios (alumnos) y agruparlos por grupos (clases). Este profesor debería poder dar de alta a usuarios y el sistema generaría unos usuarios con nombre de usuario y una contraseña para poder acceder. El profesor también debería poder gestionar las contraseñas de los alumnos por si hay que modificar, recordar u otras acciones.

Una vez que los alumnos están creados, también será necesario poder agrupar alumnos en clases, teniendo acciones por usuario y por grupo completo.

Una funcionalidad imprescindible es que el profesor pueda compartir con alumnos y/o clases proyectos específicos, pudiendo modificar el proyecto y que se sincronice con todos los usuarios compartidos.

Por parte de los alumnos es necesario poder tener un estado de “enviar al profesor” para que el profesor reciba los proyectos de los alumnos y pueda entrar a ver el proyecto de ese alumno.

Un requisito que sería complementario sería que ese alumno que ha sido creado con un usuario por un profesor, si el alumno quiere registrarse manualmente en la herramienta y ya tiene esos proyectos creados en la cuenta invitada, puedan unificarse y asignar ese usuario invitado por el profesor al usuario registrado por el alumno.

La herramienta está enfocada a la educación y como además está metida en planes de enseñanza del sistema educativo español, es necesario crear una gestión

de usuarios y unas funcionalidades específicas para esta parte.

Es necesario dotar a la herramienta de una parte de creación de usuario llamado “profesor” que pueda registrar usuarios (alumnos) y agruparlos por grupos (clases). Este profesor debería poder dar de alta a usuarios y el sistema generaría unos usuarios con nombre de usuario y una contraseña para poder acceder. El profesor también debería poder gestionar las contraseñas de los alumnos por si hay que modificar, recordar u otras acciones.

Una vez que los alumnos están creados, también será necesario poder agrupar alumnos en clases, teniendo acciones por usuario y por grupo completo.

Una funcionalidad imprescindible es que el profesor pueda compartir con alumnos y/o clases proyectos específicos, pudiendo modificar el proyecto y que se sincronice con todos los usuarios compartidos.

Por parte de los alumnos es necesario poder tener un estado de “enviar al profesor” para que el profesor reciba los proyectos de los alumnos y pueda entrar a ver el proyecto de ese alumno.

Un requisito que sería complementario sería que ese alumno que ha sido creado con un usuario por un profesor, si el alumno quiere registrarse manualmente en la herramienta y ya tiene esos proyectos creados en la cuenta invitada, puedan unificarse y asignar ese usuario invitado por el profesor al usuario registrado por el alumno.

4.2 Detección del esquema hardware automáticamente

La herramienta tiene una parte de hardware donde el usuario añade una placa y unos componentes para después programarlo en la parte de software.

En el futuro, si el usuario ya tiene conectados los componentes a la placa de forma física y la conecta mediante USB al ordenador, la plataforma podría reconocer el modelo de placa y los componentes que tiene conectados para agilizar el paso de la parte de hardware.

4.3 Programación inalámbrica

Con el componente Bluetooth conectado a una placa y un ordenador con la tecnología Bluetooth podría establecerse una conexión entre ellos para poder cargar los programas sin necesidad de cables.

Habría un proceso de detección de placas cerca conectadas por Bluetooth para seleccionar la que quieras programar y una acción de programar sin cables para activar esta función.

4.4 Widgets

Con la nueva herramienta Web2Board es posible añadir widgets que complementen la plataforma con información referente a los componentes. Por ejemplo se podría medir la distancia que va midiendo un ultrasonidos, la temperatura y humedad, o el sonido que recibe un micro, los grados de un potenciómetro, etc... Crear widgets para recibir más feedback de los componentes es una opción que algunos usuarios han demandado y que se va a llevar a cabo en un futuro cercano.

4.5 Varias placas

Muchos usuarios, principalmente educadores que tienen que preparar talleres y clases para varios alumnos, demandan una opción que sea poder cargar todas las placas que tengan conectadas y no tengan que ir una por una. Para ellos se ha planteado una forma de poder conectar la cantidad que se quiera de placas y que la plataforma las reconozca todas y de la opción al usuario de poder elegir en cuál/cuáles cargar, o incluso cargar el programa en todas ellas.

4.6 Detección de errores

Una mejora para la comprensión de errores en la parte de programación sería poder indicar qué conflictos se encuentran en el programa. Para ello habría que poder ir compilando el programa e ir indicando los bloques que van creando conflicto según

se vaya creando. Incluso poder recomendar sugerencias de bloques para hacer más ágil el proceso de creación del programa.

4.7 Robots

En la parte de Hardware, que es donde se encuentran las placas y los componentes, habría que incluir un apartado de robots ya creados con un esquema de componentes predefinidos que se venden en el mercado para agilizar el proceso de creación del hardware, en vez de tener que seleccionar la placa e ir añadiendo componentes.

Al final de la segunda fase de este proyecto, se incorporó el primer robots, llamado Zowi. Este es un módulo de programación que incluye una placa diseñada específicamente para Zowi que se puede corresponderse con una placa Arduino Uno y también incluye el siguiente listado de componentes:

- 4 Servomotores, corresponden a dos servomotores por pata del robot.
- Sensor de ultrasonidos, los ojos
- Micrófono, las orejas
- Zumbador, el sonido que genera el robot
- 2 Botones programables y otro de encendido y apagado
- Módulo bluetooth, comunicación con la aplicación móvil
- Matriz LED, para realizar las diferentes expresiones con la boca

En el empleable E3.2 se detalla más en profundidad cómo se ha abordado Zowi y otros proyectos de robots como es PrintBot.

4.8 Integración en ROS

ROS, en inglés Robot Operating System, Sistema Operativo Robótico, es un framework para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo en un clúster heterogéneo.

ROS provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes.

En Bitbloq se pretende trabajar con ROS para poder facilitar las comunicación entre los diversos robots disponibles y Bitbloq. También se pretende realizar con ROS las interconexiones entre los diferentes nodos disponibles y poder realizar operaciones entre ellos.

Además, al hacer uso de ROS y ser más genérico nos aportaría un mayor alcance en el uso y programación de nuestros componentes o robots, pudiendo realizar simulaciones dentro de la aplicación de Bitbloq, utilizar las operaciones básicas que el framework ofrece y poder enlazar con otros proyectos desarrollados en ROS.

Por último, ROS posee una gran comunidad en la que podríamos apoyarnos para futuros desarrollos.

5 Anexos

5.1 Anexo I. Infraestructura de servicios.

Corbel como plataforma

Corbel es una plataforma genérica que ofrece todo lo necesario para desarrollar cualquier aplicación de forma rápida y sencilla, así facilitando puntos clave como la autenticación, la gestión de recursos y eventos de la aplicación. Además es de código abierto, desarrollada bajo licencia Apache 2. Esta plataforma tan sólo posee dependencias con maven y con docker compose.

Conceptos básicos: Autenticación y autorización

Los conceptos básicos que implementa corbel son la autenticación y la autorización. Para entender estos términos primero se ha de explicar los de dominios, clientes y usuarios.

Dominios, clientes y usuarios

Corbel cuenta con una estructura organizativa para gestionar los diferentes dominios, aplicaciones y usuarios. Definimos dentro de la terminología corbel:

- **Domain:** un dominio/producto gestionado dentro de corbel
- **Client:** una de las aplicaciones perteneciente a un Domain
- **User:** uno de los usuarios perteneciente a un Domain

La lista de dominios, clientes y usuarios quedarán almacenadas en la base de datos de IAM. A continuación se muestra un ejemplo de Domain, Client y User que se usará para siguientes ejemplos:

- **iam.domain**

```
{  
  "_id" : "orpheus",  
  "createdBy" : "IamSell on integration-silkroad-app.dev",  
  "createdDate" : ISODate("2015-02-26T15:02:13.383Z"),  
  "description" : "Domain for Orpheus",  
  "scopes" : [  
    ...  
  ]  
}
```

- **iam.client**

```
{  
  "_id" : "d2d9eda7",  
  "createdBy" : "IamSell on integration-silkroad-app.dev",  
  "createdDate" : ISODate("2015-02-26T15:02:14.615Z"),  
  "domain" : "orpheus",  
  "key" : "26dbeb5dcf12e4d3909f8db3435d27d300b256e515b147bbba960dec1d627a2a",  
  "name" : "orpheus-web",  
  "scopes" : [  
    ...  
  ],  
  "signatureAlgorithm" : "HS256"  
}
```

- **iam.user**

```
{  
  "_id" : "74427e62a44dc48ae8da70d2f3da996d",  
  "createdBy" : "IamSell on integration-silkroad-app.dev",  
  "createdDate" : ISODate("2015-02-26T15:02:10.020Z"),  
  "domain" : "orpheus",  
  "email" : "silkroad.qa@gmail.com",
```

```
"firstName" : "silkroad-qa",
"scopes" : [
...
],
"username" : "SilkroadUser"
}
```

Scopes

En el ejemplo detallado anteriormente se ha usado un nuevo término, scopes. Llamamos Scopes al mecanismo genérico que permite realizar el control de acceso a los diferentes recursos de cada módulo de corbel. Puesto que corbel es una plataforma de contenidos digitales genérica, el mecanismo de gestión de acceso está pensado para adaptarse a cualquier situación.

Scopes se implementa mediante una serie de reglas, donde todos los accesos estarán prohibido salvo que cumplan con al menos un Scope. Cada Scope contará con un nombre único dentro del sistema. Todas las peticiones que se realizan a los módulos corbel han de contener un accessToken generado por IAM a partir de las credenciales de clientes y usuarios. A la hora de generar el accessToken, se pedirá a IAM la lista de Scopes que se requieran, lista que se comprobará si es legal de acuerdo a la configuración interna que más tarde se detallará.

Los Scopes permiten controlar el acceso de las siguientes formas:

- Control a nivel de módulo de corbel
- Control a nivel de tipo de dato (Content-Type) de la petición
- Control a nivel de método HTTP
- Control a nivel de URIs
 - Soporte de expresiones regulares
 - Soporte de plantillas
 - * Reemplazo por identificación de usuario

Antes de continuar, se debe ver cómo se define en corbel los conceptos de dominio (o producto), aplicaciones y usuarios.

Por definición de Scopes se puede entender dos procesos:

- Definición de dónde aplica cada Scope, es decir, la lista de Scopes soportados por cada Domain, Client y User
- Definición de cómo se define cada Scope: reglas de acceso

La definición de cada Scope también quedará almacenada en la base de datos de IAM. Para describir cómo se define un Scope concreto, partamos de un ejemplo:

```
iam.scope
{
  "_id" : "resources:music:streaming",
  "audience" : "http://resources.bqws.io",
  "rules" : [
    {
      "mediaTypes" : [
        "audio/mp3",
        "audio/aacp"
      ],
      "methods" : [
        "GET"
      ],
      "type" : "http_access",
      "uri" : "v.*/resource/music:Track/*"
    }
  ]
},
```

```
{  
  "_id" : "resources:music:edit_playlist",  
  "audience" : "http://resources.bqws.io",  
  "rules" : [  
    {  
      "mediaTypes" : [  
  
        "application/json"  
      ],  
      "methods" : [  
  
        "PUT",  
  
        "POST"  
      ],  
      "type" : "http_access",  
      "uri" : "v.*/resource/music:Playlist/-.*"  
    },  
    {  
      "mediaTypes" : [  
  
        "application/json"  
      ],  
      "methods" : [  
  
        "POST"  
      ],  
      "type" : "http_access",  
      "uri" : "v.*/resource/music:Playlist/?"  
    }  
  ]  
}
```

En este ejemplo se puede ver que se han definido dos Scopes : resources:music:streaming y resources:music:edit_playlist.

Para ellos se ha registrado:

- audience: Indica en qué módulo de corbel aplica este Scope. Los valores pueden ser:

- <http://iam.bqws.io> para el módulo de autenticación IAM
- <http://resources.bqws.io> para el módulo recursos Resources
- <http://evci.bqws.io> para el módulo de notificaciones EVCI
- <http://ec.bqws.io> para el módulo de comercio electrónico EC

- rules: Lista de reglas que se aplican para ver si la petición se ajusta al Scope (sirve con una de la lista para que sea válido):

- Cada regla es un diccionario con los siguientes campos:
 - * mediaTypes: formato de datos que ha de llevar la petición
 - * methods: Lista de operaciones HTTP permitidas en el Scope
 - * type: tipo de acceso
 - * uri: ruta de acceso. Las rutas soportan:
 - Expresiones regulares, como por ejemplo */resource/music:Playlist/?
 - Plantillas, como por ejemplo v.*/resource/music:Playlist/-.*

De esta forma si un Domain, Client y User contienen el Scope resources:music:edit_playlist, y se obtiene un accessToken de IAM con el correspondiente Scope la siguiente petición sería válida:

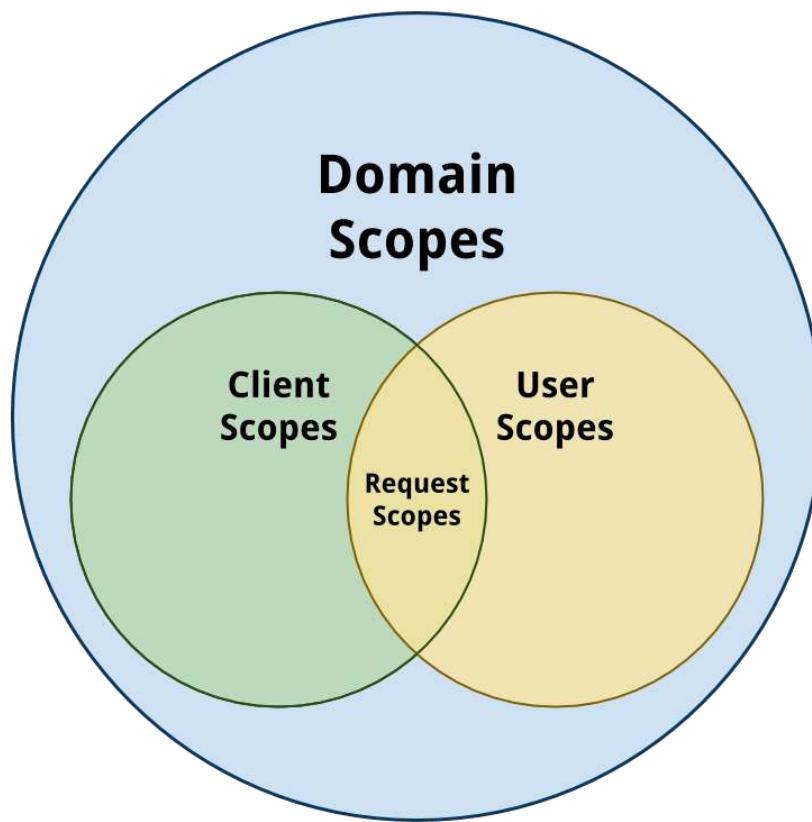
```
curl 'http://resources-int.bqws.io/v1.0/resource/music:Playlist/'  
-H 'Authorization: Bearer token'  
-H 'Content-Type: application/json'  
-H 'Accept: application/json'  
--data "{playlist:my_playlist}"
```

Tras ver este ejemplo, se debe explicar dónde se aplican estos scopes. Los Scopes se pueden aplicar tanto a Domain, como Client o User, para conseguir así

la configuración de acceso deseada. El uso de los Scopes en los diferentes niveles implica:

- Domain: Cada Domain podrá definir todos los Scopes que se van a utilizar para un producto concreto.
- Client: Cada Client podrá definir sus propios Scopes, siempre y cuando sean un subconjunto de los del Domain al que pertenece.
- User: Cada User podrá definir sus propios Scopes como un subconjunto de los del Domain al que pertenece.

Finalmente, una petición utilizará la intersección de los Scopes definidos por Client y User, que a su vez será un subconjunto de los Scopes definidos por Domain.



La lista de Scopes quedará almacenadas en la base de datos de IAM. Esta lista contendrá los nombre únicos que identifican a cada Scope. A continuación se

muestra un ejemplo con las entradas necesarias en la base de datos para definir una serie de Scopes para un Domain, Client y User:

```
iam.domain
{
  "_id" : "orpheus",
  "createdBy" : "IamSell on integration-silkroad-app.dev",
  "createdDate" : ISODate("2015-02-26T15:02:13.383Z"),
  "description" : "Domain for Orpheus",
  "scopes" : [
    "resources:music:read_catalog",
    "resources:music:edit_playlist",
    "resources:music:streaming",
    "iam:user:create",
    "iam:user:delete",
    "iam:user:read"
  ]
}

iam.client
{
  "_id" : "d2d9eda7",
  "createdBy" : "IamSell on integration-silkroad-app.dev",
  "createdDate" : ISODate("2015-02-26T15:02:14.615Z"),
  "domain" : "orpheus",
  "key" : "26dbeb5dcf12e4d3909f8db3435d27d300b256e515b147bbba960dec1d627a2a",
  "name" : "orpheus-web",
  "scopes" : [
    "resources:music:read_catalog",
    "resources:music:edit_playlist",
```

```
"resources:music:streaming",
"iam:user:create",
"iam:user:read"
],
"signatureAlgorithm" : "HS256"
}

iam.user
{
  "_id" : "74427e62a44dc48ae8da70d2f3da996d",
  "createdBy" : "IamSell on integration-silkroad-app.dev",
  "createdDate" : ISODate("2015-02-26T15:02:10.020Z"),
  "domain" : "orpheus",
  "email" : "silkroad.qa@gmail.com",
  "firstName" : "silkroad-qa",
  "scopes" : [
    "resources:music:read_catalog",
    "resources:music:edit_playlist",
    "iam:user:create",
    "resources:music:streaming"
  ],
  "username" : "SilkroadUser"
}
```

En dicho ejemplo se puede observar que:

- Se ha definido un dominio con nombre de producto orpheus que permite realizar una serie de acciones correspondientes a la lista de Scopes:
 - resources:music:read_catalog
 - resources:music:edit_playlist
 - resources:music:streaming

- iam:user:create
 - iam:user:delete
 - iam:user:read
-
- Cabe destacar que estos Scopes son nombres únicos que designan a un Scope, y a pesar de tener un nombre relevante, no aportan nada a la semántica de la regla.
 - Se ha definido un cliente para el dominio orpheus denominado orpheus-web , que podrá hacer todas las operaciones definidas en su dominio, salvo iam:user:delete, lo que significa que orpheus-web no podrá llevar a cabo la acción definida por las reglas iam:user:delete.
 - Se ha definido un usuario para el dominio orpheus denominado Silkroad-User que podrá hacer todas las operaciones definidas en su dominio, salvo iam:user:delete, lo que significa que orpheus-web no podrá llevar a cabo la acción definida por las reglas iam:user:delete y iam:user:read.

Los scopes pueden ser clasificados en:

Scopes compuestos

Es posible definir scopes que engloben a varios scopes. Una definición válida de scope compuesto podría ser:

```
{  
  _id: "bitbloq:web",  
  type: "composite_scope",  
  rules: [],  
  scopes: [  
    "resources:bitbloq:avatar:edit",  
    "evci:comp:base",
```

```
"resources:bitbloq:program",  
  
"resources:bitbloq:boards",  
"iam:comp:base",  
  
"resources:bitbloq:avatar:view",  
  
"resources:bitbloq:project",  
  
"resources:bitbloq:release"  
]  
}
```

Scopes con parámetros configurables

Los scopes soportan parámetros definidos con una expresión regular. Puede verse un ejemplo a continuación:

Supongamos que queremos prestar un libro desde el módulo de préstamos. Esto se haría mediante un asset que permitiría acceder temporalmente a ese libro concreto. Sin los scopes configurables, habría que crear un scope por cada uno de los libros para poder asignarlo en los préstamos; en cambio, con los scopes configurables, sólo tenemos que tener uno para todos los libros. El nuevo scope tendría una forma similar al siguiente:

```
{  
_id: "borrow:book",  
audience: "http://resources.bqws.io",  
rules: [  
{  
mediaTypes: [  
  
"application/json"  
],  
methods: [  

```

```
"GET"
],
type: "http_access",
uri: "v.*/resource/books:Book/"
}
],
scopes: [],
parameters: {
resourceId: "[A-Za-z]+"
}
}
```

Scopes básicos por servicios:

Cada uno de los módulos ofrecidos por corbel tiene una funcionalidad susceptible de ser controlada por diferentes Scopes. A continuación se muestran los Scopes que podríamos denominar como genéricos para todos los proyectos:

Módulo IAM

```
{
  "_id" : "iam:user:delete",
  "audience" : "http://iam.bqws.io",
  "rules" : [
    {
      "mediaTypes" : [
        "application/json"
      ],
      "methods" : [
        "DELETE"
      ],
      "type" : "http_access",
      "uri" : "v.*/user/.*"
    }
  ]
}
```



```
        },
        {
          "_id" : "iam:user:create",
          "audience" : "http://iam.bqws.io",
          "rules" : [
            {
              "mediaTypes" : [
                "application/json"
              ],
              "methods" : [
                "POST"
              ],
              "type" : "http_access",
              "uri" : "v.*/user/.*/identity/?"
            },
            {
              "mediaTypes" : [
                "application/json"
              ],
              "methods" : [
                "PUT",
                "POST",
                "PUT"
              ],
              "type" : "http_access",
              "uri" : "v.*/user(/.+)?"
            }
          ]
        },
        {
          "id" : "iam:user:update",

```

```
"audience" : "http://iam.bqws.io",
"rules" : [
{
"mediaTypes" : [
    "application/json"
],
"methods" : [
    "PUT"
],
"type" : "http_access",
"uri" : "v.*/user/.*"
}
]
},
{
"_id" : "iam:user:me",
"audience" : "http://iam.bqws.io",
"rules" : [
{
"mediaTypes" : [
    "application/json"
],
"methods" : [
    "GET"
],
"type" : "http_access",
"uri" : "v.*/user/me"
}
]
},
{
"_id" : "iam:user:read",
```

```
"audience" : "http://iam.bqws.io",
"rules" : [
{
"mediaTypes" : [
    "application/json"
],
"methods" : [
    "GET"
],
"type" : "http_access",
"uri" : "v1.*/user/?(!me$).*"
}
]
},
{
"_id" : "iam:token:upgrade",
"audience" : "http://iam.bqws.io",
"rules" : [
{
"mediaTypes" : [
    "application/json"
],
"methods" : [
    "GET"
],
"type" : "http_access",
"uri" : "v.*/oauth/token/upgrade/?"
}
]
}
```

Módulo EC

124

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289

Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa
Operativo Plurirregional de Crecimiento Inteligente 2014-2020

```
{  
  "_id" : "ec:product",  
  "audience" : "http://ec.bqws.io",  
  "rules" : [  
    {  
      "mediaTypes" : [  
  
        "application/json"  
      ],  
      "methods" : [  
  
        "GET",  
  
        "POST",  
  
        "PUT",  
  
        "DELETE"  
      ],  
      "type" : "http_access",  
      "uri" : "v.*/product/?.*"  
    }  
  ]  
},  
{  
  "_id" : "ec:order",  
  "audience" : "http://ec.bqws.io",  
  "rules" : [  
    {  
      "type" : "http_access",  
      "mediaTypes" : [  
  
        "application/json"  
      ],  
    }  
  ]  
};
```

```
"methods" : [  
  
    "GET",  
  
    "POST",  
  
    "PUT",  
  
    "DELETE"  
,  
    "uri" : "v.*/order/",  
    "tokenType" : "user"  
}  
]  
},  
{  
    "_id" : "ec:purchase:user",  
    "audience" : "http://ec.bqws.io",  
    "rules" : [  
    {  
        "mediaTypes" : [  
  
            "application/json"  
        ],  
        "methods" : [  
  
            "GET",  
            "PUT"  
        ],  
        "type" : "http_access",  
        "uri" : "v.*/purchase(?:/-.*)?/?$",  
        "tokenType" : "user"  
    }  
]
```

```
},
{
  "_id" : "ec:purchase:admin",
  "audience" : "http://ec.bqws.io",
  "rules" : [
    {
      "mediaTypes" : [
        "application/json"
      ],
      "methods" : [
        "GET",
        "PUT"
      ],
      "type" : "http_access",
      "uri" : "v.*/purchase/.*",
      "tokenType" : "user"
    }
  ],
  "},
  {
    "_id" : "ec:paymentmethod:user",
    "audience" : "http://ec.bqws.io",
    "rules" : [
      {
        "mediaTypes" : [
          "application/json"
        ],
        "methods" : [
          "GET",

```

```
"POST",
"DELETE"
],
"type" : "http_access",
"uri" : "v.*/paymentmethod(:?/-.*)?/?$",
"tokenType" : "user"
}
]
},
{
"_id" : "ec:paymentmethod:admin",
"audience" : "http://ec.bqws.io",
"rules" : [
{
"mediaTypes" : [
"application/json"
],
"methods" : [
"GET",
"POST",
"DELETE"
],
"type" : "http_access",
"uri" : "v.*/paymentmethod/.*",
"tokenType" : "user"
}
]
},
{
}
```

```
"_id" : "ec:payment:user",
"audience" : "http://ec.bqws.io",
"rules" : [
{
"mediaTypes" : [
    "application/json"
],
"methods" : [
    "GET"
],
"type" : "http_access",
"uri" : "v.*/payment(?:/-.*)?/?$"
}
]
},
{
"_id" : "ec:payment:admin",
"audience" : "http://ec.bqws.io",
"rules" : [
{
"mediaTypes" : [
    "application/json"
],
"methods" : [
    "GET"
],
"type" : "http_access",
"uri" : "v.*/payment/.*"
}
]
```

Módulo EVCI

```
{  
  "_id" : "evci:event:publish",  
  "audience" : "http://evci.bqws.io",  
  "rules" : [  
    {  
      "mediaTypes" : [  
  
        "application/json"  
      ],  
      "methods" : [  
  
        "POST"  
      ],  
      "type" : "http_access",  
      "uri" : "v.*/event/.*"  
    }  
  ]  
}
```

Autenticación

Para hacer uso de cualquier servicio de corbel, es necesario obtener un accessToken, para ello, la aplicación debe autenticarse usando el servicio IAM.

Pero para obtener un accessToken de IAM es necesario crear un JWT (JSON Web Token)

Crear un JWT

JWT es un estándar que permite, de una forma segura, expresar la identidad y las necesidades que tiene el cliente que solicita la autorización de acceso. NO se recomienda construir el JWT a mano porque, aunque es muy sencillo, es posible cometer errores en la codificación y firmado que nos lleven a la frustración. Se recomienda el uso de alguna librería que permita hacer esto con un par de líneas de código.

Para construir un JWT, se necesitan dos JSON:

- El primero es el header del JWT, que en nuestro caso puede ser:

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```

lxxiv

- El siguiente JSON es el payload o claims del JWT que para nuestro caso sería:

```
{  
  "iss": "fb2a979d",  
  "aud": "http://iam.bqws.io",  
  "scope": "evci:event:publish resources:sap:message",  
  "exp": "TIMESTAMP_NOW_PLUS_EXPIRES  
}
```

iss: Issuer: El identificador del cliente o clientId

aud: Audience, A quién va dirigido este JWT, en este caso para http://iam.bqws.io

scope: Scopes que el cliente solicita tener acceso

exp: Tiempo de validez para el accessToken solicitado representado en epoch_time actual en segundos más el tiempo de validez (máximo 1h).

El JWT finalmente se firma con el algoritmo HMAC SHA256 (o RS256) con secret como clave para generar el assertion:

```
base64Encode(header) + "." + base64Encode(payload) + "." + sign(base64Encode(header)  
+ "." + base64Encode(payload), clientSecret)
```

Este assertion es el que se envía como cuerpo al solicitar un accessToken junto al grant_type. El grant_type siempre toma el valor deurn:ietf:params:oauth:grant-type:jwt-bearer.

```
POST https://iam.bqws.io/v1.0/oauth/token  
Accept:application/json  
Content-Type:application/x-www-form-urlencoded; charset=UTF-8
```

Form Data (parsed and decoded data)
assertion:_JWT_ASSERTION_
grant_type:urn:ietf:params:oauth:grant-type:jwt-bearer

Como salida se obtiene lo siguiente:

Más info

- IETF
- JWT playground

En el siguiente apartado se pasa a explicar cómo se obtiene un accessToken en corbel.

Cómo obtener un accessToken

A continuación se describen los diferentes métodos para obtener un accessToken, ya sea a través de autenticación básica con IAM, o a través del protocolo OAuth (facebook, google o corbel-oauth).

Token Object

TokenObject es la estructura de datos que representa un accessToken, este está compuesto por:

- **accessToken**: El token propiamente dicho que da acceso a otros recursos.
- expiresAt: El tiempo hasta el cual el token es válido.
- **refreshToken**: El token a usar para obtener un tokenObject sin tener que generar un JWT completo.

```
{  
  
    "accessToken": "valid-token",  
    "expiresAt": 1385377605000,  
    "refreshToken": "refresh-token"  
}
```

Básica (IAM Basic)

Para autenticarse en IAM directamente

Dado el siguiente `_JWT_ASSERTION_` generado con el siguiente código JavaScript:

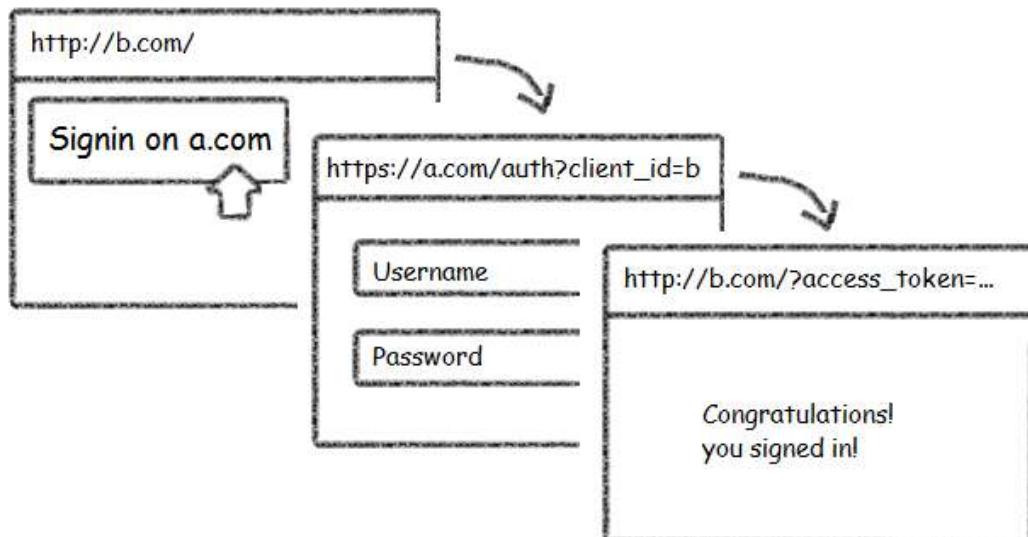
```
var claims = {  
    iss: 'clientId',           // App identifier  
    secret: 'clientSecret',    // App secret  
    aud: 'aud': 'http://iam.bqws.io',  
    scope: 'scope:operation1 scope:operation2',  
    version: '0.0.1'           // client app version  
    claims.exp: getExpiration(), // Current millis timestamp plus 1h  
    'basic_auth.username': 'username',  
    'basic_auth.password': 'password',  
    'device_id': 'deviceId'      // client device identifier  
};  
  
var _JWT_ASSERTION_ = jwt.generate(claims);
```

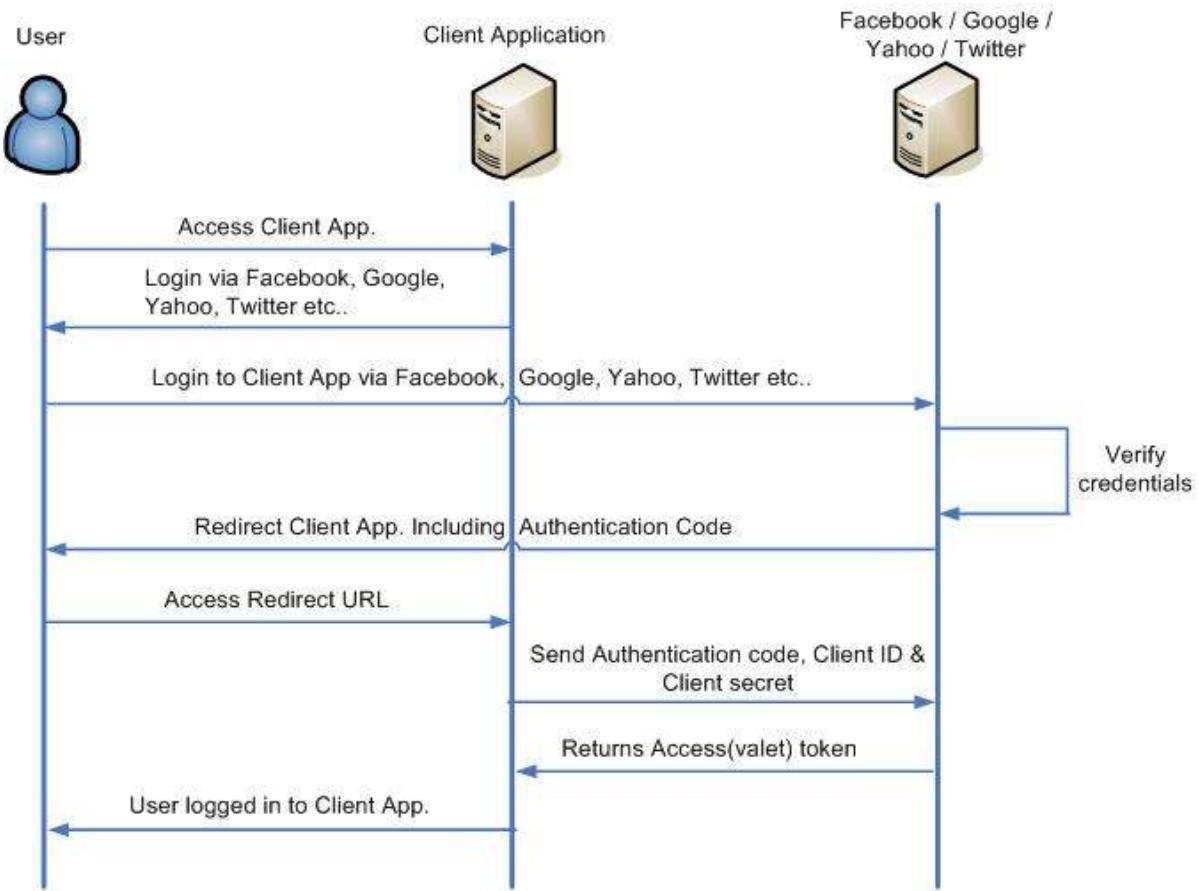
Se puede obtener un `accessToken` directamente con IAM a través de la siguiente petición:

```
curl 'https://iam.bqws.io/v1.0/oauth/token'  
-H 'Content-Type: application/x-www-form-urlencoded; charset=UTF-8'  
-H 'Accept: application/json'  
--data "assertion=_JWT_ASSERTION_&grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer"
```

Basada en oauth

OAuth es un protocolo open source, que permite autorización segura de una API de modo estándar y simple para aplicaciones de escritorio, móviles y web. OAuth permite a un usuario del sitio A compartir su información en el sitio A (proveedor de servicio) con el sitio B (llamado consumidor) sin compartir toda su identidad.





1. Crear credenciales de aplicación en el servicio externo

- Crear aplicación en Facebook [Pendiente]
- Crear aplicación en Google [Pendiente]
- Crear aplicación en corbel-js [Pendiente]

1. Redirigir al servicio de oauth externo (Facebook, Google, corbel-oauth)

- Para el caso de google, abrimos una nueva ventana en la siguiente URL:
<https://accounts.google.com/o/oauth2/auth?queryString>
- En queryString deben aparecer los siguientes parámetros:
- clientId: Id del cliente,

- scopes: String separado por espacios con los scopes del servicio externo que se solicita (no los de corbel)
response_type: El tipo de clave del servicio externo que se solicita, en nuestro caso code, a intercambiar por un accessToken
 - redirect_uri: URL a la que redirigir cuando el proceso de autenticación en el servicio externo se complete, ha de ser exactamente la misma que la URL permitida a la hora de crear la aplicación en el servicio externo.
1. Al completarse la autenticación al servicio externo, éste devuelve el code del servicio externo para intercambiarlo por un accessToken de IAM. Esto se puede lograr de varias formas:
 - Redirect automático a IAM: Se puede establecer el anterior campo redirect_uri para que apunte directamente a IAM para intercambiar automáticamente el code por un accessToken, por ejemplo: <https://iam.bqws.io/v1.0/oauth/token>
 - Pros: sencillo y automático
 - Contras: al final del flujo termina en una url que devuelve JSON que hay que procesar de alguna forma.
 - Redirect a la aplicación cliente: También es posible apuntar a una URL de cliente para que éste procese el code devuelto y lo envíe a IAM.
 - Pros: mayor control sobre el flujo y la respuesta.
 - Contras: intercambiar el code por el token manualmente. Dar soporte a la URL del cliente (web, interceptores, ...)

Assets

Un Asset define un conjunto de scopes que un usuario puede tener durante un periodo de tiempo. Con este mecanismo, podemos definir scopes dinámicos de recursos a usuarios.

Un ejemplo de Asset sería el siguiente:

```
{  
  _id: "0b2dff5c39934c210d80056a94eb27bc6c6d6378",  
  userId: "fooid",  
  name: "assettest13",  
  productId: "producttest13",  
  expire: ISODate("2025-01-01T00:01:00Z"),  
  active: true,  
  scopes: [  
    "ec:purchase:user"  
  ]  
}
```

Incrementar scopes con assets

Una vez autenticado, para poder obtener accesos a los scopes que nos dan acceso a los assets, tenemos que realizar una petición a:

```
GET https://assets.bqws.io/v1.0/asset/access  
Accept:application/json  
Authorization: Bearer accessToken
```

Esta petición redirigirá automáticamente a IAM para obtener un accessToken con accesos incrementados (en función de los assets).

Resources API

La API de recursos en corbel está implementada sobre una interfaz muy flexible que permite servir cualquier tipo de representación de un recurso. Usando los patrones definidos por esta API, se puede desplegar cualquier tipo de recurso con un impacto mínimo sobre los clientes y servidor.

Versionado

El primer nivel de todas las URL de la API es siempre la versión de ésta última, y tiene la forma v{número}. Todas las URLs contenidas en este documento se expresan partiendo de la versión como raíz

<https://resources-staging.bqws.io/v1.0/resource/media:Track/555>

URIs: completas vs forma canónica

Todo recurso en la API se identifica a través de su URI. En corbel, por razones prácticas, siempre utilizamos la forma canónica de la URI (canonical form), formada por namespace:id, en lugar de la URI ‘completa’ (fully qualified name). Los namespaces se definen mediante la propia API y los clientes los pueden consultar en /api/namespaces

Ejemplo namespaces

```
{  
  "media": "http://ontology.bqreaders.com/media/",  
  "product": "http://ontology.bqreaders.com/product/",  
  "entity": "http://ontology.bqreaders.com/entity/",  
  "api": "http://ontology.bqreaders.com/api/"  
}
```

Con estos namespaces podríamos crear las siguientes formas canónicas

URI	Canonical form
http://ontology.bqreaders.com/media/Album	media:Album
http://ontology.bqreaders.com/product/Cellphone/aquaris5	product:Cellphone/aquaris5
http://ontology.bqreaders.com/entity/Publisher/planeta	entity:Publisher/planeta
http://ontology.bqreaders.com/api/limit	api:limit

URI templates

Corbel cuenta únicamente con 3 formas diferentes de URL

- **Collection:** conjunto de recursos. URI template: /resource/{collection}

/resource/media:Track
/resource/product:Cellphone
/resource/entity:Artist

- **Resource:** recurso particular. URI template: /resource/{collection}/{resource_id}

/resource/media:Track/555
/resource/product:Cellphone/aquaris5
/resource/entity:Artist/123

- **Relation:** relación entre un recurso particular y una colección. URI template: /resource/{collection}/{resource_id}/{relation_collection} Para referirse a un recurso de la relación en particular, se hace mediante un matrix param “r” con el template {relation_collection}/{resource_id}

/resource/media:Album/456/media:Track
/resource/media:Book/555/media:Author
/resource/media:Videogame/asdf/media:Related;r=media:Related/123

Parámetros de la petición

La petición puede contener parámetros que modifican la representación de la respuesta. Dichos parámetros deben ser especificados mediante su forma canónica.

Parámetros permitidos

Option	Description	Possible Values	Example
api:pageSize	Limits the number of resources returned from a collection in one page	Positive Integer	/resource/entity:Artist/456/album

api:page	The page of the results returned from a collection	Positive Integer	/resource/entity:Artist/456/album
api:sort	Sorts the resources returned in the direction given	JSON String	/resource/media:Artist?api:sort=
api:query	Query for a collection of resources. Using the query language described by this document	JSON Query String	/resource/media:Artist?api:query=Killers”}
api:aggregation	Do aggregation operations with the resources	JSON Aggregation String	/resource/media:Artist?api:aggre

API query language

El parámetro api:query permite al cliente realizar filtros en las búsquedas. Dicho parámetro debe ser especificado como JSON Array, no vacío, donde cada elemento es un filtro que se aplicará en la llamada. Cada filtro, a su vez, es un JSON Object que tampoco puede estar sin contenido.

Filtros

El filtro tiene la siguiente forma:

```
{
  operador: {
    campo : valor
  }
}
```

Operador

El operador representa el filtro que se va a aplicar. Comienzan siempre con el carácter \$.

Operator	Work with	Description
----------	-----------	-------------

\$eq	primitive	Check if the field and the value are equals
\$gt	primitive	If the field is greater than the value
\$gte	primitive	If the field is greater or equal than the value
\$lt	primitive	if the field is less than the value
\$lte	primitive	if the field is less or equal than the value
\$ne	primitive	if the field is not equal to the value
\$in	array	if the field is in the array of values
\$all	array	if the field is the same array than the values
\$like	primitive	Matches if the field is like the value (case insensitive). It admits a regex in the value
\$elem_match	array	Matches if any of the elements contained in the filed array has a field that matches the given expression: {\$elem_match: {field:[{expression}]}}

Campo

El campo representa un atributo de la entidad de la colección sobre la que se está aplica el filtro.

Valor

Representa el valor a filtrar. Consideramos 3 tipos de valor: primitivo, array de primitivos y array de filtros* (solo para \$elem_match).

Tipos permitidos

- number
- string
- boolean
- date: con el formato definido en la ISO 8601. Para poder diferenciar un campo date de un campo string, el valor debe tener la forma “ISODate(date)”.
- period: con el formato definido en la ISO 8601, pero solo soporta periodos a nivel de año, mes y día. Para poder diferencia un campo period de un campo string, el valor debe tener la forma “Period(period)”.

- array

Ejemplos de valor
2
3.5
"any string"
true
"ISODate(2012-07-14T01:00:00+01:00)"
"Period(P1Y1D)"
[2, 3, 4]

Ejemplos de filtro

```
api:query=[{"$gt":{"releaseDate":"ISODate(2013-01-01T00:00:00Z)"}]}  
api:query=[{"album":"Octavarium"}, {"$lt":{"duration":10}}]  
api:query=[{"$in":{"categories":["pop","rock"]}}]  
api:query=[{"$like":{"artist":"killers"} }]  
api:query=[] //Bad request, empty array  
api:query=[{"album":"Scenes form a Memory"},{} ] //Bad request, empty filter object.  
api:query=[{"album":"Scenes Form a Memory", "artist":"Dream Theater"}] //Bad request, multiple attribute on a filter.  
api:query=[{"album":"Scenes Form a Memory", {"artist":"Dream Theater"} } ] //Good  
api:query=[{"$elem_match":{"albums":[{"$in":{"categories":["pop","rock"]}}]} } ]
```

API aggregation language

El parámetro api:aggregation permite al usuario realizar operaciones de agregación en los listados. Está formado por un JSON Object que contiene un operador y un parámetro:

```
{  
operador: param  
}
```

Operator	Parameter	Description	Return
\$count	“*”	Count all the elements of the request	{count : n}
\$count	field: String	Count all the elements of the request that contain the field “field”	{count : n}
\$avg	field: String	Average the parameter field of the elements of the request	{average: n}
\$sum	field: String	Sum the parameter field of the elements of the request	{sum: n}

Ejemplos de agregación

```
api:aggregation={"$count":"*"}  
api:aggregation={"$count":"field"}  
api:aggregation={"$avg":"field"}  
api:aggregation={"$sum":"field"}
```

Negociación de contenido

En ocasiones, es posible que existan diferentes representaciones de un mismo recurso. Siguiendo los standards HTTP, un cliente puede obtener la representación que desee mediante el uso de la cabecera Accept en la petición. En el caso de que el cliente no especifique la representación mediante esta cabecera, el servidor siempre tratará de devolver la representación que tiene establecida por defecto: application/json.

Cuando existen múltiples representaciones de un recurso, el servidor devolverá la cabecera Alternates indicando las representaciones disponibles.

Ejemplo de negociación

El siguiente ejemplo muestra cómo obtener las representaciones disponibles de un determinado recurso de audio

lxxxiii

- Client request

HEAD /resource/media:Track/555 HTTP/1.1

Accept: audio/*

lxxxiv

- Server response

HTTP/1.1 300 Multiple Choices

Alternates: {"/resource/media:Track/555" 1.0 {type audio/mp3}}, {"/resource/media:Track/555" 0.7 {type audio/audio/x-ms-wmal}}

Vary: negotiate, accept

lxxxv

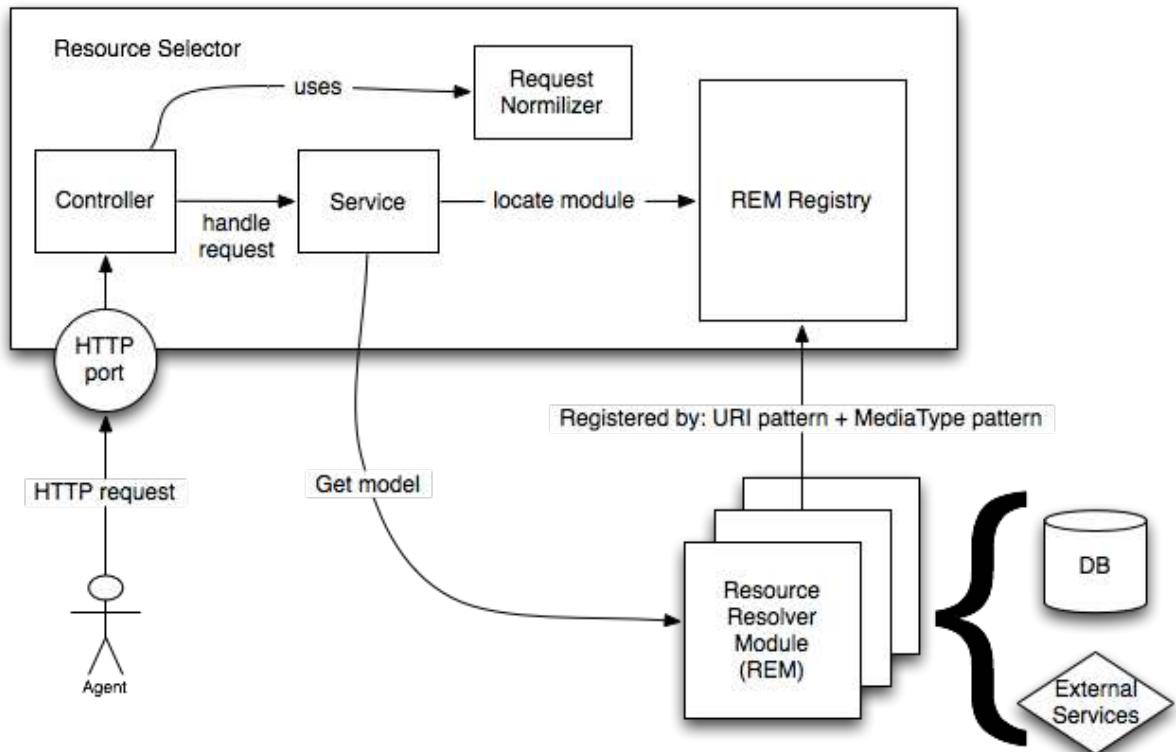
- Client request

GET /resource/media:Track/555 HTTP/1.1

Accept: audio/mp3

Implementación

La implementación de la API de recursos se basa en una arquitectura modular que permite la extensibilidad de la API a nuevos dominios mediante un sistema de plugins.



Cuando llega una petición, el servicio recupera un Resource Resolver Module (REM) del registro, que será quien finalmente resuelva la petición, accediendo a base de datos y/o servicios externos.

La selección del REM se realiza en base a la URI + MediaType + HTTP Method. En el caso del MediaType, se escoge aquel REM que sea más específico, por ejemplo, si el cliente en la petición indica image/jpeg, y hay 2 REM candidatos, uno registrado con image/* y el otro con image/jpeg, se escogerá éste último.

Cada plugin puede registrar tantos REM como requiera.

REM

Hay 3 operaciones que puede implementar un rem: collection, resource y relation. Cada plugin REM implementará aquellas funciones necesarias para la resolución de sus recursos, es decir, no todos los REM tienen por qué implementar las 3 operaciones.

Existen 3 plugin genéricos:

lxxxvi

- RESMI. Resuelve todas las peticiones cuyo MediaType sea application/json.
- RESTOR. Resuelve todas las peticiones GET, PUT y DELETE de cualquier MediaType. Este plugin se encarga de gestionar los binarios.
- IMAGE. Resuelve aquellas peticiones GET cuyo MediaType sea image/*. Este plugin se usa para hacer escalados de imágenes.

Conceptos básicos: Eventos asíncronos

EVCI (EVents Common Interface) es un módulo de corbel para escuchar eventos de forma asíncrona. Esto es útil cuando al cliente no le interesa esperar al procesamiento de una petición para recibir respuesta. También cuando se produce un evento muchas veces de forma simultánea, ya que evci utiliza un sistema de colas para no bloquear la api durante el procesamiento de los eventos, y asegura que el orden de llegada de los eventos será el orden en el que serán atendidos.

El comportamiento para responder a esos eventos es fácilmente extensible a través de Eworkers, que son plugins programados en Java.

Los eventos no son más que un json con formato libre, que deberá atenerse al eworker que lo vaya a escuchar.

EVCI API

La api de evci se compone de un solo endpoint que se corresponde a la acción de enviar evento. Los detalles a cerca del endpoint pueden consultarse en apiary.

Un ejemplo de petición a evci podría ser el siguiente:

POST <https://evci.bqws.io/v1.0/event/log:ClientLog>

body: {

146

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289

Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020

```
"client": {"browser": "Chrome", "version": "36.0.1985.143", "os": "Linux x86_64", "screen_width": 1366, "so": "silkroad-tests", "app_version": "1.0.0"},  
"logs": [  
    {"timestamp": 1410774700033, "level": "INFO", "message": "My log message!!"},  
  
    {"timestamp": 1410774700054, "level": "ERROR", "message": "ERROR!!!"}  
]
```

Donde 'log:ClientLog' es el tipo de evento enviado y el body el contenido en si del evento.

Si el json está bien formado y cumple el formato que espera el eworker, EVCI siempre devolverá un HTTP status 202 (accepted), indicativo de que la petición ha sido aceptada y se procesará eventualmente.

Versionado

El primer nivel de todas las URL de Corbel es siempre la versión de ésta última, y tiene la forma v{número}.

<https://evci-staging.bqws.io/v1.0/event/test:Event>

Autorización

Como todas las api's de corbel, evci necesita de autorización. Para ello, hay que enviar un access token de iam en la cabecera Authorization. Más información sobre como conseguir un access token en la documentación de IAM.

Eworkers

Los eworkers son los plugins de EVCI. Son programados en java y tienen que implementar la eworker-api de la que provee Corbel. Una vez programado, el eworker se registra para un tipo de evento, que es el path param del endpoint de EVCI. Una vez registrado, este eworker será el que procese los eventos de su tipo, y será responsable de validar el formato del evento.

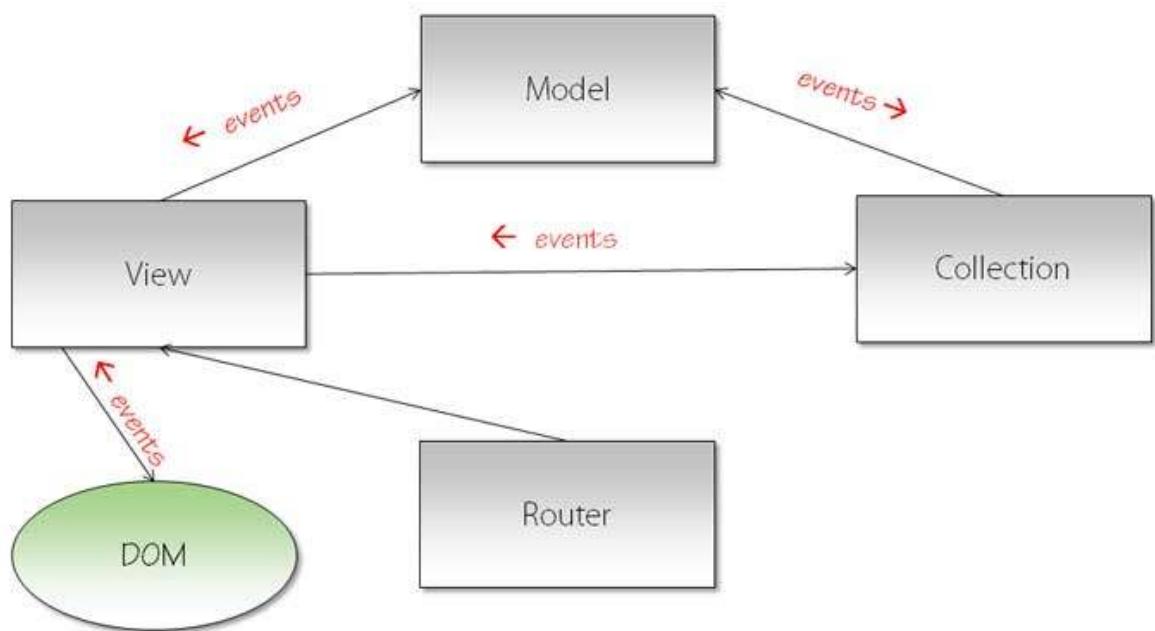
Un ejemplo de eworker es el logs-eworker, que es el encargado de procesar y almacenar los logs de los clientes. La información de esta api se encuentra en apiary.

5.2 Anexo II. CoreJS

Se llama coreJS Base al conjunto de tecnologías y módulos con el que se creó la primera versión de Bitbloq.

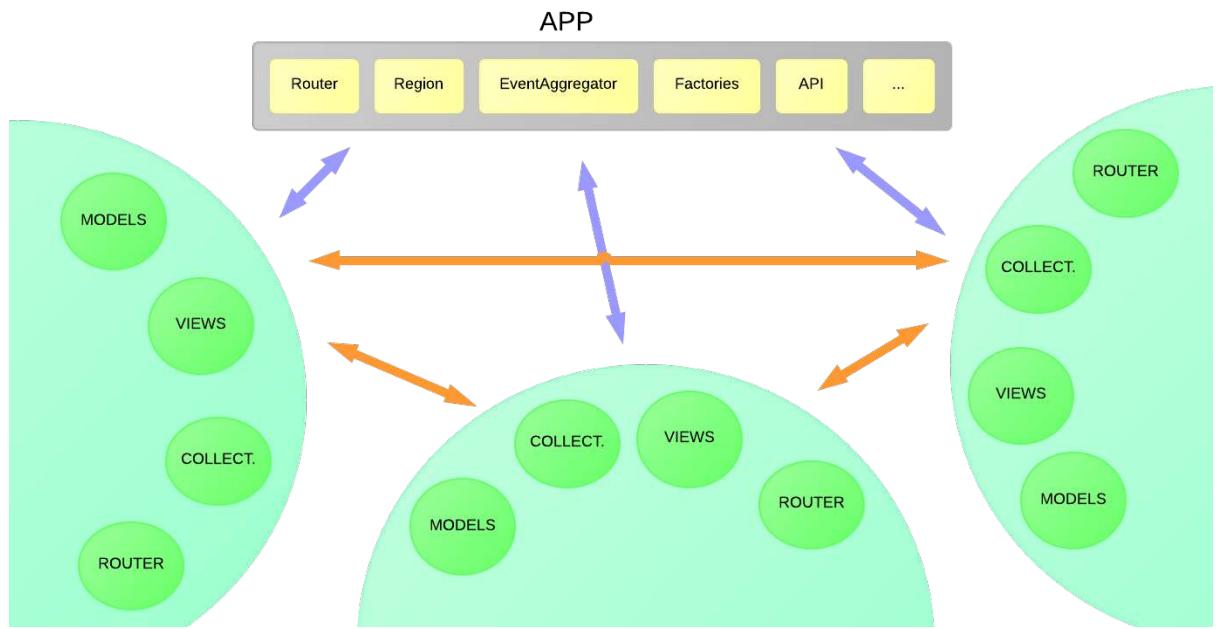
Componentes Principales

Los elementos básicos de una aplicación basada en CoreJS/Backbone son los siguientes:



Arquitectura de aplicación

La arquitectura en coreJS se distribuyen de la siguiente forma:



Repositorios

Este proyecto está alojado en los siguientes repositorios GIT:

lxxxvii

- **coreJS Doc** El repositorio de esta misma documentación

`git@github.com:bq-devtools/corejs-doc.git`

lxxxvii

- **coreJS Boilerplate** Esqueleto de proyecto coreJS listo para el desarrollo.

`git clone ssh://git@stash.bq.com:7999/swplat/corejs-app-boilerplate.git`

lxxxvii

- **coreJS Build** En este repositorio define el proceso de construcción y otras tareas automatizadas a través de Grunt.

`git clone ssh://git@stash.bq.com:7999/swplat/corejs-build.git`

lxxxvii

- **coreJS Base** En este repositorio se guarda los módulos principales de toda webapp (localización, integración con backend, factorías, configuración, ...), así como módulos reutilizables entre aplicaciones.

```
git clone ssh://git@stash.bq.com:7999/swplat/corejs-base.git
```

lxxxvii

- **coreJS Components Boilerplate** En este repositorio define un arquetipo que almacena los módulos de coreJS que pueden reutilizar|extender otros proyectos.

```
git clone ssh://git@stash.bq.com:7999/swplat/corejs-component-boilerplate.git
```

lxxxviii

- **coreJS [productName] Modules** En este repositorio almacena los módulos de la webapp específica en función del [productName]:

```
git@bitbucket.org:mundoreader/[productName]-corejs-app.git  
git@bitbucket.org:mundoreader/bookland-corejs-app.git  
git@bitbucket.org:mundoreader/mercurio-corejs-app.git  
git@bitbucket.org:mundoreader/orpheus-corejs-app.git
```

lxxxvii

- Los **desarrolladores de webapps** trabajarán principalmente con los repositorios de coreJS [productName] Modules, creando módulos específicos.

lxxxvii

- Los **desarrolladores del coreJS** trabajarán principalmente con el repositorio de coreJS Base

Estructura de archivos

[251C?][2500?][2500?] init-env.sh	// Script de inicialización del proyecto
[251C?][2500?][2500?] pom.xml	// Maven stuff

```
[251C?][2500?][2500?] package.json           // Dependencias de desarrollo (npm)
[251C?][2500?][2500?] bower.json             // Dependencias con librerías externas
[251C?][2500?][2500?] README.md              // Este documento
[251C?][2500?][2500?] target                // Workspace de temporales de Grunt
[251C?][2500?][2500?] src/test               // Workspace de test
[2514?][2500?][2500?] src/main/webapp        // Workspace de desarrollo de Grunt
    [251C?][2500?][2500?] stylesheets          // Hojas de estilo SASS
[2514?][2500?][2500?] scripts
[251C?][2500?][2500?] app.js                 // Contexto de aplicación
[251C?][2500?][2500?] main.js                // Script que arranca la aplicación
[251C?][2500?][2500?] engine                 // Componentes internos de la aplicación
[2502?]  [251C?][2500?][2500?] services.js // Comunicación con Backend
[2502?]  [251C?][2500?][2500?] common.js   // Parámetros de la app
[2502?]  [251C?][2500?][2500?] factory.js  // Factoría de objetos
[2502?]  [251C?][2500?][2500?] locale.js   // Módulo de i18n
[2502?]  [251C?][2500?][2500?] logger.js   // Módulo de log
[2502?]  [251C?][2500?][2500?] polyfills.js// Polyfill varios
[2502?]  [251C?][2500?][2500?] utils.js    // Funciones varias
[2502?]  [251C?][2500?][2500?] user.js     // Autenticación y gestión de usuario
[2502?]  [251C?][2500?][2500?] resource.js // Librería para realizar peticiones a
backend
[2502?]  [251C?][2500?][2500?] session.js // Gestión de los datos de sesión del
usuario
[2502?]  [251C?][2500?][2500?] router.js   // Enrutador de la webapp
[2502?]  [251C?][2500?][2500?] ...
[2502?]  [2514?][2500?][2500?] start.js   // Inicializador de módulo
[2514?][2500?][2500?] vendor            // Librerías externas ajenas a Bower
```

Configuración

La aplicación tiene un documento de parámetros y constantes por defecto en el repositorio de la aplicación con nombre corejs-[projectName]-app, en src/main/webapp/res/config/config.js, que permiten la configuración de la aplicación, así como su fácil acceso para el resto de los módulos.

A continuación se explican cada uno de los parámetros.

Clave	Descripción	Tipo	Obligatorio
mode	Establece el entorno general de la aplicación, ofrece la oportunidad a los desarrolladores de cambiar el comportamiento de la aplicación dependiendo del entorno (formularios pre-cumplimentados, mocks, stubs, etc).	String	no
version	Establece la versión pública de la aplicación. Nos permite enviar la versión con la que se están generando las trazas para el logToServer. Debería seguir las convenciones establecidas en semver.org .	String (0.0.1)	sí
appName	Establece el nombre interno de la aplicación web. Nos permite identificar qué aplicación realiza qué operaciones.	String (corejs-app)	no
clientType	Establece el tipo de cliente. Normalmente se usa para backends que ofrecen servicios a diferentes tipos de cliente y cada uno puede tener necesidades específicas.	String (WEB)	no
lang	Establece el idioma por defecto	String (es-ES)	no
logLevel	Establece el nivel de log que se desea registrar: 0: trace, 1: debug, 2: info, 3: warn, 4: error, 5: silent	Integer (0)	no
logBuffer	Establece la cantidad de logs a almacenar hasta que se envía al servidor	Integer (10)	no

logToServer	Establece el nivel de log que se desea enviar al servidor: 0: trace, 1: debug, 2: info, 3: warn, 4: error, 5: silent	Integer (2)	no
logServerEndpoint	Establece la ruta del servidor a donde enviar los logs	String (end-point/)	no
resourcesEndpoint	Establece la ruta base hasta los recursos del backend.	String	sí
evciEndPoint	Establece la ruta base hasta el registro de eventos del backend.	String	sí
ecEndpoint	Establece la ruta base hasta la gestión del E-Commerce del backend.	String	sí
oauthEndpoint	Establece la ruta base hasta el servidor de OAuth para la autenticación.	String	sí
oauthClientId	ClientId proporcionado por el proveedor del servicio de OAuth correspondiente.	String	sí
oauthSecret	SecretId proporcionado por el proveedor del servicio de OAuth correspondiente.	String	sí
oauthService	Nombre del servicio de OAuth.	String	sí
iamEndpoint	Establece la ruta base hasta el servidor de IAM.	String	sí
clientId	ClientId proporcionado por el proveedor del servicio de IAM.	String	sí
clientSecret	SecretId proporcionado por el proveedor del servicio de IAM correspondiente.	String	sí
claimAud	Valor de la variable aud para la generación del JWT .	String	sí
claimGrantType	Valor de la variable grant-type para la generación del JWT .	String	sí

claimScopes	Valor de la variable scopes para la generación del JWT para una sesión* anónima.	String	sí
claimScopesUser	Valor de la variable scopes para la generación del JWT para usuarios autenticados.	String	sí
claimScopesRegister	Valor de la variable scopes para la generación del JWT para registrar usuarios nuevos En formato Array!.	String	sí
claimExp	Valor de la variable exp para la generación del JWT .	Integer (3500 (segundos))	no
grantType	Valor de la variable grant-type para la generación del JWT para registrar usuarios nuevos.	String	sí
jwtAlgorithm	Valor de la variable alg para la generación del JWT para registrar usuarios nuevos.	String	sí
autoTokenRefresh	Activa el autorefresco de token cuando una petición que requiere autenticación falla por 401.	Boolean (false)	no

Además la aplicación se puede configurarse de varias formas, estableciendo los parámetros en el código a través de una variable global de configuración CFG, o en tiempo de ejecución a través del archivo app/res/config/config.json.

window.CFG

Se puede establecer la configuración de la aplicación definiendo los parámetros en una variable global que deberá llamarse CFG. Esta variable global debe de estar disponible antes de que se cargue cualquier JavaScript de la aplicación. Unos ejemplos:

Producción (sin logToServer)

```
varCFG ={
  "version": "0.1.0",
  "appName": "corejs-app",
```

```
"evciEndpoint": "http://evci-int.bqws.io/v1.0/",  
  
"resourcesEndpoint": "http://resources.int.bqws.io/v1.0/",  
  
"oauthEndpoint": "http://oauth.int.bqws.io/v1.0/",  
"oauthClientId": "temp-client",  
"oauthSecret": "temp-secret",  
"oauthService": "silkroad",  
"oauthGrantType": "authorization_code",  
  
"iamEndpoint": "http://iam.int.bqws.io/v1.0/",  
"clientId": "providedClientId",  
"clientSecret": "providedSecret",  
  
"claimAud": "http://iam.bqws.io",  
"claimGrantType": "urn:ietf:params:oauth:grant-type:jwt-bearer",  
"claimScopes": "resources:bookland:read_catalog iam:user:create iam:user:delete iam:user:read evci:event:publish",  
"claimScopesUser": "resources:bookland:read_catalog iam:user:delete iam:user:read iam:user:me evci:event:publish",  
"claimScopesRegister": ["resources:bookland:read_catalog", "iam:user:read", "iam:user:me",  
"iam:user:delete", "evci:event:publish"],  
"claimExp": "3500",  
  
"jwtAlgorithm": "HS256",  
"grantType": "urn:ietf:params:oauth:grant-type:jwt-bearer",  
  
"autoTokenRefresh": true  
};
```

Producción (con logToServer)

```
varCFG = {  
"version": "0.1.0",
```

```
"appName":"corejs-app",  
  
"logLevel":0,  
"logBuffer":20,  
"logToServer":2,  
  
"evciEndpoint":"http://evci-int.bqws.io/v1.0/",  
  
"resourcesEndpoint":"http://resources.int.bqws.io/v1.0/",  
  
"oauthEndpoint":"http://oauth.int.bqws.io/v1.0/",  
"oauthClientId":"temp-client",  
"oauthSecret":"temp-secret",  
"oauthService":"silkroad",  
"oauthGrantType":"authorization_code",  
  
"iamEndpoint":"http://iam.int.bqws.io/v1.0/",  
"clientId":"providedClientId",  
"clientSecret":"providedSecret",  
  
"claimAud":"http://iam.bqws.io",  
"claimGrantType":"urn:ietf:params:oauth:grant-type:jwt-bearer",  
"claimScopes":"resources:bookland:read_catalog iam:user:create iam:user:delete iam:user:read  
evci:event:publish",  
"claimScopesUser":"resources:bookland:read_catalog iam:user:delete iam:user:read  
iam:user:me evci:event:publish",  
"claimScopesRegister":["resources:bookland:read_catalog", "iam:user:read", "iam:user:me",  
"iam:user:delete", "evci:event:publish"],  
"claimExp":"3500",  
  
"jwtAlgorithm":"HS256",  
"grantType":"urn:ietf:params:oauth:grant-type:jwt-bearer",  
  
"autoTokenRefresh":true
```

};

Desarrollo

```
varCFG ={
  "mode": "DEVELOPER",
  "version": "0.1.0",
  "lang": "es-ES",

  "evciEndpoint": "http://evci-int.bqws.io/v1.0/",

  "resourcesEndpoint": "http://dev.resources.int.bqws.io/v1.0/",

  "oauthEndpoint": "http://dev.oauth.int.bqws.io/v1.0/",
  "oauthClientId": "temp-client",
  "oauthSecret": "temp-secret",
  "oauthService": "silkroad",
  "oauthGrantType": "authorization_code",

  "iamEndpoint": "http://dev.iam.int.bqws.io/v1.0/",
  "clientId": "providedClientId",
  "clientSecret": "providedSecret",

  "claimAud": "http://iam.bqws.io",
  "claimGrantType": "urn:ietf:params:oauth:grant-type:jwt-bearer",
  "claimScopes": "resources:bookland:read_catalog iam:user:create iam:user:delete iam:user:read
evci:event:publish",
  "claimScopesUser": "resources:bookland:read_catalog iam:user:delete iam:user:read
iam:user:me evci:event:publish",
  "claimScopesRegister": ["resources:bookland:read_catalog", "iam:user:read", "iam:user:me",
  "iam:user:delete", "evci:event:publish"],
  "claimExp": "3500",

  "jwtAlgorithm": "HS256",
  "grantType": "urn:ietf:params:oauth:grant-type:jwt-bearer",
```

```
"autoTokenRefresh":true  
};
```

config.json

Si no se define configuración por variable global como se ha mencionado antes, la aplicación tratará de bajarse la configuración definida en src/main/webapp/res/config/config.json.

Los atributos del JSON son los mismos definidos antes para la variable CFG.

Gestión de Dependencias

Las dependencias de la aplicación se gestionan con RequireJS, y el archivo donde se define el árbol de dependencias entre las librerías está definido en:

lxxxviii

- Aplicación: src/main/webapp/scripts/define.js
- Tests: src/test/scripts/define.js

Contexto de Aplicación

El contexto de aplicación hace referencia al concepto de aplicación y alberga todas las funcionalidades básicas para la gestión de una aplicación. Este objeto está basado en el objeto Application que provee **Marionette**. En particular:

lxxxix

- Nos ayuda a gestionar la inicialización de la aplicación.
- Establece las regiones principales (header, content, footer, ...).
- Facilita la integración de los diferentes módulos que componen la aplicación. (EventAggregator, RequestResponse, Commands)
- Tiene acceso a los parámetros que definen la configuración y comportamiento de la aplicación.
- Ofrece acceso al resto de módulos de la aplicación.

Obtener el contexto

Para tener acceso al contexto de la aplicación basta con encapsular nuestro código JavaScript en el siguiente esquema:

```
'use strict';
/* global define */
define([
  'corejs/app', // coreJS component
], function(app) {
  // Code here!
});
```

Inicialización de la aplicación

El proceso de inicialización de una aplicación web puede llegar a ser muy complejo, para ello, app provee mecanismos necesarios para poder definir este proceso de forma distribuida de tal forma que cada tarea específica de inicialización pueda estar definido en el módulo que le corresponda.

Una aplicación Marionette se inicializa cuando se lanza la llamada `app.start()`, y con ello, todos sus manejadores definidos.

Para añadir nuevas tareas al proceso de inicialización basta con incluir el siguiente código donde sea pertinente.

xc

- En el instante que se lanza `app.start()`:

```
app.addInitializer(function() {
  // Initialization code here
});
```

xc

- Despues de que se ejecuten todos los inicializadores:

```
app.on('initialize:after', function() {
  // Post initialization code here
})
```

});

Definir las Regiones Principales

El objeto Application de Marionette además permite definir las regiones principales de la región. En Marionette, una región define cualquier elemento del DOM desde la cual, se mostrarán vistas. Para definir regiones principales de la aplicación basta con hacer lo siguiente:

```
app.addRegions({  
header:'[data-region="header"]', // <header>  
main:'[data-region="main"]', // <main>  
footer:'[data-region="footer"]' // <footer>  
});
```

Con las regiones definidas, en otra parte del código podemos mostrar vistas dentro de estas regiones:

```
app.header.show(newHeaderView());  
app.main.show(newMainView());  
app.footer.show(newFooterView());
```

Lo realmente interesante de las regiones es que se puede llamar al método show() repetidas veces, con la certeza de que Marionette llamará al método close() en la vista existente de la región para cerrar la vista con seguridad, liberando correctamente los recursos.

Es posible definir regiones anidadas con el componente Layout de Marionette, pero eso lo explicaremos con detalle más adelante.

Módulos base

Los módulos base son aquellos que vienen incluidos en coreJS base por defecto.

Logger

El logger nos ayuda a registrar los mensajes por consola de la aplicación, establecer diferentes niveles de mensajes, y enviar información relevante a un servidor específico para posterior análisis.

Ejemplo de uso:

```
// config & params are objects
app.log.info('engine', 'Logger.setConfig', config, true, [3], 3.14);
app.log.debug('engine:config:loaded', config);
app.log.debug('api.request', params);
```

Se recomienda además seguir la siguiente convención con el fin de poder filtrar el log como explicaremos más adelante:

```
app.log.debug('moduleName', 'ClassName.methodName',[ ...]);
app.log.debug('moduleName:event:name',[ ...]);
app.log.debug('moduleName.method.message',[ ...]);
```

Niveles de Log

Los errores ordenados de mayor a menor nivel son los siguientes: * SILENT(5): Este nivel sirve para anular el log * ERROR(4): Muestra por consola los errores de la aplicación * WARN(3): Muestra las alertas de la aplicación y mensajes de mayor nivel * INFO(2): Muestra las trazas de información y mensajes de mayor nivel * DEBUG(1): Muestra mensajes de depuración y mensajes de mayor nivel * TRACE(0): Muestra trazas de desarrollo y mensajes de mayor nivel * ALL(0): Muestra todos los mensajes.

```
app.log.setLevel(app.log.level.INFO);
```

Filtrar Log

A veces es necesario filtrar el log para que muestre únicamente las trazas de log que nos interesan. Para filtrar todos los logs que no sean de una sección en particular podemos hacer:

```
app.log.filter('engine');
```

Con ésto conseguimos que sólo se muestran las trazas que comiencen por la palabra engine.

Log To Server

El log tiene la capacidad de enviar cierta información a un servidor específico, para ello, primero hay que **configurar**lo y **habilitarlo**.

```
varconfig ={
logToServer:app.log.level.INFO, // Nivel de log mínimo a enviar al servidor
logBuffer:10, // N° de trazas antes de enviar al servidor
logServerEndpoint:'url' // Ruta hasta el servidor que almacena logs
logLevel:app.log.level.WARN // Nivel de log mínimo a mostrar en cliente
}
app.log.setConfig(config);
```

Integración con Silkroad

Si se integra con Silkroad, éste automáticamente sobreescribe el comportamiento para subir logs para que haga uso del módulo de EVCI, para ello, es necesario establecer en la configuración de la aplicación res/config/config.json la clave evciEndpoint.

Opcionalmente, se puede definir un namespace donde estará disponible el log en EVCI estableciendo la variable de configuración loggerNamespace en res/config/config.json, en su defecto, EVCI usara el valor del campo app.common.get('appName').

Recomendaciones

A la hora de construir los mensajes de log, hay que tener en cuenta que si el mensaje que se construye es complejo, es una buena práctica detectar el nivel de log establecido para evitar tener que construir mensajes innecesarios, por ejemplo:

```
varcomplexLogMessageBuilder =function() {
// Complex code here
returnlogString;
}

if(app.log.getLevel() ===app.log.level.INFO) {
app.log.info(complexLogMessageBuilder());
}
```

Factory

Este módulo se encarga de gestionar los diferentes tipos de objetos de la aplicación (Model, Collection, Layout, View, ...)

// Models

```
app.factory.add('ComponentNameModel', ComponentNameModel);
app.factory.new('ComponentNameModel', options);
app.factory.singleton('ComponentNameModel', options);
app.factory.get('ComponentNameModel');
```

// Collections

```
app.factory.add('ComponentNameCollection', ComponentNameCollection);
app.factory.new('ComponentNameCollection', options);
app.factory.singleton('ComponentNameCollection', options);
app.factory.get('ComponentNameCollection');
```

// Views

```
app.factory.add('ComponentNameView', ComponentNameView);
app.factory.new('ComponentNameView', options);
app.factory.singleton('ComponentNameView', options);
app.factory.get('ComponentNameView');
```

// Layouts

```
app.factory.add('ComponentNameLayout', ComponentNameLayout);
app.factory.new('ComponentNameLayout', options);
app.factory.singleton('ComponentNameLayout', options);
app.factory.get('ComponentNameLayout');
```

Cookies

Se encarga de obtener y establecer las cookies del documento actual.

// Writing a cookie

```
app.cookies.setItem(name, value[, end[, path[, domain[, secure]]]]);
```

// Getting a cookie

```
app.cookies.getItem(name);
```

// Removing a cookie

```
app.cookies.removeItem(name[, path],domain);
```

```
// Testing a cookie
app.cookies.hasItem(name);
```

User

Este módulo es el responsable de encapsular toda la complejidad relacionada con los protocolos de autenticación, altas de usuarios y autorización de peticiones a recursos.

Registro con Corbel

Si la aplicación puede registrar nuevos usuarios en la plataforma, es posible realizarse con la siguiente llamada:

```
varparams ={
  username:'username',
  email:'email@domain.com',
  password:'password',
  firstName:'firstName',
  lastName:'lastName',
  oauthService:'silkroad'
};

app.user.register(params).then(function(data) {
  // Success code
}).fail(function(silkRoadError) {
  // Error code
});
```

Parámetros de configuración

xci

- iamEndpoint: Endpoint hasta la API de IAM de Silkroad.
- oauthEndpoint: Endpoint hasta la API de Oauth de Silkroad.
- grantType: El valor de grantType para generar el JWT.

- jwtAlgorithm: Algoritmo usado para el JWT.
- clientSecret: Clave de cliente proporcionada por Silkroad para comunicarse con él.

Registro con Google+ | Facebook

Para registrar a través de Google+|Facebook, existen 2 métodos:

xcii

- **1 Click Register:** Este método obtiene los datos de perfil necesario de Google+|Facebook para registrarlos automáticamente en IAM.

```
varparams :{
  oauthService:'google' // facebook/google
};
app.api.register(params).then(function(data) {
  // Success code
}).fail(function(error) {
  // Error code
});
```

xcii

- **2 Step register:** Este método devuelve los datos de perfil necesario de Google+|Facebook para registrarlos en un paso posterior en IAM.

```
varparams :{
  oauthService:'google' // facebook/google
};
app.api.me(params).then(function(oauthUser) {
  // Refresh form with oauthUser data
}).fail(function(error) {
  // Error code
});
```

Autenticación con Silkroad

```
varparams :{
  username:'username',
  password:'password',
  oauthService:'silkroad',
  remember:1
};
app.api.login(params).then(function(data) {
  // Success code
}).fail(function(error) {
  // Error code
});
```

Parámetros de configuración

xciii

- iamEndpoint: Endpoint hasta la API de IAM de Silkroad.
- oauthEndpoint: Endpoint hasta la API de Oauth de Silkroad.
- grantType: El valor de grantType para generar el JWT.
- jwtAlgorithm: Algoritmo usado para el JWT.
- clientSecret: Clave de cliente proporcionada por Silkroad para comunicarse con él.

Autenticación con Google+ | Facebook

```
varparams :{
  oauthService:'google',    // facebook/google
  remember:1
};
app.api.login(params).then(function(data) {
  // Success code
}).fail(function(error) {
  // Error code
});
```

});

Parámetros de configuración

xciv

- googleClientId: Id cliente de la aplicación creada en Google para webapps.
- facebookClientId: Id cliente de la aplicación creada en Facebook.

Autorización

Una vez autenticado, el módulo mantiene de forma interna la información necesaria para identificarlo, además se encarga de construir las peticiones automáticamente con la autorización necesaria.

La aplicación deberá tener una clave que lo identifica como cliente autorizado client_id, y una clave secreta que le autoriza a realizar peticiones secret, ambos parámetros deberán estar definidos en la configuración de la aplicación, ya sea en tiempo de ejecución o en tiempo de despliegue.

Con dichos parámetros establecidos, la aplicación durante su inicialización, generará automáticamente una autorización de comunicación con SilkRoad ya sea como cliente anónimo, o como un usuario específico si decidió recordar su acceso. Todo ello de forma transparente al desarrollador.

Session

Gestiona los datos de la sesión del usuario, sus credenciales de sesión y el estado global de la aplicación.

```
// Stores data in user session
var data = {key:'value'};
app.session.set('key', data);

// Retrieve data from user session
```

```
vardata =app.session.get('key', data);
// data = {key: 'value'};

// Destroy session
app.session.destroy();

/***
 * Ask for user autorization
 * return boolean object
 */
app.session.gatekeeper();

/***
 * Set an application status
 * @param {string} status - Name of the status
 * @param {boolean} active - true if active, false id disabled
 */
app.session.setStatus('new-user', true);

// Remove a specific status
app.session.removeStatus('new-user');

// Create a logged session, this method are called
// automatically with app.user.login success
app.session.logged({
access:token:'token',
oauthService:'silkroad', // silkroad/facebook/google
user:user // user object
persistent:false // persistent session (login with remember)
});
```

Resources

Este módulo es el responsable de solicitar los recursos del backend necesarios.

ios para mostrar la información necesaria, encapsulando toda la complejidad relacionada con los protocolos de autenticación, autorización de peticiones, las búsquedas avanzadas y el mapeo a los modelos de datos.

En Silkroad se pueden obtener 3 tipos de recursos:

Recurso específico

```
app.resources.resource('books:Book', 'id').get().then(function(data) {  
}).fail(function(silkRoadError) {  
});
```

xcv

- **Con Backbone**

```
varbook =app.factory.new('BookModel');  
  
book.fetch({  
resourceType:'books:Book',  
id:'9788467040203'  
}).then(function(data) {  
}).fail(function(silkRoadError) {  
});
```

Búsquedas de entidades

```
varparams ={  
query:[  
{ '$eq':{field3:true} },  
{ '$eq':{field4:true} },  
{ '$gt':{field5:'value'} },  
{ '$gte':{field5:'value'} },  
{ '$lt':{field5:'value'} },  
]
```

```
{ '$lte':{field5:'value'} },
{ '$ne':{field5:'value'} },
{ '$in':{field2:['pepe', 'juan']} },
{ '$all':{field5:['pepe', 'juan']} },
{ '$like':{field5:'value'} }
],
page:{page:1, size:5},
sort:{field1:resources.sort.ASC}
};
```

```
app.resources.collection('books:Book').get(params).then(function(data) {

}).fail(function(silkRoadError) {

});
```

xcvi

• Con Backbone

```
var books =newBackbone.Collection([], {
model:Backbone.Model,
}),
params ={
resourceType:'books:Book',
query:[
{ '$eq':{field3:true} },
{ '$eq':{field4:true} },
{ '$gt':{field5:'value'} },
{ '$gte':{field5:'value'} },
{ '$lt':{field5:'value'} },
{ '$lte':{field5:'value'} },
{ '$ne':{field5:'value'} },
{ '$in':{field2:['pepe', 'juan']} },
{ '$all':{field5:['pepe', 'juan']} },
```

```
{ '$like':{field5:'value'} }  
],  
page:{page:1, size:5},  
sort:{field1:resources.sort.ASC}  
};  
  
books.fetch(params).then(function(data) {  
  
}).fail(function(silkRoadError) {  
  
});
```

Búsquedas de entidades en base a una relación

Este tipo de búsquedas permite listar entidades en base a la relación con otra entidad existente. Por ejemplo buscar 'los autores de un libro específico'.

```
varparams ={  
query:[  
{ '$eq':{field3:true} },  
{ '$eq':{field4:true} },  
{ '$gt':{field5:'value'} },  
{ '$gte':{field5:'value'} },  
{ '$lt':{field5:'value'} },  
{ '$lte':{field5:'value'} },  
{ '$ne':{field5:'value'} },  
{ '$in':{field2:['pepe', 'juan']} },  
{ '$all':{field5:['pepe', 'juan']} },  
{ '$like':{field5:'value'} }  
],  
page:{page:1, size:5},  
sort:{field1:resources.sort.ASC}  
};
```

```
app.resources.relation('books:Book', 'id', 'author').get(params).then(function(data)  
{
```

```
}).fail(function(silkRoadError) {
```

```
});  
xcvii
```

• Con Backbone

```
varbooks =newBackbone.Collection([], {  
model:Backbone.Model,  
}),  
params ={  
resourceType:'books:Book',  
id:'id',  
rel:'author',  
query:[  
{ '$eq':{field3:true} },  
{ '$eq':{field4:true} },  
{ '$gt':{field5:'value'} },  
{ '$gte':{field5:'value'} },  
{ '$lt':{field5:'value'} },  
{ '$lte':{field5:'value'} },  
{ '$ne':{field5:'value'} },  
{ '$in':{field2:['pepe', 'juan']} },  
{ '$all':{field5:['pepe', 'juan']} },  
{ '$like':{field5:'value'} }  
],  
page:{page:1, size:5},  
sort:{field1:resources.sort.ASC}  
};
```

```
books.fetch(params).then(function(data) {
```

```
}).fail(function(silkRoadError) {  
});
```

Parámetros de configuración

xcviii

- resourcesEndpoint: Endpoint hasta la API de recursos de Silkroad.

Polyfills

En este módulo se pueden definir polyfills necesarios para que la aplicación funcione como se espera para los navegadores soportados.

Por defecto incorpora los siguientes polyfills: localStorage, sessionStorage y Date.toISOString entre otros.

Navegadores Soportados

CoreJS y sus módulos tienen como objetivo dar soporte a los siguientes navegadores:

xcix

- IE10+
- Chrome 35+
- Firefox 29+
- Safari 7+
- Opera 20+

5.3 Anexo III. Ejemplo de xml de un proyecto de Bitbloq 1

```
<xml xmlns="http://www.w3.org/1999/xhtml">
    <block type="bq_bluetooth_def" id="3202" inline="false" x="-225" y="-183">
        <field name="TOGGLE">FALSE</field>
        <value name="BAUD_RATE">
            <block type="math_number" id="3203">
```

```
<field name="NUM">38400</field>
</block>
</value>
<value name="PIN">
  <block type="math_number" id="3204">
    <field name="NUM">10</field>
  </block>
</value>
<value name="PIN2">
  <block type="math_number" id="3205">
    <field name="NUM">11</field>
  </block>
</value>
<next>
  <block type="procedures_callnoreturn" id="3206">
    <mutation name="RECIBIR" />
    <field name="PROCEDURES">RECIBIR</field>
    <next>
      <block type="base_delay" id="3207" inline="true">
        <value name="DELAY_TIME">
          <block type="math_number" id="3208">
            <field name="NUM">500</field>
          </block>
        </value>
      <next>
        <block type="procedures_callnoreturn" id="3209" disabled="true">
          <mutation name="ENVIAR" />
          <field name="PROCEDURES">ENVIAR</field>
          <next>
            <block type="base_delay" id="3210" inline="true" disabled="true">
              <value name="DELAY_TIME">
                <block type="math_number" id="3211">
                  <field name="NUM">500</field>
                </block>
              </value>
            </block>
          </next>
        </block>
      </next>
    </block>
  </next>
</block>
```

```
</value>
<next>
<block type="variables_global" id="3212" inline="false">
<field name="VAR">StopI</field>
<value name="VALUE">
<block type="math_number" id="3213">
<field name="NUM">10</field>
</block>
</value>
<next>
<block type="inout_digital_write" id="3214" inline="true">
<field name="STAT">HIGH</field>
<value name="PIN">
<block type="pin_digital" id="3215">
<field name="PIN">13</field>
</block>
</value>
<next>
<block type="variables_global" id="3216" inline="false">
<field name="VAR">ir_dcho</field>
<value name="VALUE">
<block type="bq_infrared" id="3217" inline="false">
<value name="PIN">
<block type="pin_digital" id="3218">
<field name="PIN">2</field>
</block>
</value>
</block>
</value>
<next>
<block type="variables_global" id="3219" inline="false">
<field name="VAR">ir_izdo</field>
<value name="VALUE">
```



```
</next>
</block>
<block type="procedures_defnoreturn" id="3224" x="455" y="-16">
<mutation />
<field name="NAME">ENVIAR</field>
<comment pinned="false" h="68" w="437">Si pulsamos el botón, enviaremos por bluetooth una 'B'</comment>
<statement name="STACK">
<block type="serial_available" id="3225">
<statement name="DO">
<block type="bq_bluetooth_send" id="3226" inline="false">
<value name="SNT">
<block type="serial_read" id="3227" />
</value>
<next>
<block type="base_delay" id="3228" inline="true">
<value name="DELAY_TIME">
<block type="math_number" id="3229">
<field name="NUM">1000</field>
</block>
</value>
</block>
</next>
</block>
</statement>
</block>
</statement>
</block>
<block type="procedures_defnoreturn" id="3230" x="470" y="448">
<mutation />
<field name="NAME">Modo_siguelineas</field>
<statement name="STACK">
<block type="serial_println" id="3231" inline="false">
<value name="CONTENT">
```

```
<block type="variables_get" id="3232">
    <field name="VAR">ir_dcho</field>
</block>
</value>
<next>
<block type="controls_if" id="3233" inline="false">
    <mutation else="1" />
    <value name="IF0">
        <block type="logic_compare" id="3234" inline="true">
            <field name="OP">EQ</field>
        <value name="A">
            <block type="variables_get" id="3235">
                <field name="VAR">ir_dcho</field>
            </block>
        </value>
        <value name="B">
            <block type="variables_get" id="3236">
                <field name="VAR">Negro</field>
            </block>
        </value>
        </block>
    </value>
    <statement name="DO0">
        <block type="servo_cont" id="3237" inline="false">
            <field name="ROT">0</field>
            <value name="PIN">
                <block type="pinDigital" id="3238">
                    <field name="PIN">9</field>
                </block>
            </value>
            <value name="DELAY_TIME">
                <block type="math_number" id="3239">
                    <field name="NUM">10</field>
                </block>
            </value>
        </block>
    </statement>
</block>
```

```
</value>
</block>
</statement>
<statement name="ELSE">
    <block type="servo_cont" id="3240" inline="false">
        <field name="ROT">90</field>
        <value name="PIN">
            <block type="pin_digital" id="3241">
                <field name="PIN">6</field>
            </block>
        </value>
        <value name="DELAY_TIME">
            <block type="math_number" id="3242">
                <field name="NUM">10</field>
            </block>
        </value>
    </block>
</statement>
<next>
    <block type="controls_if" id="3243" inline="false">
        <mutation else="1" />
        <value name="IF0">
            <block type="logic_compare" id="3244" inline="true">
                <field name="OP">EQ</field>
                <value name="A">
                    <block type="variables_get" id="3245">
                        <field name="VAR">ir_izdo</field>
                    </block>
                </value>
                <value name="B">
                    <block type="variables_get" id="3246">
                        <field name="VAR">Negro</field>
                    </block>
                </value>
            </block>
        </value>
    </block>
</next>
```

```
</block>
</value>
<statement name="DO0">
    <block type="servo_cont" id="3247" inline="false">
        <field name="ROT">0</field>
        <value name="PIN">
            <block type="pin_digital" id="3248">
                <field name="PIN">9</field>
            </block>
        </value>
        <value name="DELAY_TIME">
            <block type="math_number" id="3249">
                <field name="NUM">10</field>
            </block>
        </value>
    </block>
</statement>
<statement name="ELSE">
    <block type="servo_cont" id="3250" inline="false">
        <field name="ROT">90</field>
        <value name="PIN">
            <block type="pin_digital" id="3251">
                <field name="PIN">6</field>
            </block>
        </value>
        <value name="DELAY_TIME">
            <block type="math_number" id="3252">
                <field name="NUM">10</field>
            </block>
        </value>
    </block>
</statement>
</block>
</next>
```

```
</block>
</next>
</block>
</statement>
</block>
<block type="procedures_defnoreturn" id="3253" x="-221" y="495">
<mutation />
<field name="NAME">RECIBIR</field>
<comment pinned="false" h="47" w="359">Si recibe una 'A' encenderemos
el LED.</comment>
<statement name="STACK">
<block type="bt_serial_available" id="3254">
<comment pinned="false" h="80" w="160" />
<statement name="DO">
<block type="variables_local" id="3255" inline="false">
<field name="VAR">dato</field>
<value name="VALUE">
<block type="bq_bluetooth_receive" id="3256" />
</value>
<next>
<block type="controls_switch" id="3257" inline="true">
<mutation case="6" />
<value name="IF0">
<block type="variables_get" id="3258">
<field name="VAR">dato</field>
</block>
</value>
<value name="SWITCH1">
<block type="math_number" id="3259">
<field name="NUM">85</field>
<comment pinned="false" h="80" w="160">En código Ascii:
U = 85</comment>
</block>
</value>
```

```
<statement name="DO1">
  <block type="procedures_callnoreturn" id="3260">
    <mutation name="Avanzar" />
    <field name="PROCEDURES">Avanzar</field>
    <next>
      <block type="procedures_callnoreturn" id="3261">
        <mutation name="Parar" />
        <field name="PROCEDURES">Parar</field>
      </block>
    </next>
  </block>
</statement>
<value name="SWITCH2">
  <block type="math_number" id="3262">
    <field name="NUM">68</field>
    <comment pinned="false" h="80" w="160">En código Ascii:
      D = 68</comment>
  </block>
</value>
<statement name="DO2">
  <block type="procedures_callnoreturn" id="3263">
    <mutation name="Retroceder" />
    <field name="PROCEDURES">Retroceder</field>
    <next>
      <block type="procedures_callnoreturn" id="3264">
        <mutation name="Parar" />
        <field name="PROCEDURES">Parar</field>
      </block>
    </next>
  </block>
</statement>
<value name="SWITCH3">
  <block type="math_number" id="3265">
    <field name="NUM">76</field>
```

```
<comment pinned="false" h="80" w="160">En código Ascii:  
    L (Left) = 76</comment>  
</block>  
</value>  
<statement name="DO3">  
    <block type="procedures_callnoretturn" id="3266">  
        <mutation name="Girar_dcha" />  
        <field name="PROCEDURES">Girar_dcha</field>  
        <next>  
            <block type="procedures_callnoretturn" id="3267">  
                <mutation name="Parar" />  
                <field name="PROCEDURES">Parar</field>  
            </block>  
        </next>  
    </block>  
</statement>  
<value name="SWITCH4">  
    <block type="math_number" id="3268">  
        <field name="NUM">82</field>  
        <comment pinned="false" h="80" w="160">En código Ascii:  
            R (Right) = 82</comment>  
        </block>  
</value>  
<statement name="DO4">  
    <block type="procedures_callnoretturn" id="3269">  
        <mutation name="Girar_izda" />  
        <field name="PROCEDURES">Girar_izda</field>  
        <next>  
            <block type="procedures_callnoretreturn" id="3270">  
                <mutation name="Parar" />  
                <field name="PROCEDURES">Parar</field>  
            </block>  
        </next>  
    </block>
```

```
</statement>
<value name="SWITCH5">
    <block type="math_number" id="3271">
        <field name="NUM">83</field>
        <comment pinned="false" h="80" w="160">En código Ascii:
            S (Stop) = 83</comment>
    </block>
</value>
<statement name="DO5">
    <block type="procedures_callnoreturn" id="3272">
        <mutation name="Parar" />
        <field name="PROCEDURES">Parar</field>
    </block>
</statement>
<value name="SWITCH6">
    <block type="math_number" id="3273">
        <field name="NUM">73</field>
        <comment pinned="false" h="80" w="160">En código Ascii:
            I (Modo Siguelineas) = 73</comment>
    </block>
</value>
<statement name="DO6">
    <block type="variables_set" id="3274" inline="false">
        <field name="VAR">StopI</field>
        <value name="VALUE">
            <block type="math_number" id="3275">
                <field name="NUM">0</field>
            </block>
        </value>
        <next>
            <block type="controls_whileUntil" id="3276" inline="false">
                <field name="MODE">WHILE</field>
                <value name="BOOL">
                    <block type="logic_compare" id="3277" inline="true">
```

```
<field name="OP">NEQ</field>
<value name="A">
  <block type="variables_get" id="3278">
    <field name="VAR">StopI</field>
  </block>
</value>
<value name="B">
  <block type="math_number" id="3279">
    <field name="NUM">77</field>
  </block>
</value>
<block type="procedures_callnoreturn" id="3280">
  <mutation name="Modo_siguelineas" />
<field name="PROCEDURES">Modo_siguelineas</field>
<next>
  <block type="bt_serial_available" id="3281">
    <statement name="DO">
      <block type="variables_set" id="3282" inline="false">
        <field name="VAR">StopI</field>
        <value name="VALUE">
          <block type="bq_bluetooth_receive" id="3283" />
        </value>
      </block>
    </statement>
  </block>
</next>
</block>
</statement>
</block>
</next>
</block>
```

```
</statement>
</block>
</next>
</block>
</statement>
</block>
</statement>
</block>
<block type="procedures_defnoreturn" id="3284" x="-218" y="1285">
<mutation />
<field name="NAME">Avanzar</field>
<statement name="STACK">
<block type="servo_cont" id="3285" inline="false">
<field name="ROT">180</field>
<value name="PIN">
<block type="pin_digital" id="3286">
<field name="PIN">9</field>
</block>
</value>
<value name="DELAY_TIME">
<block type="math_number" id="3287">
<field name="NUM">10</field>
</block>
</value>
<next>
<block type="servo_cont" id="3288" inline="false">
<field name="ROT">0</field>
<value name="PIN">
<block type="pin_digital" id="3289">
<field name="PIN">6</field>
</block>
</value>
<value name="DELAY_TIME">
<block type="math_number" id="3290">
```

```
<field name="NUM">10</field>
</block>
</value>
<next>
<block type="base_delay" id="3291" inline="true">
<value name="DELAY_TIME">
<block type="math_number" id="3292">
<field name="NUM">1000</field>
</block>
</value>
</block>
</next>
</block>
</next>
</block>
</statement>
</block>
<block type="procedures_defnoreturn" id="3293" x="460" y="1288">
<mutation />
<field name="NAME">Retroceder</field>
<statement name="STACK">
<block type="servo_cont" id="3294" inline="false">
<field name="ROT">0</field>
<value name="PIN">
<block type="pin_digital" id="3295">
<field name="PIN">9</field>
</block>
</value>
<value name="DELAY_TIME">
<block type="math_number" id="3296">
<field name="NUM">10</field>
</block>
</value>
<next>
```

```
<block type="servo_cont" id="3297" inline="false">
    <field name="ROT">180</field>
    <value name="PIN">
        <block type="pin_digital" id="3298">
            <field name="PIN">6</field>
        </block>
    </value>
    <value name="DELAY_TIME">
        <block type="math_number" id="3299">
            <field name="NUM">10</field>
        </block>
    </value>
    <next>
        <block type="base_delay" id="3300" inline="true">
            <value name="DELAY_TIME">
                <block type="math_number" id="3301">
                    <field name="NUM">1000</field>
                </block>
            </value>
        </block>
    </next>
    </block>
</statement>
</block>
<block type="procedures_defnoreturn" id="3302" x="-222" y="1682">
    <mutation />
    <field name="NAME">Girar_izda</field>
    <statement name="STACK">
        <block type="servo_cont" id="3303" inline="false">
            <field name="ROT">0</field>
            <value name="PIN">
                <block type="pin_digital" id="3304">
```

```
<field name="PIN">9</field>
</block>
</value>
<value name="DELAY_TIME">
<block type="math_number" id="3305">
<field name="NUM">10</field>
</block>
</value>
<next>
<block type="servo_cont" id="3306" inline="false">
<field name="ROT">0</field>
<value name="PIN">
<block type="pin_digital" id="3307">
<field name="PIN">6</field>
</block>
</value>
<value name="DELAY_TIME">
<block type="math_number" id="3308">
<field name="NUM">10</field>
</block>
</value>
<next>
<block type="base_delay" id="3309" inline="true">
<value name="DELAY_TIME">
<block type="math_number" id="3310">
<field name="NUM">1000</field>
</block>
</value>
</block>
</next>
</block>
</statement>
```

```
</block>
<block type="procedures_defnoreturn" id="3311" x="459" y="1685">
  <mutation />
  <field name="NAME">Girar_dcha</field>
  <statement name="STACK">
    <block type="servo_cont" id="3312" inline="false">
      <field name="ROT">180</field>
      <value name="PIN">
        <block type="pinDigital" id="3313">
          <field name="PIN">9</field>
        </block>
      </value>
      <value name="DELAY_TIME">
        <block type="math_number" id="3314">
          <field name="NUM">10</field>
        </block>
      </value>
    <next>
      <block type="servo_cont" id="3315" inline="false">
        <field name="ROT">180</field>
        <value name="PIN">
          <block type="pinDigital" id="3316">
            <field name="PIN">6</field>
          </block>
        </value>
        <value name="DELAY_TIME">
          <block type="math_number" id="3317">
            <field name="NUM">10</field>
          </block>
        </value>
      <next>
        <block type="base_delay" id="3318" inline="true">
          <value name="DELAY_TIME">
            <block type="math_number" id="3319">
```

```
<field name="NUM">1000</field>
</block>
</value>
</block>
</next>
</block>
</next>
</block>
</statement>
</block>
<block type="procedures_defnoreturn" id="3320" x="-211" y="2091">
<mutation />
<field name="NAME">Parar</field>
<statement name="STACK">
<block type="servo_cont" id="3321" inline="false">
<field name="ROT">90</field>
<value name="PIN">
<block type="pin_digital" id="3322">
<field name="PIN">9</field>
</block>
</value>
<value name="DELAY_TIME">
<block type="math_number" id="3323">
<field name="NUM">10</field>
</block>
</value>
<next>
<block type="servo_cont" id="3324" inline="false">
<field name="ROT">90</field>
<value name="PIN">
<block type="pin_digital" id="3325">
<field name="PIN">6</field>
</block>
</value>
```

```
<value name="DELAY_TIME">
  <block type="math_number" id="3326">
    <field name="NUM">10</field>
  </block>
</value>
</block>
</next>
</block>
</statement>
</block>
</xml>
```

5.4 Anexo IV. Ejemplo de proyectos de bloques, código y de robots de Bitbloq 2

Ejemplo de proyecto de **bloques** de bitbloq 2:

```
"name" : "proyecto ejemplo",
"description" : "ejemplo",
"userTags" : [ ],
"hardwareTags" : [
  "rtc",
  "servocont",
  "buzz",
  "led",
  "sound",
  "ldr",
  "sp",
  "bq ZUM"
],
"compiled" : false,
"imageType" : "",
```

```
"videoUrl" : "",  
"defaultTheme" : "infotab_option_colorTheme",  
"software" : {  
    "vars" : {  
        "name" : "varsBloq",  
        "enable" : true,  
        "child" : [ ],  
        "content" : [  
            []  
        ]  
    },  
    "setup" : {  
        "name" : "setupBloq",  
        "enable" : true,  
        "child" : [  
            {  
                "name" : "clockRTCInit",  
                "enable" : true,  
                "content" : [  
                    [  
                        {  
                            "alias" : "dynamicDropdown",  
                            "_id" : "RTC",  
                            "value" : "reloj_tiempo_real_0"  
                        }  
                    ]  
                ]  
            }  
        ],  
        "content" : [  
            []  
        ]  
    },  
    "loop" : {
```

```
"name" : "loopBloq",
"enable" : true,
"childs" : [
{
  "name" : "buzzer",
  "enable" : true,
  "content" : [
    [
      {
        "alias" : "dynamicDropdown",
        "_id" : "BUZZER",
        "value" : "buzzer_0"
      },
      {
        "alias" : "staticDropdown",
        "_id" : "NOTE",
        "value" : "261"
      },
      {
        "alias" : "numberInput",
        "_id" : "SECONDS",
        "value" : "2000"
      }
    ]
  ],
  "name" : "continuousServoStart",
  "enable" : true,
  "content" : [
    [
      {
        "alias" : "dynamicDropdown",
        "_id" : "SERVO",
        "value" : "SERVO_0"
      }
    ]
  ]
},
```

```
        "value" : "servo_cont_0"
    },
    {
        "alias" : "staticDropdown",
        "_id" : "DIRECTION",
        "value" : "180"
    }
]
]
},
{
    "name" : "continuousServoStop",
    "enable" : true,
    "content" : [
        [
            {
                "alias" : "dynamicDropdown",
                "_id" : "SERVO",
                "value" : "servo_cont_0"
            }
        ]
    ]
},
{
    "name" : "continuousServoStart",
    "enable" : true,
    "content" : [
        [
            {
                "alias" : "dynamicDropdown",
                "_id" : "SERVO",
                "value" : "servo_cont_0"
            },
            {

```

```
        "alias" : "staticDropdown",
        "_id" : "DIRECTION",
        "value" : "180"
    }
]
],
},
{
    "name" : "if",
    "enable" : true,
    "child" : [
        {
            "name" : "serialSend-v1",
            "enable" : true,
            "content" : [
                [
                    {
                        "alias" : "dynamicDropdown",
                        "_id" : "SERIAL",
                        "value" : "serial_port_0"
                    },
                    {
                        "alias" : "bloqInput",
                        "bloqInputId" : "DATA",
                        "value" : {
                            "name" : "clockRTC",
                            "enable" : true,
                            "content" : [
                                [
                                    {
                                        "alias" : "staticDropdown",
                                        "_id" : "RTC_FUNC",
                                        "value" : "getDate"
                                    },
                                ],
                            ]
                        }
                    }
                ]
            ]
        }
    ]
}
```

```
{
    "alias" : "dynamicDropdown",
    "_id" : "RTC",
    "value" : "reloj_tiempo_real_0"
}
]
}
},
{
    "alias" : "staticDropdown",
    "_id" : "LN",
    "value" : "println"
}
]
},
{
    "name" : "serialSend-v1",
    "enable" : true,
    "content" : [
        [
            {
                "alias" : "dynamicDropdown",
                "_id" : "SERIAL",
                "value" : "serial_port_0"
            },
            {
                "alias" : "bloqInput",
                "bloqInputId" : "DATA",
                "value" : {
                    "name" : "clockRTCAvanced",
                    "enable" : true,
                    "content" : [

```

```
[  
  {  
    "alias" : "staticDropdown",  
    "_id" : "FUNCTION",  
    "value" : "getHour"  
  },  
  {  
    "alias" : "dynamicDropdown",  
    "_id" : "RTC",  
    "value" : "reloj_tiempo_real_0"  
  }]  
]  
}  
},  
{  
  "alias" : "staticDropdown",  
  "_id" : "LN",  
  "value" : "println"  
}  
]  
]  
}  
],  
"content" : [  
  [  
    {  
      "alias" : "bloqInput",  
      "bloqInputId" : "ARG1",  
      "value" : {  
        "name" : "readSensor",  
        "enable" : true,  
        "content" : [  
          [  
            {  
              "alias" : "functionCall",  
              "bloqInputId" : "ARG1",  
              "functionName" : "getHour",  
              "value" : {}  
            }]  
          ]  
        ]  
      }]  
    }]  
  ]  
]
```

```
{  
    "alias" : "dynamicDropdown",  
    "_id" : "SENSOR",  
    "value" : "light_sensor_0"  
}  
]  
]  
}  
,  
{  
    "alias" : "staticDropdown",  
    "_id" : "OPERATOR",  
    "value" : "=="  
},  
{  
    "alias" : "bloqInput",  
    "bloqInputId" : "ARG2",  
    "value" : {  
        "name" : "number",  
        "enable" : true,  
        "content" : [  
            [  
                {  
                    "alias" : "numberInput",  
                    "_id" : "VALUE",  
                    "value" : "0"  
                }  
            ]  
        ]  
    }  
}  
]  
}
```

```
        ],
      "content" : [
        []
      ]
    }
},
"hardware" : {
  "components" : [
    {
      "_id" : "rtc",
      "icon" : "rtc",
      "type" : "analog",
      "width" : NumberLong(128),
      "height" : NumberLong(93),
      "pin" : {
        "sda" : "A4",
        "scl" : "A5"
      },
      "pins" : {
        "analog" : [
          "sda",
          "scl"
        ]
      },
      "coordinates" : {
        "x" : 59.8526703499079247,
        "y" : 79.755671902268758
      },
      "category" : "clocks",
      "name" : "reloj_tiempo_real_0",
      "connected" : true,
      "uid" : "bac8275b-e197-4500-89ca-34e1a812afb0",
      "endpoints" : {
        "sda" : {
```

200

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289

Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa
Operativo Plurirregional de Crecimiento Inteligente 2014-2020

```
"type" : "analog",
"uid" : "1bf98a53-4c9b-470f-8603-aa75742cf110"
},
"scl" : {
    "type" : "analog",
    "uid" : "5a0d7f43-f81c-4da1-960a-0f2f9904cc50"
}
},
{
    "_id" : "servocont",
    "icon" : "servoRotContinua",
    "width" : NumberLong(125),
    "height" : NumberLong(106),
    "pins" : {
        "digital" : [
            "S"
        ]
    },
    "coordinates" : {
        "x" : 67.3112338858195187,
        "y" : 10.9947643979057599
    },
    "category" : "continuousServos",
    "name" : "servo_cont_0",
    "connected" : true,
    "uid" : "e092a928-01cd-4bdd-9f4c-ea99c8e3cf80",
    "pin" : {
        "S" : "2"
    },
    "endpoints" : {
        "S" : {
            "type" : "digital",
            "uid" : "2581837a-1b81-437c-a437-ed086873247b"
        }
    }
}
```

```
        }
    },
},
{
    "_id" : "buzz",
    "icon" : "buzzer",
    "type" : "digital",
    "width" : NumberLong(85),
    "height" : NumberLong(80),
    "pins" : {
        "digital" : [
            "s"
        ]
    },
    "coordinates" : {
        "x" : 50.7366482504604051,
        "y" : 6.2827225130890048
    },
    "category" : "buzzers",
    "name" : "buzzer_0",
    "connected" : true,
    "uid" : "1f2729a6-ba6f-46ac-ae53-c763ccb753ef",
    "pin" : {
        "s" : "6"
    },
    "endpoints" : {
        "s" : {
            "type" : "digital",
            "uid" : "36a95bf8-e377-40d1-bcac-12f504892e9c"
        }
    }
},
{
    "_id" : "led",
```

```
"icon" : "LED",
"width" : NumberLong(55),
"height" : NumberLong(83),
"pins" : {
    "digital" : [
        "s"
    ]
},
"coordinates" : {
    "x" : 37.4769797421731141,
    "y" : 3.6649214659685860
},
"category" : "leds",
"name" : "led_0",
"connected" : true,
"uid" : "dd3b1eba-0a17-41e9-8bee-a093a3d1368b",
"pin" : {
    "s" : "10"
},
"endpoints" : {
    "s" : {
        "type" : "digital",
        "uid" : "43042cc9-95c9-4512-b0f5-55b6df765353"
    }
},
{
    "_id" : "sound",
    "icon" : "micro",
    "type" : "digital",
    "width" : NumberLong(100),
    "height" : NumberLong(102),
    "pins" : {
        "digital" : [

```

```
        "s"
    ]
},
"coordinates" : {
    "x" : 80.2946593001841507,
    "y" : 27.2251308900523554
},
"category" : "sensors",
"name" : "sensor_sonido_0",
"connected" : true,
"uid" : "5dd0af9-adcc-465c-bbf6-975d2c0aa22f",
"pin" : {
    "s" : "4"
},
"endpoints" : {
    "s" : {
        "type" : "digital",
        "uid" : "51b5aafc-ae36-48fc-af64-1438b672796d"
    }
}
},
{
    "_id" : "ldr",
    "icon" : "LDR",
    "type" : "analog",
    "width" : NumberLong(90),
    "height" : NumberLong(65),
    "pins" : {
        "analog" : [
            "s"
        ]
    },
    "coordinates" : {
        "x" : 47.5138121546961329,
```

```
"y" : 79.5811518324607334
},
"category" : "sensors",
"name" : "light_sensor_0",
"connected" : true,
"uid" : "803b0bde-e8dc-4408-91de-8d29697313cb",
"pin" : {
    "s" : "A2"
},
"endpoints" : {
    "s" : {
        "type" : "analog",
        "uid" : "a99677d7-e663-44ba-acc4-e4f604ed8b7e"
    }
}
},
{
    "_id" : "sp",
    "icon" : "serial",
    "baudRate" : "9600",
    "width" : NumberLong(115),
    "height" : NumberLong(71),
    "pin" : {
        "s" : "serial"
    },
    "pins" : {
        "serial" : [
            "s"
        ]
    },
    "coordinates" : {
        "x" : 19.8895027624309400,
        "y" : 42.4083769633507828
    },
}
```

```
"category" : "serialElements",
"name" : "serial_port_0",
"connected" : true,
"uid" : "15688833-ab82-49b7-8abb-2060c1b39da5",
"endpoints" : [
  {
    "s" : {
      "type" : "serial",
      "uid" : "cb22dfa2-0e98-48e1-9a5f-042eeaf51a58"
    }
  }
],
"connections" : [
  {
    "pinSourceUid" : "1bf98a53-4c9b-470f-8603-aa75742cf110",
    "pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53ba01"
  },
  {
    "pinSourceUid" : "5a0d7f43-f81c-4da1-960a-0f2f9904cc50",
    "pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53ba00"
  },
  {
    "pinSourceUid" : "2581837a-1b81-437c-a437-ed086873247b",
    "pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53bd11"
  },
  {
    "pinSourceUid" : "36a95bf8-e377-40d1-bcac-12f504892e9c",
    "pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53bd07"
  },
  {
    "pinSourceUid" : "43042cc9-95c9-4512-b0f5-55b6df765353",
    "pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53bd03"
  },
  {
    "pinSourceUid" : "43042cc9-95c9-4512-b0f5-55b6df765353",
    "pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53bd03"
  }
]
```

```
"pinSourceUid" : "51b5aafc-ae36-48fc-af64-1438b672796d",
"pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53bd09"
},
{
    "pinSourceUid" : "a99677d7-e663-44ba-acc4-e4f604ed8b7e",
    "pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53ba03"
},
{
    "pinSourceUid" : "cb22dfa2-0e98-48e1-9a5f-042eeaf51a58",
    "pinTargetUid" : "6be0dd9d-2e52-4b7d-9dfc-c9edad53bc05"
}
],
"board" : "bq ZUM"
},
"code" : "
/** Included libraries */
#include <Servo.h>
#include <SoftwareSerial.h>
#include <BitbloqSoftwareSerial.h>
#include <Wire.h>
#include <BitbloqRTC.h>

/* Global variables and function definition */
int buzzer_0 = 6; RTC_DS1307 reloj_tiempo_real_0; Servo servo_cont_0; int led_0 = 10; int sensor_sonido_0 = 4; int light_sensor_0 = A2; bqSoftwareSerial serial_port_0(0,1,9600);

void setup() { servo_cont_0.attach(2); pinMode(led_0, OUTPUT); pinMode(sensor_so INPUT); pinMode(light_sensor_0, INPUT); reloj_tiempo_real_0.adjust(DateTime(__DATE__, __TIME__)); }

void loop() { tone(buzzer_0,261,2000); delay(2000); servo == 0) { serial_port_0.println(reloj_tiempo_real_0.getDate()); serial_port_0.println(reloj_tiempo_real_0.getTime()); }

    "creatorUsername" : "ejemplo",
    "_acl" : {
        "user:123456asdfg" : {
            "permission" : "ADMIN",
            "properties" : {}
        }
    },
    "_createdAt" : ISODate("2015-12-10T08:21:47.102Z"),
    "_updatedAt" : ISODate("2015-12-10T08:27:50.583Z")
}
```

Ejemplo de proyecto base de **código**:

```
{  
    "_id" : "123456asdfg:18941914681956-189d4fgsdfg94sdf",  
    "creatorId" : "123456asdfg",  
    "name" : "proyecto ejemplo",  
    "description" : "ejemplo",  
    "userTags" : [  
        "proyectoPropio"  
    ],  
    "hardwareTags" : [  
        "bq ZUM"  
    ],  
    "hardware" : {  
        "board" : "bq ZUM"  
    },  
    "compiled" : false,  
    "imageType" : "",  
    "videoUrl" : "",  
    "code" : "/*** Included libraries ***/\n#include <Servo.h>\n#include <SoftwareSerial.h>\n#include <BitbloqSoftwareSerial.h>\n#include <Wire.h>\n#include <BitbloqRTC.h>\n\n/** Global variables and function definition ***/\nint buzzer_0 = 6;\nRTC_DS1307 reloj_tiempo_real_0;\nServo servo_cont_0;\nint led_0 = 10;\nint sensor_sonido_0 = 4;\nint light_sensor_0 = A2;\nSoftwareSerial serial_port_0(0, 1, 9600);  
/** Setup ***/  
void setup() {  
    servo_cont_0.attach(2);  
    pinMode(led_0, OUTPUT);  
    pinMode(sensor_sonido_0, INPUT);  
    pinMode(light_sensor_0, INPUT);  
    reloj_tiempo_real_0.adjust(DateTime(__DATE__,  
__TIME__));  
}  
/** Loop ***/  
void loop() {  
    tone(buzzer_0, 261, 2000);  
    delay(2000);  
    servo_cont_0.write(180);  
    servo_cont_0.write(90);  
    servo_cont_0.write(180);  
    if (analogRead(A2) == 0) {  
        serial_port_0.println(reloj_tiempo_real_0.getHour());  
    }  
}  
    "codeProject" : true,  
    "creatorUsername" : "ejemplo",  
    "_acl" : {
```

```
"user:123456asdfg" : {  
    "permission" : "ADMIN",  
    "properties" : {}  
}  
},  
"_createdAt" : ISODate("2015-12-10T08:35:35.688Z"),  
"_updatedAt" : ISODate("2015-12-10T08:37:43.625Z")  
}
```

Ejemplo proyecto de **Zowi**:

```
{
    "_id" : "123456asdfg:ca1565d4",
    "creatorId" : "123456asdfg",
    "name" : "proyecto zowi",
    "description" : "",
    "userTags" : [
        "init"
    ],
    "hardwareTags" : [
        "zowi",
        "Arduino UNO"
    ],
    "compiled" : false,
    "imageType" : "",
    "videoUrl" : "",
    "defaultTheme" : "infotab_option_colorTheme",
    "software" : {
        "vars" : {
            "name" : "varsBloq",
            "enable" : true,
            "childs" : [],
            "content" : [
                []
            ]
        }
    }
}
```

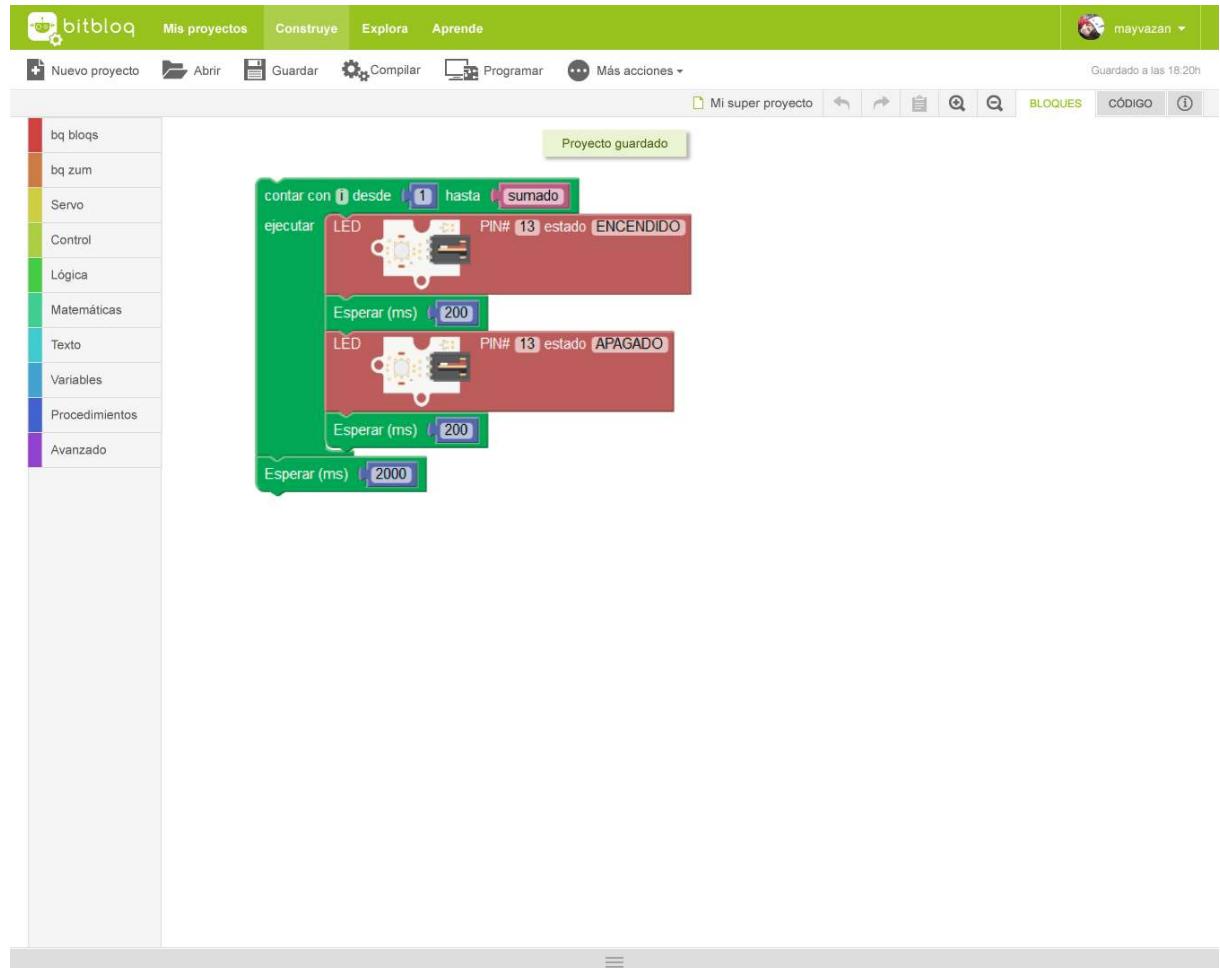
```
        ],
    },
    "setup" : {
        "name" : "setupBloq",
        "enable" : true,
        "child" : [
            {
                "name" : "zowiHome",
                "enable" : true,
                "content" : [
                    []
                ]
            }
        ],
        "content" : [
            []
        ]
    },
    "loop" : {
        "name" : "loopBloq",
        "enable" : true,
        "child" : [
            {
                "name" : "zowiMovementsSimple",
                "enable" : true,
                "content" : [
                    [
                        {
                            "alias" : "staticDropdown",
                            "_id" : "MOVEMENT",
                            "value" : "walk"
                        },
                        {
                            "alias" : "numberInput",

```

```
        "_id" : "STEPS",
        "value" : "4"
    }
]
],
},
{
    "name" : "zowiSounds",
    "enable" : true,
    "content" : [
        [
            {
                "alias" : "staticDropdown",
                "_id" : "SOUND",
                "value" : "S_happy"
            }
        ]
    ],
},
{
    "name" : "zowiGestures",
    "enable" : true,
    "content" : [
        [
            {
                "alias" : "staticDropdown",
                "_id" : "GESTURE",
                "value" : "ZowiHappy"
            }
        ]
    ],
}
],
"content" : [
```

```
        ],
    },
},
"hardware" : {
    "components" : [],
    "connections" : [],
    "board" : "Arduino UNO",
    "robot" : "zowi"
},
"code" : "\n/**\n * Included libraries\n */\n#include <BitbloqZowi.h>\n#include <BitbloqUS.h>\n#include <BitbloqBatteryReader.h>\n#include <BitbloqLed-Matrix.h>\n#include <Servo.h>\n#include <BitbloqOscillator.h>\n#include <EEP-ROM.h>\n\n/**\n * Global variables and function definition\n */\nZowi zowi;\n\n/**\n * Setup\n */\nvoid setup()\n{\n    zowi.init();\n    zowi.home();\n}\n\n/**\n * Loop\n */\nvoid loop()\n{\n    zowi.walk(4);\n    zowi.sing(S_happy);\n    zowi.playGesture(ZowiHappy);\n}\n\n",
"creatorUsername" : "ejemplo",
"_acl" : {
    "user:123456asdfg" : {
        "permission" : "ADMIN",
        "properties" : {}
    }
},
"_createdAt" : ISODate("2015-12-10T08:39:20.054Z"),
"_updatedAt" : ISODate("2015-12-10T08:52:01.617Z")
}
```

5.5 Anexo V. Wireframes bitbloq V1



BOTBLOQ: Ecosistema integral para el diseño, fabricación y programación de robots DIY



bitbloq

Quién | Por qué | Cómo funciona | Para quién | FAQs | Blog | Entrar

¿Quieres programar jugando?

Con Bitbloq puedes programar fácilmente y de una forma visual e intuitiva, haciendo de la programación y la electrónica algo fácil y divertido para los más pequeños.

Pruébalo ahora

Aprender a divertirse, divertirse aprendiendo

No hace falta que tengas conocimientos de programación, con Bitbloq aprenderás de una forma rápida e intuitiva

Facil y divertido

Nuevo aprendizaje

Programa tu robot

Comparte, modifica, prueba...

¡La imaginación al poder!

Con Bitbloq y la electrónica de bq vas a poder crear un robot único, entrando en el mundo de la robótica (¡vaya se detenrá para crear el robot que tienes en tu mente!).

Envíala un vistazo a los robots de bq.

Aprende a crear tecnología

La tecnología forma parte de nuestro día a día, pero no nos paramos a pensar cómo funciona o qué es necesario para lo haga un aparato. Con Bitbloq conseguiras que los más pequeños entiendan el funcionamiento de las cosas, creándoles una nueva forma de ver la tecnología.

Padres

¿Quieres que tus hijos aprendan robótica, electrónica y/o programación? Con Bitbloq les das las herramientas necesarias para que se diviertan en este mundo tan emocionante.

Mira como ofrecen Bitbloq los más pequeños:

Niños

¿Te imaginas poder crear un robot? ¿Y que ademáis hacerlo que lo pides? Con Bitbloq vais a poder crear lo que quieras, no hay límites.

Dile todo lo que puedes hacer!

Profesores

Los niños utilizan la tecnología como medios, usanlos para la mayoría no entiende cómo funciona realmente, con Bitbloq enseñarlos a ser conscientes de ello.

Usa nuestra barra de escritor

FAQs

¿Es Bitbloq una copia de Scratch? ▾

La respuesta es no. Los desarrolladores, no tenían licencia para utilizar la programación de Scratch, por lo tanto han desarrollado su propia programación.

¿Puedo hacer videollamadas con Bitbloq? ▾

La respuesta es no. Los desarrolladores, no tienen licencia para utilizar la programación de Scratch, por lo tanto han desarrollado su propia programación.

¿Puedo programar la placa controladora con Scratch? ▾

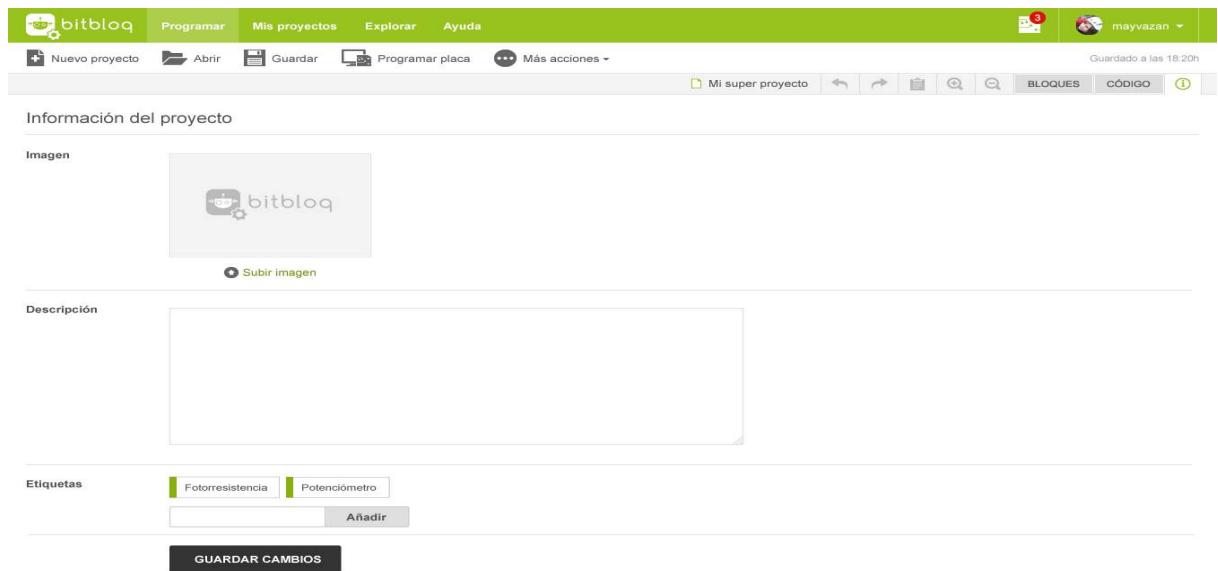
Si, puedes programar la placa controladora con Scratch.

¿Sirven para lo mismo? ▾

Cookies | Privacidad | Condiciones generales

214

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020



5.6 Anexo VI. Wireframes bitbloq V2

Se pueden consultar los Wireframes aquí: http://3ht0k9.axshare.com/#p=00_sitemap

5.7 Anexo VII. Diseños de la aplicación

A continuación se adjuntan los diseños de las diferentes vistas de la aplicación. En primer lugar se muestran los diseños a la página principal o landing.

BOTBLOQ: Ecosistema integral para el diseño, fabricación y programación de robots DIY



The screenshot displays the official website for botbloq. At the top, there's a navigation bar with links for 'CARACTERÍSTICAS', 'DESCARGA', 'Sobre nosotros', 'ENTRAR', and 'REGISTRARME'. The main headline reads 'Una nueva forma de programar, fácil, sencilla e intuitiva.' Below it, a subtext states 'Con bitbloq programarás con bloques visuales, sin necesidad de saber código.' A prominent green button labeled 'PRUEBALO AHORA' is centered. The central part of the page features a screenshot of the botbloq software interface, which shows a graphical programming environment with various blocks and a robot setup. Below this, three sections are highlighted: 'Para los más pequeños' (with an icon of a lightbulb), 'Tecnología en familia' (with an icon of a house), and 'Tecnología en el aula' (with an icon of a person holding a tablet). Each section includes a brief description and a link to more information. Further down, a section titled '¿Eres un colegio o un profesor?' includes a quote and a contact email. The bottom section, 'Explora un mundo libre', shows examples of user projects with small thumbnail images and project titles like 'Título del proyecto con una linea d...'. The footer contains language links (Castellano, English, Italiano, Français, Portugués, Català, Euskera), a 'Contacto' link, and copyright information.

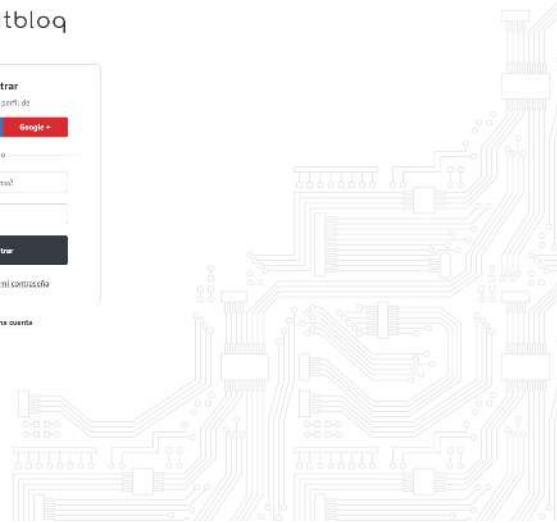
Los siguientes diseños corresponden al login y el registro de la aplicación:

216

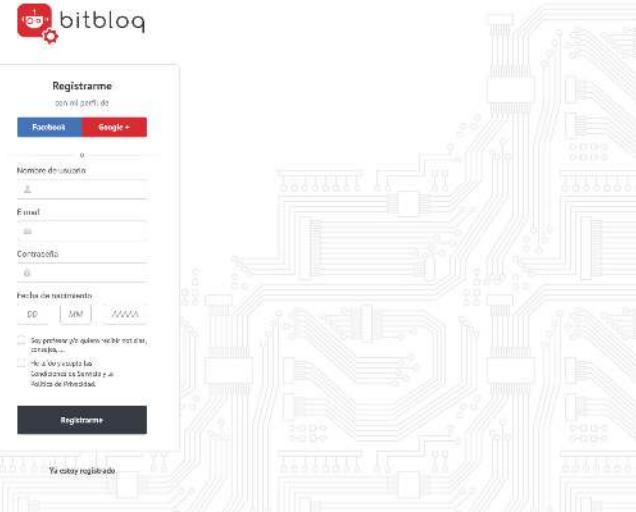
Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa Operativo Plurirregional de Crecimiento Inteligente 2014-2020



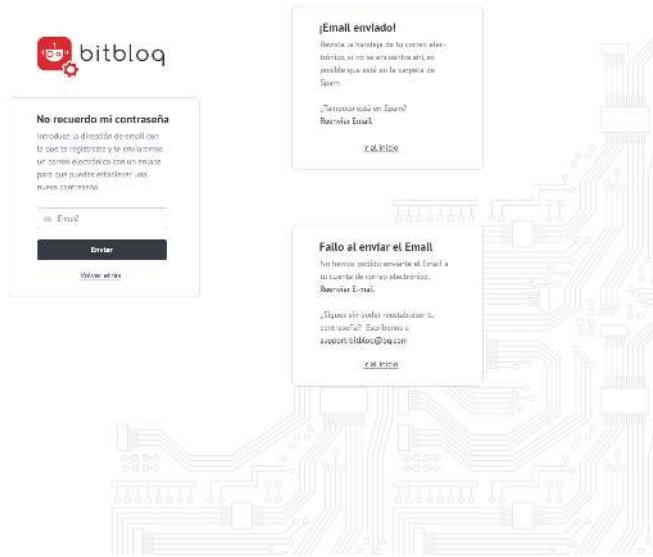
- LOGIN



- REGISTRO

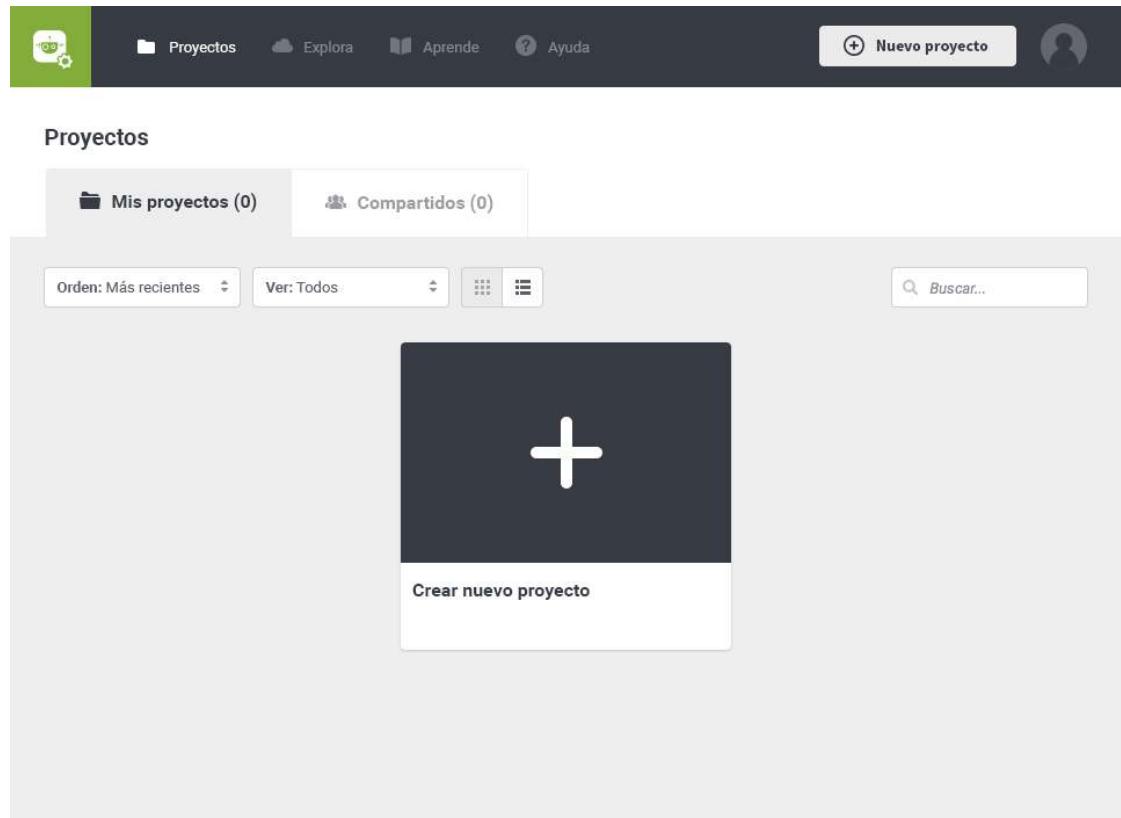


- RECORDAR LA CONTRASEÑA



Las siguientes capturas corresponden a los diseños de la vista de proyectos de un usuario:

- Cuando un usuario no tiene proyectos



cií

- Cuando un usuario tiene proyectos

Proyectos

Mis proyectos (23) Compartidos (5)

Orden: Más recientes Ver: Todos

Buscar...

Abrir proyecto

Título del proyecto con una linea d... 12 Mar. 2015 - 18:34

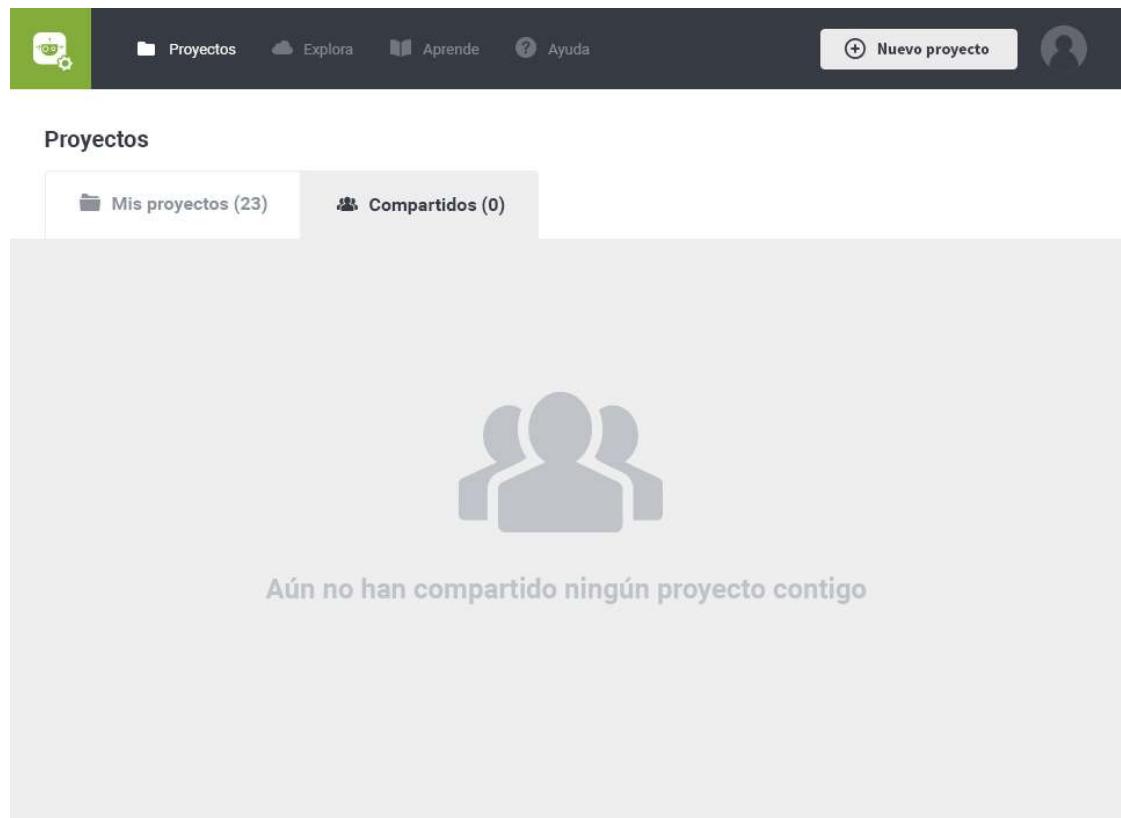
Abrir proyecto

Título del proyecto con una linea d... 12 Mar. 2015 - 12:08

Abrir proyecto

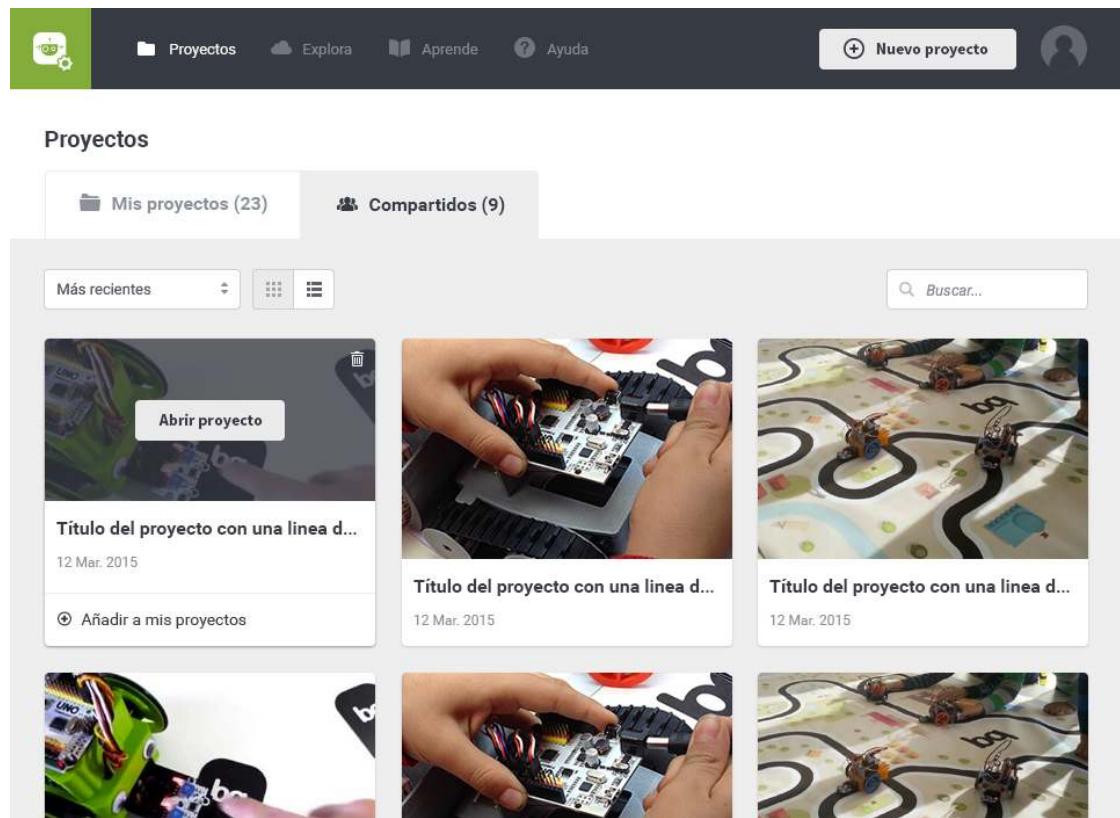
Título del proyecto con una linea d... 12 Mar. 2015 - 15:56

- Si un usuario no tiene proyectos compartidos



Aún no han compartido ningún proyecto contigo

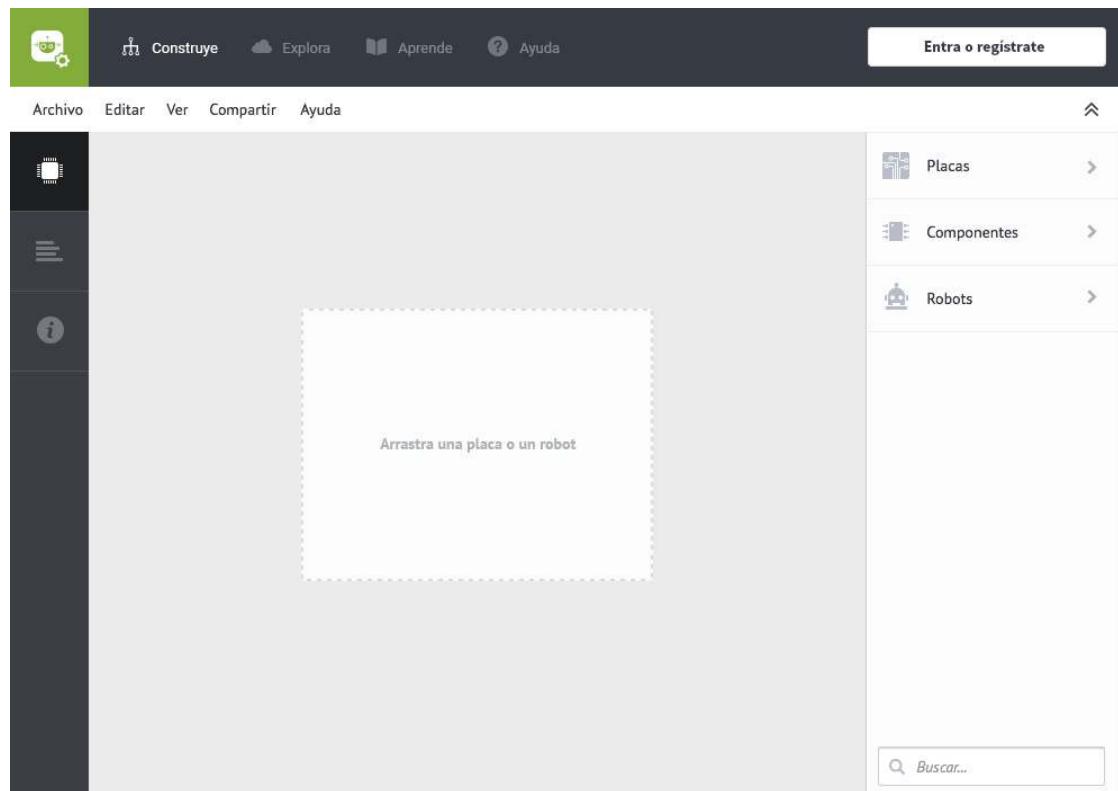
- Si un usuario tiene proyectos compartidos



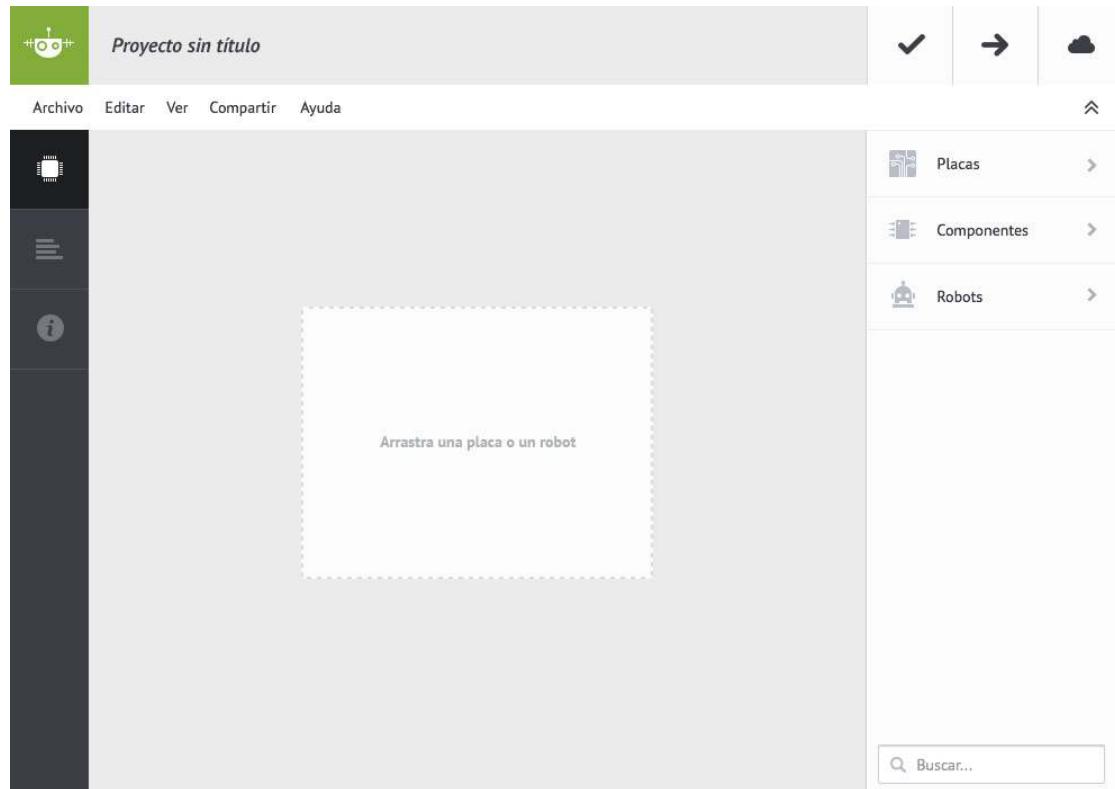
The screenshot shows the Botbloq software interface. At the top, there's a navigation bar with icons for Projects, Explore, Learn, Help, and a New Project button. Below the navigation bar is a section titled "Proyectos" (Projects). It displays two tabs: "Mis proyectos (23)" (My projects) and "Compartidos (9)" (Shared). A search bar labeled "Buscar..." is located at the top right of this section. Below the tabs, there are three project cards arranged in a grid. Each card has a thumbnail image, a title, a date, and an "Abrir proyecto" (Open project) button. The first card shows a green robot, the second shows hands working on a circuit board, and the third shows a robot on a track. At the bottom of each card, there's a "Añadir a mis proyectos" (Add to my projects) button.

Dentro del proyecto se tendrían los siguientes diseños:

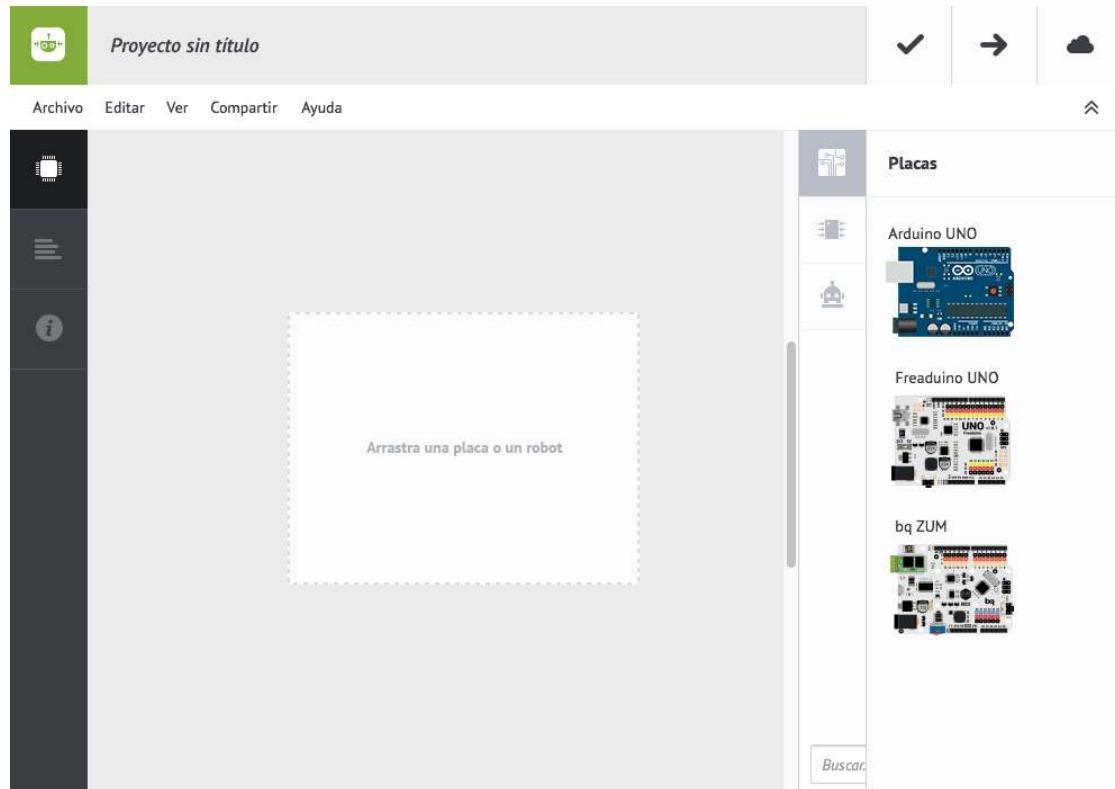
- Workspace en modo invitado:



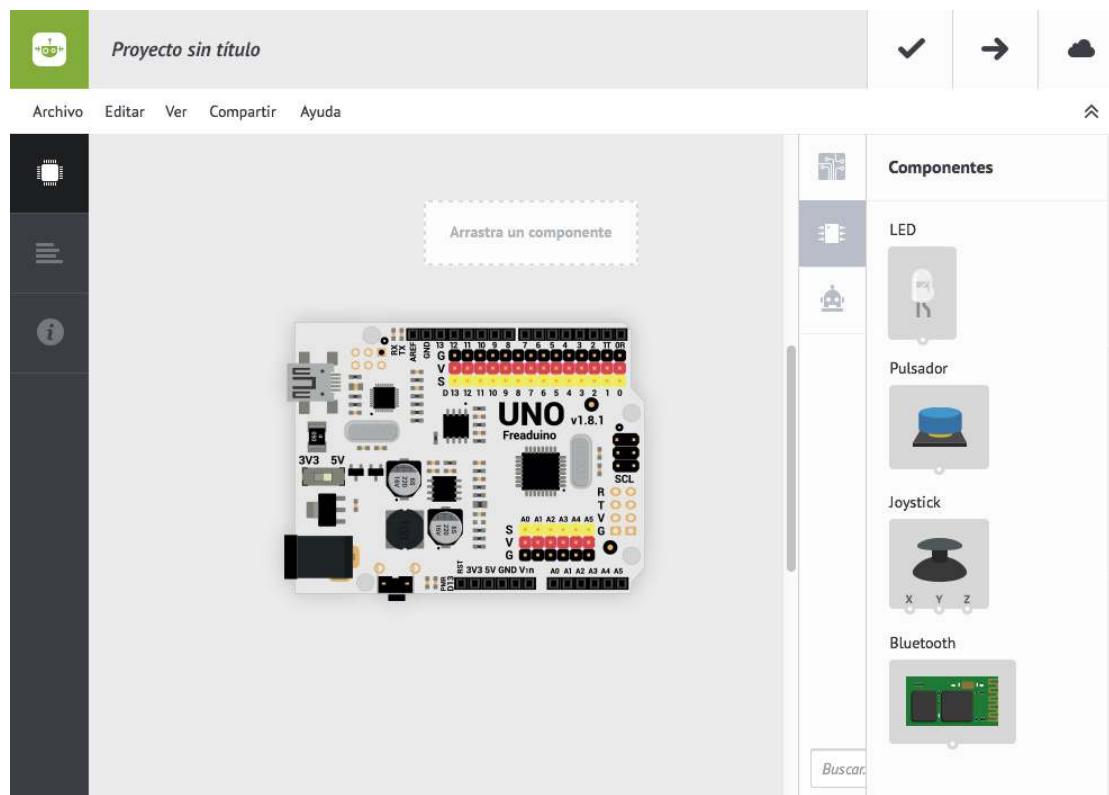
- Workspace con usuario registrado:



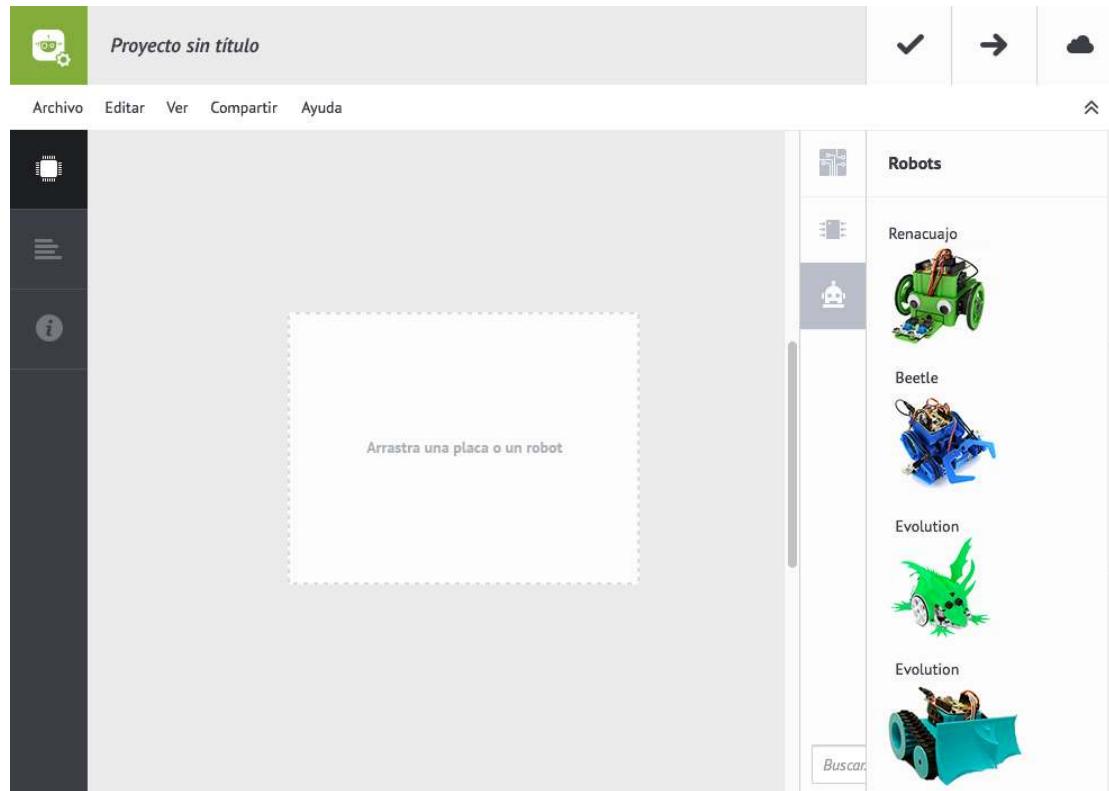
- Cuando en el workspace se despliega la pestaña de placas



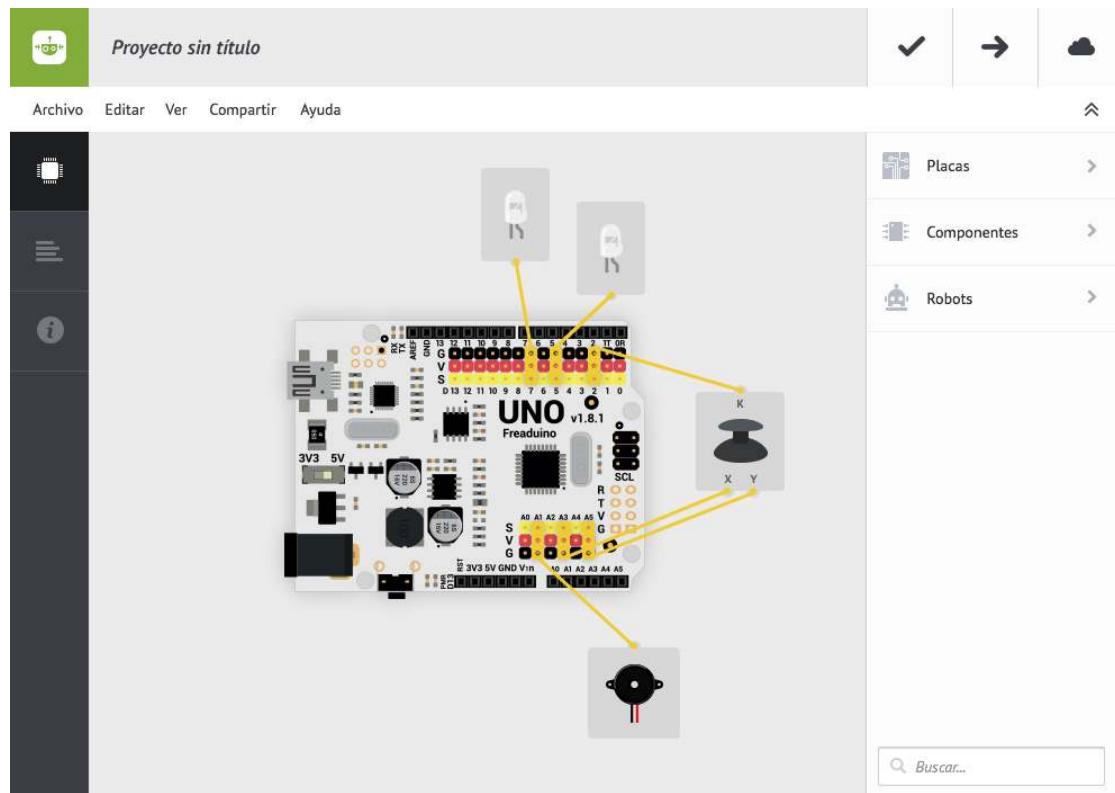
- Cuando en el workspace se despliega la pestaña de componentes:



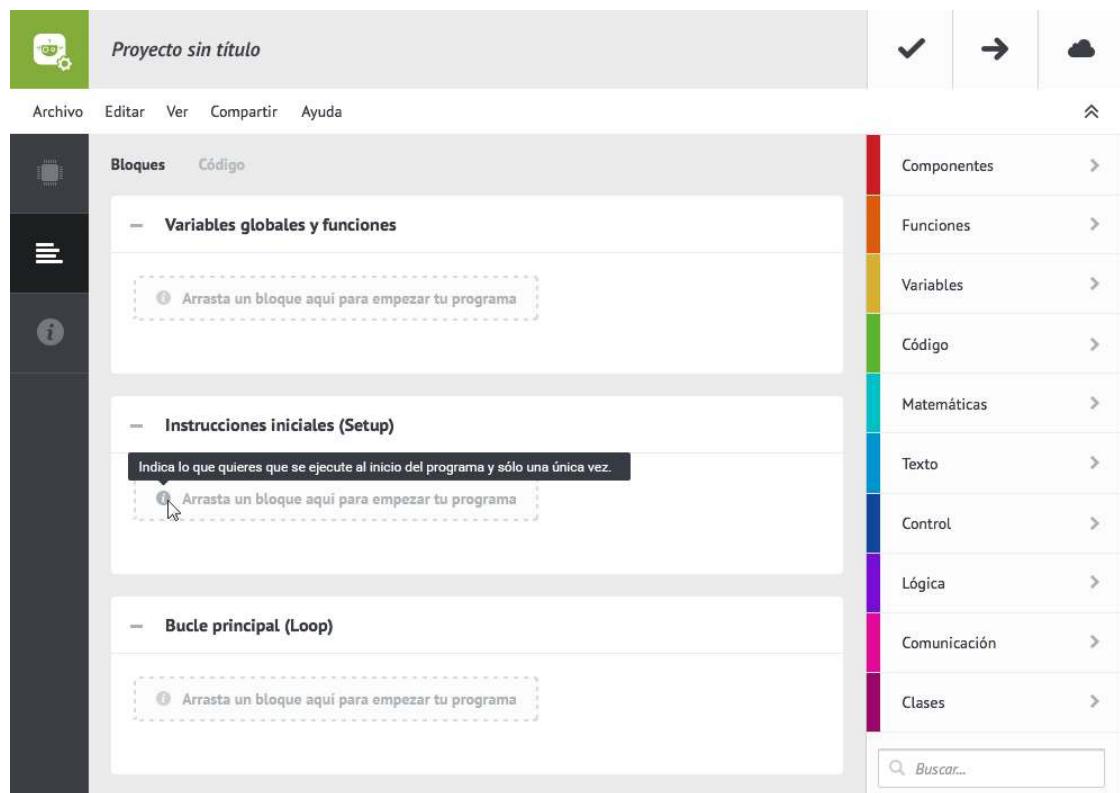
- Cuando se despliega la pestaña de robots



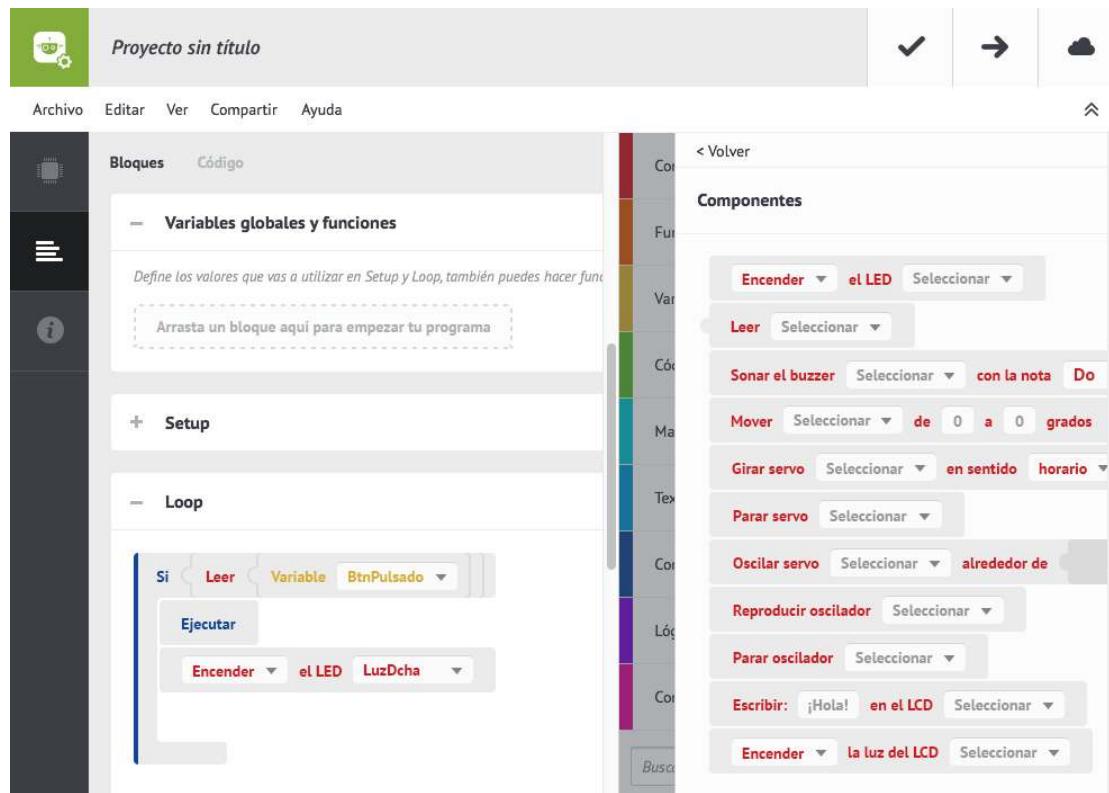
- La pestaña de hardware cuando hay placas y componentes:



- Si nos movemos a la pestaña de software:



- La pestaña de software con diferentes bloques y desplegado la pestaña que contiene los bloques:



- Si nos movemos a la pestaña de información.

The screenshot shows the 'Proyecto sin título' (Project without title) screen in the Botbloq software. The interface includes a top navigation bar with icons for save, publish, and cloud, and a menu bar with Archivo, Editar, Ver, Compartir, and Ayuda. On the left is a sidebar with icons for robot, code, and info. The main area is divided into sections: 'Información' (Information) containing fields for Nombre* (Name) and Descripción* (Description), both currently empty; 'Video' (Video) with a URL input field; 'Imagen principal*' (Main image*) with a placeholder image of an Arduino Uno board and a 'Subir imagen' (Upload image) button; 'Otras imágenes' (Other images) with two image thumbnails and 'Subir imagen' buttons; and 'Etiquetas*' (Tags) with a text input field, an 'Añadir' (Add) button, and three tags labeled 'etiqueta1', 'etiqueta2', and 'etiqueta3'. A note at the bottom states: '*Campos obligatorios para poder publicar el proyecto en Explora'.

En las siguientes capturas aparece las vistas de explora:

Explora 23.467 proyectos

Orden: Más recientes

Filtrar por:

Buscar...

Proyectos	Placas	Componentes		
<input type="checkbox"/> Compilan	<input type="checkbox"/> BQ Zum	<input type="checkbox"/> Sin componentes	<input type="checkbox"/> Joystick	<input type="checkbox"/> Servo Continuo
<input type="checkbox"/> No compilan	<input type="checkbox"/> Freaduino	<input type="checkbox"/> Ultrasonidos	<input type="checkbox"/> Zumbador	<input type="checkbox"/> Sensor de luz
<input type="checkbox"/> Proyectos de BQ	<input type="checkbox"/> Arduino UNO	<input type="checkbox"/> Bluetooth	<input type="checkbox"/> Sensores infrarrojos	<input type="checkbox"/> Potenciómetro
<input type="checkbox"/> ❤️ ¡Me gustan!		<input type="checkbox"/> Botón	<input type="checkbox"/> LCD	<input type="checkbox"/> Puerto Serie
		<input type="checkbox"/> Botonera	<input type="checkbox"/> LED	<input type="checkbox"/> Servo

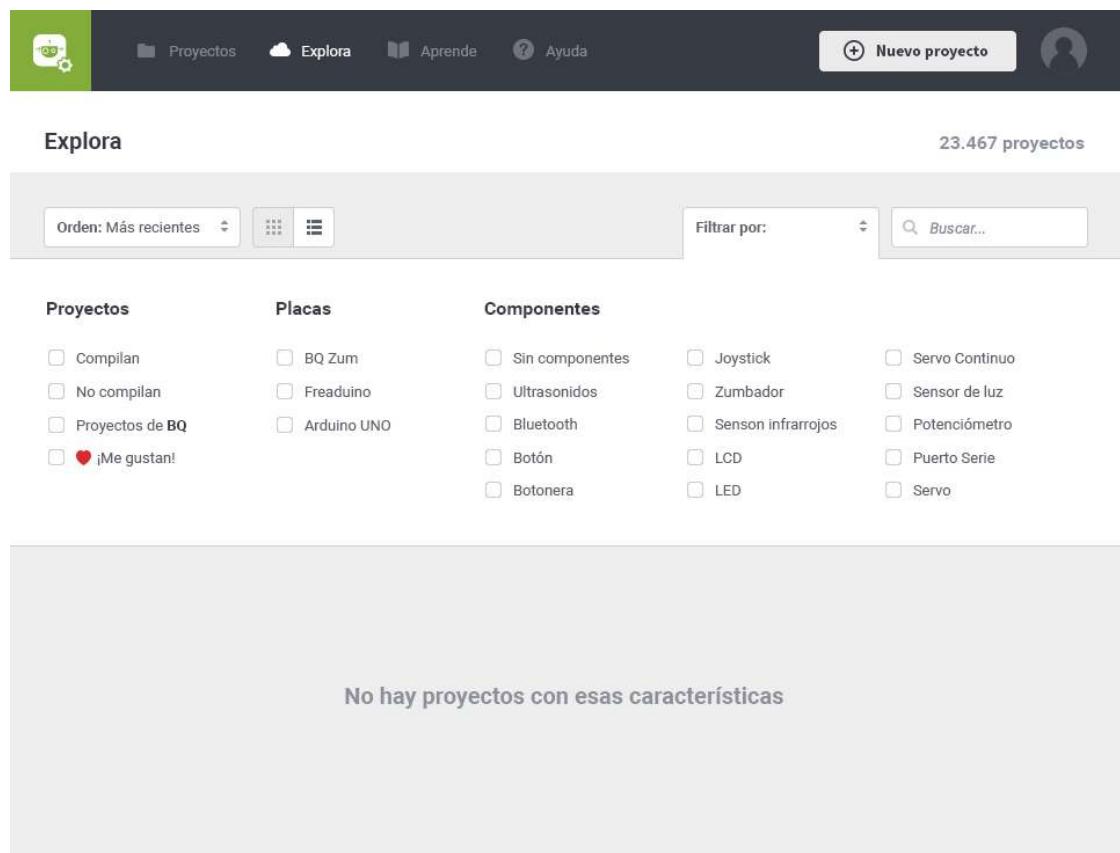
Más información

Título del proyecto con una linea d... de Username 35 21

Título del proyecto con una linea d... de Username 51 16

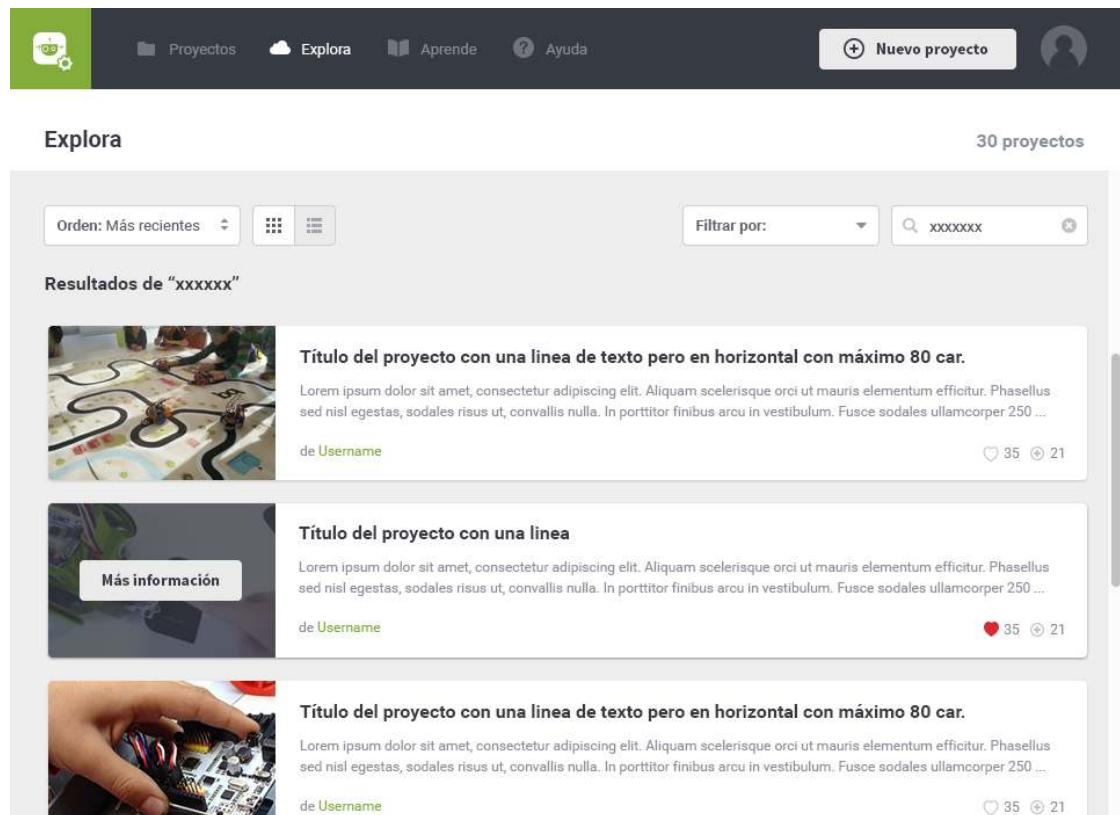
Título del proyecto con una linea d... de Username 78 54

- Los filtros de explora:



The screenshot shows the Botbloq software interface. At the top, there's a navigation bar with icons for Projects, Explore, Learn, Help, and a New Project button. Below the navigation bar is a search bar labeled "Explora" with the text "23.467 proyectos". The main area is titled "Explora" and contains three columns of filters: "Proyectos", "Placas", and "Componentes". Under "Proyectos", there are checkboxes for "Compilan", "No compilan", "Proyectos de BQ", and "¡Me gustan!". Under "Placas", there are checkboxes for "BQ Zum", "Freaduino", "Arduino UNO", "Botón", and "Botonera". Under "Componentes", there are checkboxes for "Sin componentes", "Joystick", "Zumbador", "Senson infrarrojos", "LCD", "LED", "Servo Continuo", "Sensor de luz", "Potenciómetro", "Puerto Serie", and "Servo". A message at the bottom of the search results area says "No hay proyectos con esas características".

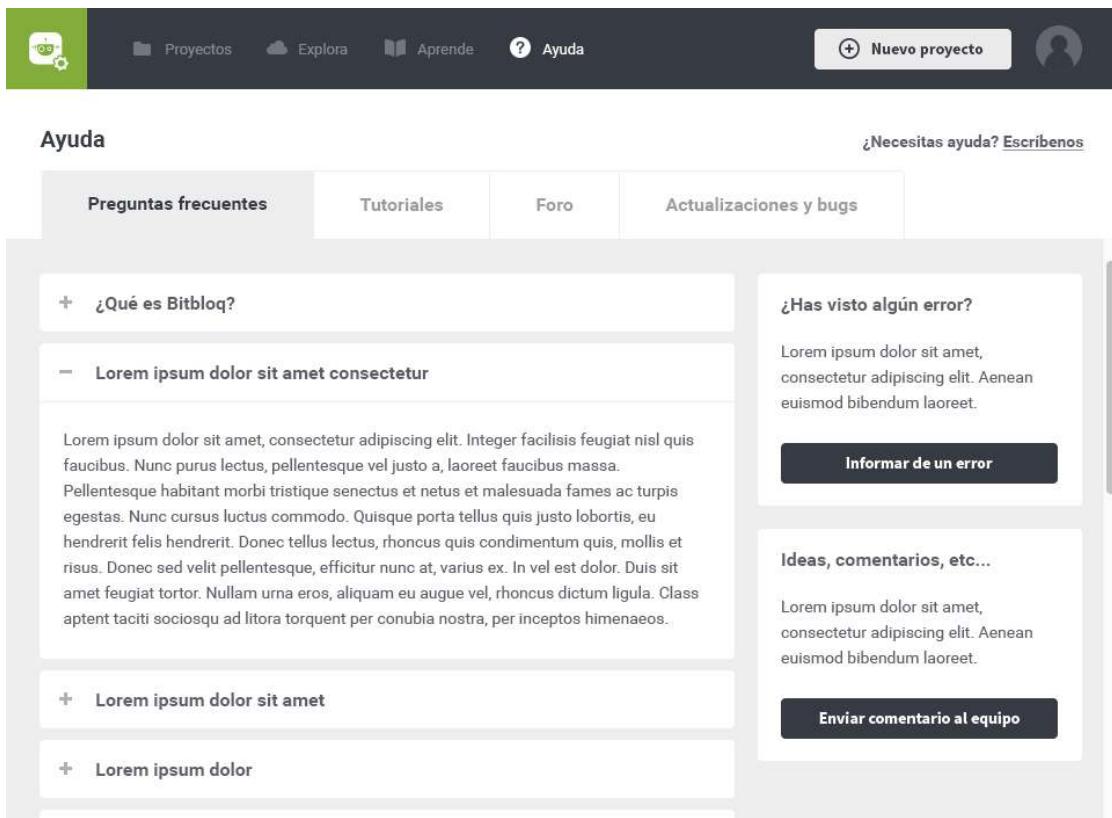
- La búsqueda de explora:



The screenshot shows the Botbloq software interface. At the top, there's a navigation bar with icons for Projects, Explore, Learn, Help, and a New Project button. Below the navigation bar is a search bar with the text 'xxxxxx'. The main area is titled 'Explora' and displays '30 proyectos'. It includes filters for sorting ('Orden: Más recientes') and searching ('Filtrar por: xxxxxxx'). The results show three projects, each with a thumbnail image, title, a brief description, the user 'de Username', and interaction counts (likes and comments).

Imagen	Título	Descripción	Usuario	Interacciones
	Título del proyecto con una linea de texto pero en horizontal con máximo 80 car.	Lore ipsum dolor sit amet, consectetur adipiscing elit. Aliquam scelerisque orci ut mauris elementum efficitur. Phasellus sed nisl egestas, sodales risus ut, convallis nulla. In porttitor finibus arcu in vestibulum. Fusce sodales ullamcorper 250 ...	de Username	35 likes, 21 comments
	Título del proyecto con una linea	Lore ipsum dolor sit amet, consectetur adipiscing elit. Aliquam scelerisque orci ut mauris elementum efficitur. Phasellus sed nisl egestas, sodales risus ut, convallis nulla. In porttitor finibus arcu in vestibulum. Fusce sodales ullamcorper 250 ...	de Username	35 likes, 21 comments
	Título del proyecto con una linea de texto pero en horizontal con máximo 80 car.	Lore ipsum dolor sit amet, consectetur adipiscing elit. Aliquam scelerisque orci ut mauris elementum efficitur. Phasellus sed nisl egestas, sodales risus ut, convallis nulla. In porttitor finibus arcu in vestibulum. Fusce sodales ullamcorper 250 ...	de Username	35 likes, 21 comments

A continuación se muestran los diseños de la pestaña de ayuda:



The screenshot shows the 'Ayuda' (Help) section of the Botbloq website. At the top, there's a navigation bar with icons for Projects, Explore, Learn, and Help, along with a 'Nuevo proyecto' (New project) button and a user profile icon.

The main area is titled 'Ayuda' and contains four tabs: 'Preguntas frecuentes' (FAQ), 'Tutoriales' (Tutorials), 'Foro' (Forum), and 'Actualizaciones y bugs' (Updates and bugs). The 'Preguntas frecuentes' tab is active.

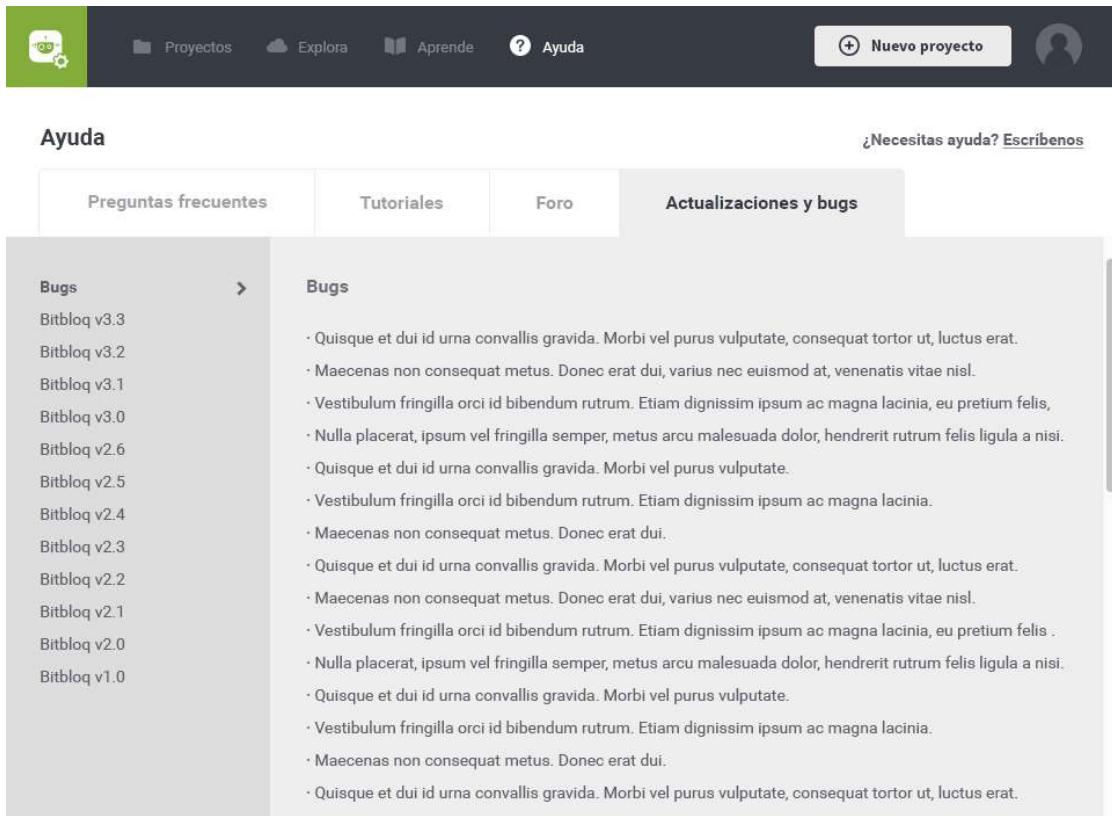
Under the FAQ tab, there are several expandable sections:

- ¿Qué es Bitbloq?**
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer facilisis feugiat nisl quis faucibus. Nunc purus lectus, pellentesque vel justo a, laoreet faucibus massa. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nunc cursus luctus commodo. Quisque porta tellus quis justo lobortis, eu hendrerit felis hendrerit. Donec tellus lectus, rhoncus quis condimentum quis, mollis et risus. Donec sed velit pellentesque, efficitur nunc at, varius ex. In vel est dolor. Duis sit amet feugiat tortor. Nullam urna eros, aliquam eu augue vel, rhoncus dictum ligula. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.
- ¿Has visto algún error?**
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.
- Informar de un error**
- Ideas, comentarios, etc...**
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet.
- Enviar comentario al equipo**

Other sections visible include 'Tutoriales' (with a 'Nuevo tutorial' button), 'Foro' (with a 'Nuevo tema' button), and 'Actualizaciones y bugs'.

The screenshot shows the 'Ayuda' (Help) section of the Botbloq website. At the top, there is a navigation bar with links for 'Proyectos', 'Explora', 'Aprende', 'Ayuda', 'Nuevo proyecto' (New project), and a user profile icon. Below the navigation bar, there are four tabs: 'Preguntas frecuentes' (FAQ), 'Tutorial', 'Foro' (Forum), and 'Actualizaciones y bugs' (Updates and bugs). The 'Tutorial' tab is currently selected. Under the 'Tutorial' tab, there is a heading 'Básicos' (Basics) followed by three tutorial cards, each with a thumbnail image and a title:

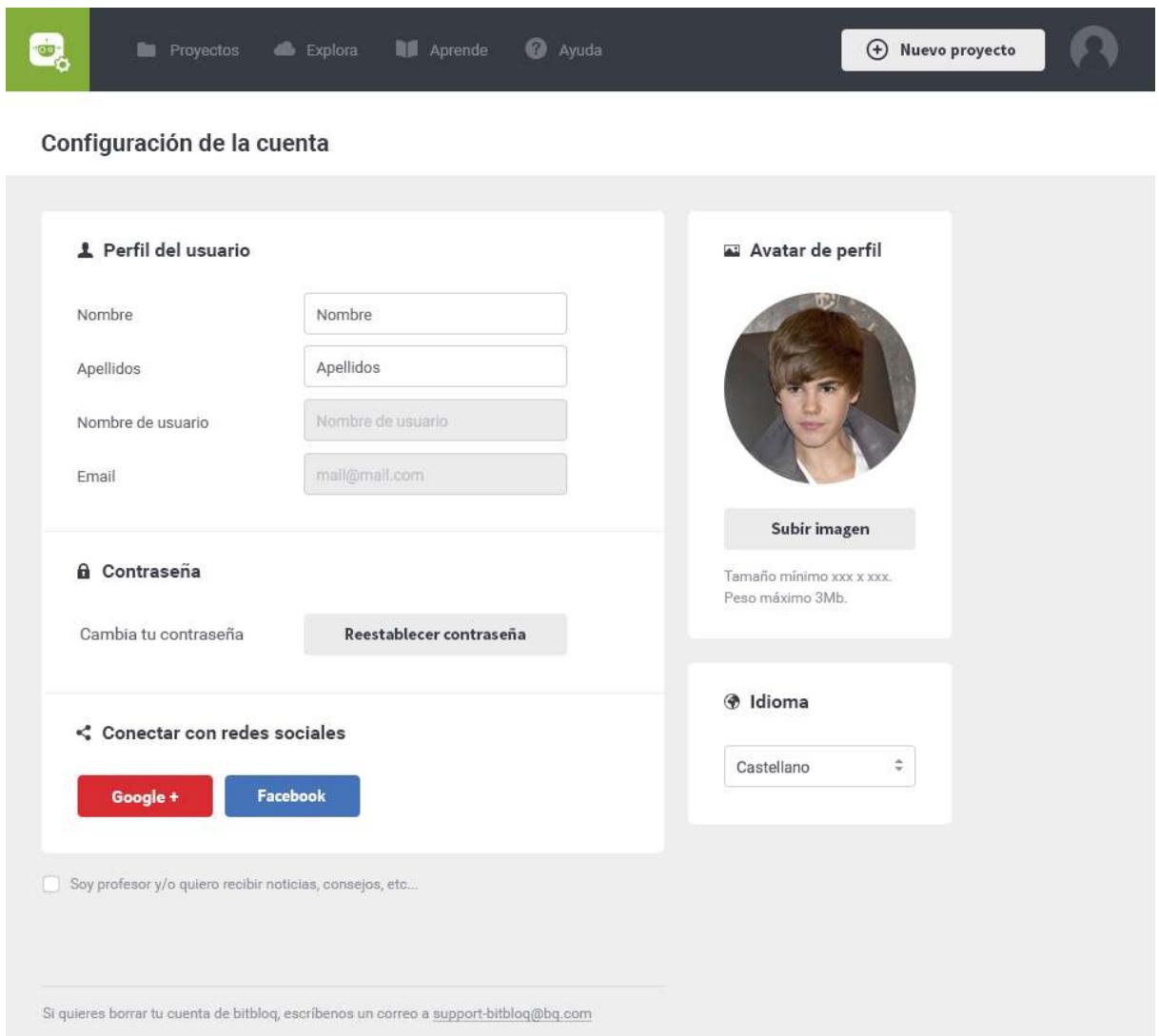
- Tutorial básico 1**: Shows a robot on a track. Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan in sapien commodo finibus. Ut arcu urna, faucibus finibus diam eu, imperdiet faucibus nisl. Ut et metus sapien. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Duis et tincidunt tortor. Sed tempor ligula a libero tempus, sit amet fermentum sem venenatis. Donec vel vestibulum velit, et rhoncus lectus.
- Tutorial básico 2**: Shows a hand connecting a component to a green circuit board. Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan in sapien commodo finibus. Ut arcu urna, faucibus finibus diam eu, imperdiet faucibus nisl. Ut et metus sapien. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Duis et tincidunt tortor. Sed tempor ligula a libero tempus, sit amet fermentum sem venenatis. Donec vel vestibulum velit, et rhoncus lectus.
- Tutorial básico 3**: Shows a close-up of hands working on a circuit board. Description: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque accumsan in sapien commodo finibus. Ut arcu urna, faucibus finibus diam eu, imperdiet faucibus nisl. Ut et metus sapien. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Duis et tincidunt tortor. Sed tempor ligula a libero tempus, sit amet fermentum sem venenatis. Donec vel vestibulum velit, et rhoncus lectus.



The screenshot shows a user interface for 'Ayuda' (Help) on the botbloq website. At the top, there are navigation links: 'Proyectos', 'Explora', 'Aprende', 'Ayuda', 'Nuevo proyecto', and a user profile icon. Below the header, there are tabs for 'Preguntas frecuentes', 'Tutoriales', 'Foro', and 'Actualizaciones y bugs'. The 'Actualizaciones y bugs' tab is selected. On the left, there's a sidebar titled 'Bugs' with a list of software versions from v1.0 to v3.3. To the right, there's a large list of bullet points under each version, which appear to be placeholder text for bug descriptions.

Bugs	Bugs
Bitbloq v3.3	<ul style="list-style-type: none">Quisque et dui id urna convallis gravida. Morbi vel purus vulputate, consequat tortor ut, luctus erat.
Bitbloq v3.2	<ul style="list-style-type: none">Maecenas non consequat metus. Donec erat dui, varius nec euismod at, venenatis vitae nisl.
Bitbloq v3.1	<ul style="list-style-type: none">Vestibulum fringilla orci id bibendum rutrum. Etiam dignissim ipsum ac magna lacinia, eu pretium felis,
Bitbloq v3.0	<ul style="list-style-type: none">Nulla placerat, ipsum vel fringilla semper, metus arcu malesuada dolor, hendrerit rutrum felis ligula a nisi.
Bitbloq v2.6	<ul style="list-style-type: none">Quisque et dui id urna convallis gravida. Morbi vel purus vulputate.
Bitbloq v2.5	<ul style="list-style-type: none">Vestibulum fringilla orci id bibendum rutrum. Etiam dignissim ipsum ac magna lacinia.
Bitbloq v2.4	<ul style="list-style-type: none">Maecenas non consequat metus. Donec erat dui.
Bitbloq v2.3	<ul style="list-style-type: none">Quisque et dui id urna convallis gravida. Morbi vel purus vulputate, consequat tortor ut, luctus erat.
Bitbloq v2.2	<ul style="list-style-type: none">Maecenas non consequat metus. Donec erat dui, varius nec euismod at, venenatis vitae nisl.
Bitbloq v2.1	<ul style="list-style-type: none">Vestibulum fringilla orci id bibendum rutrum. Etiam dignissim ipsum ac magna lacinia, eu pretium felis .
Bitbloq v2.0	<ul style="list-style-type: none">Nulla placerat, ipsum vel fringilla semper, metus arcu malesuada dolor, hendrerit rutrum felis ligula a nisi.
Bitbloq v1.0	<ul style="list-style-type: none">Quisque et dui id urna convallis gravida. Morbi vel purus vulputate.Vestibulum fringilla orci id bibendum rutrum. Etiam dignissim ipsum ac magna lacinia.Maecenas non consequat metus. Donec erat dui.Quisque et dui id urna convallis gravida. Morbi vel purus vulputate, consequat tortor ut, luctus erat.

Posteriormente aparecen las imágenes correspondientes al diseño de la configuración de cuenta de un usuario:



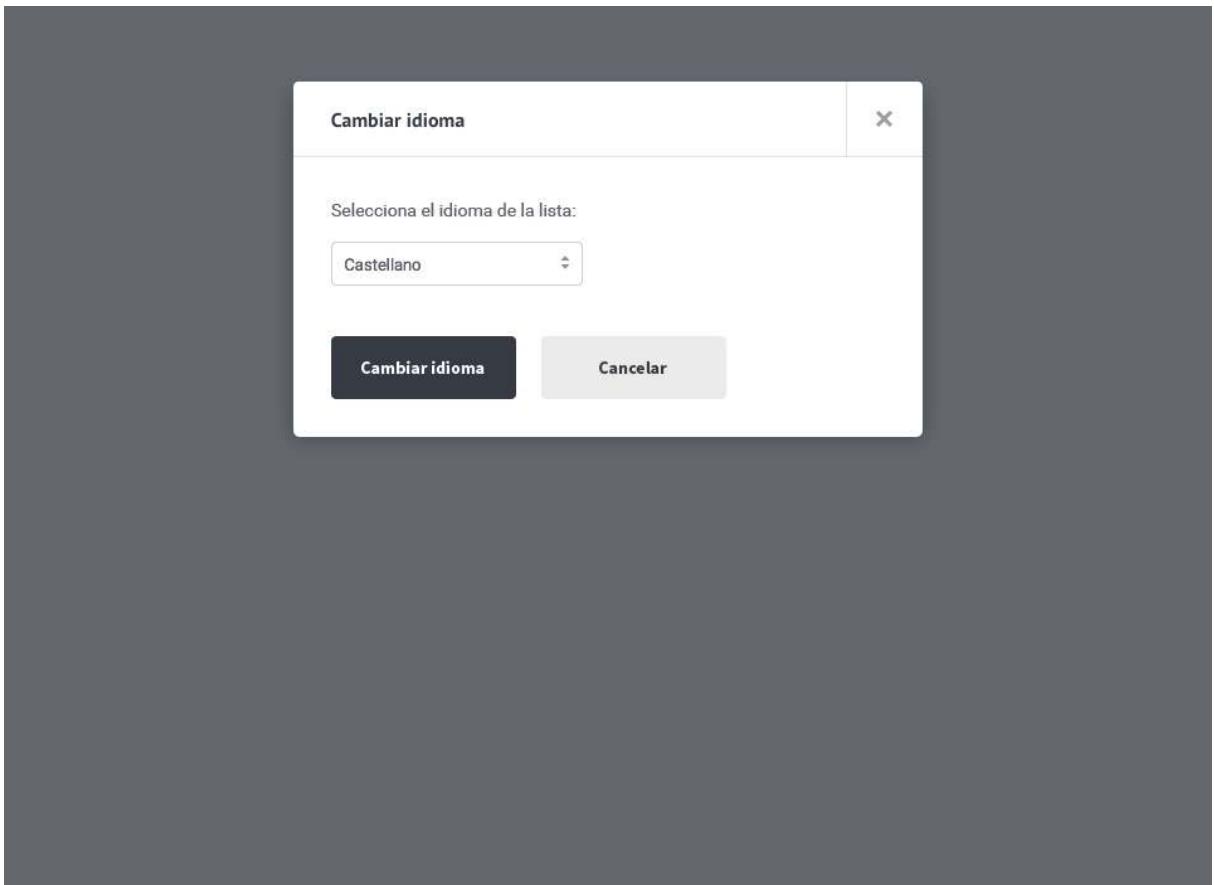
The screenshot shows the 'Configuración de la cuenta' (Account Configuration) page of the botbloq platform. At the top, there's a navigation bar with icons for Projects, Explore, Learn, Help, and a 'Nuevo proyecto' (New Project) button. On the right, there's a user profile icon.

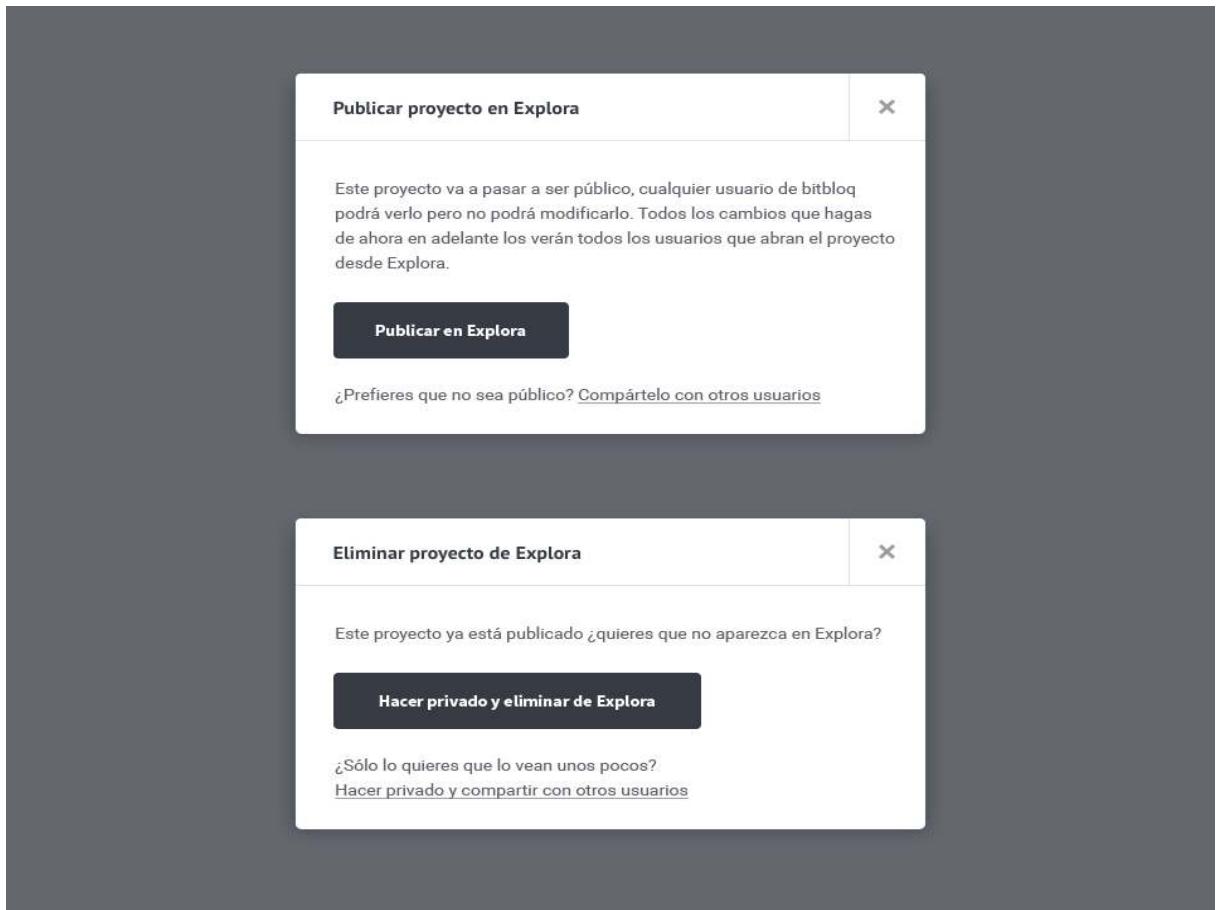
The main area is divided into several sections:

- Perfil del usuario (User Profile):** Fields for Nombre, Apellidos, Nombre de usuario, and Email.
- Avatar de perfil (Profile Avatar):** A circular profile picture of a person, with a 'Subir imagen' (Upload image) button below it. Text specifies minimum size and maximum weight (3Mb).
- Contraseña (Password):** Buttons for 'Cambia tu contraseña' (Change your password) and 'Reestablecer contraseña' (Reset password).
- Conectar con redes sociales (Connect with social networks):** Buttons for Google+ and Facebook.
- Idioma (Language):** A dropdown menu set to 'Castellano' (Spanish).

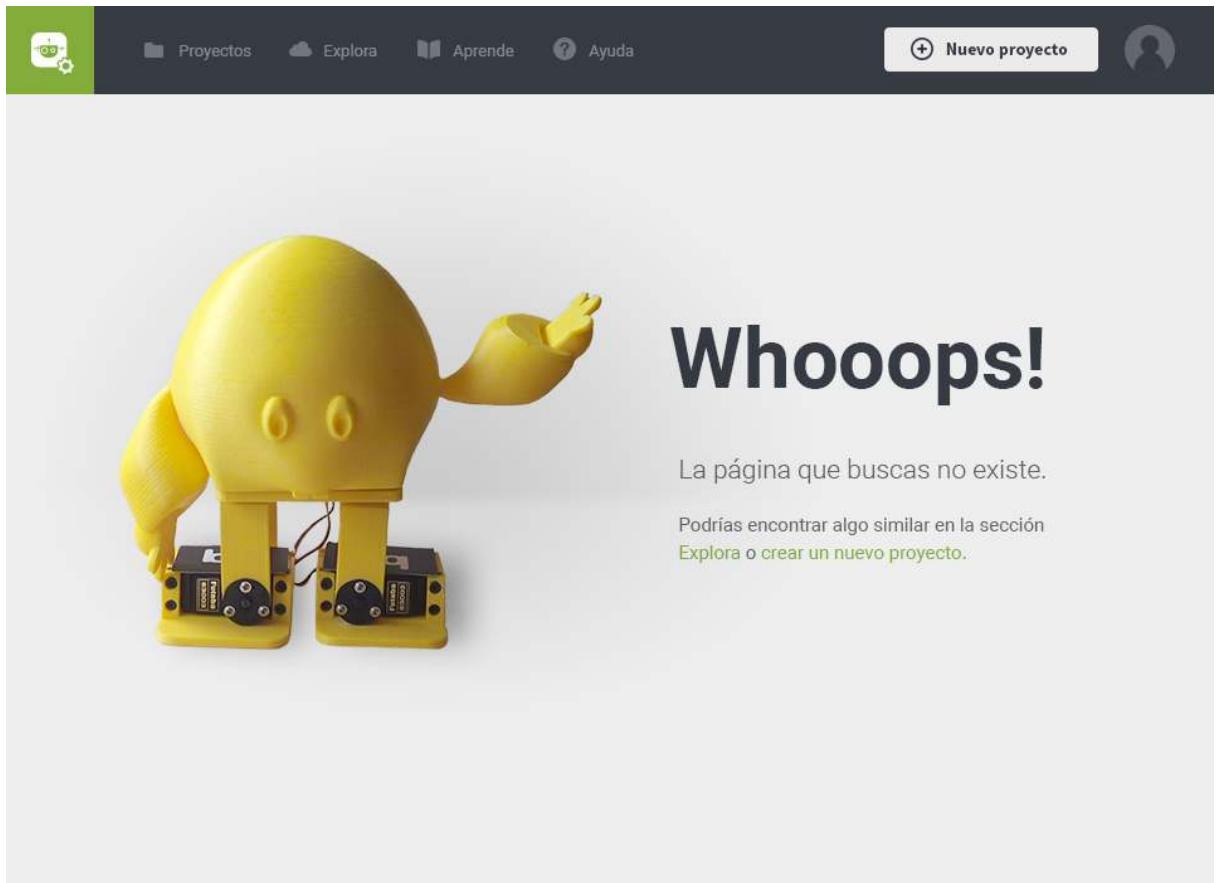
At the bottom left, there's a checkbox for 'Soy profesor y/o quiero recibir noticias, consejos, etc...' (I am a teacher and/or I want to receive news, advice, etc...). At the bottom center, a note says: 'Siquieres borrar tu cuenta de bitbloq, escríbenos un correo a support-bitbloq@bq.com'.

Seguidamente se muestran los diseños de algunos modales que aparecen en el sistema:





A continuación se muestran los diseños de las páginas de incompatibilidad y de las páginas de error:





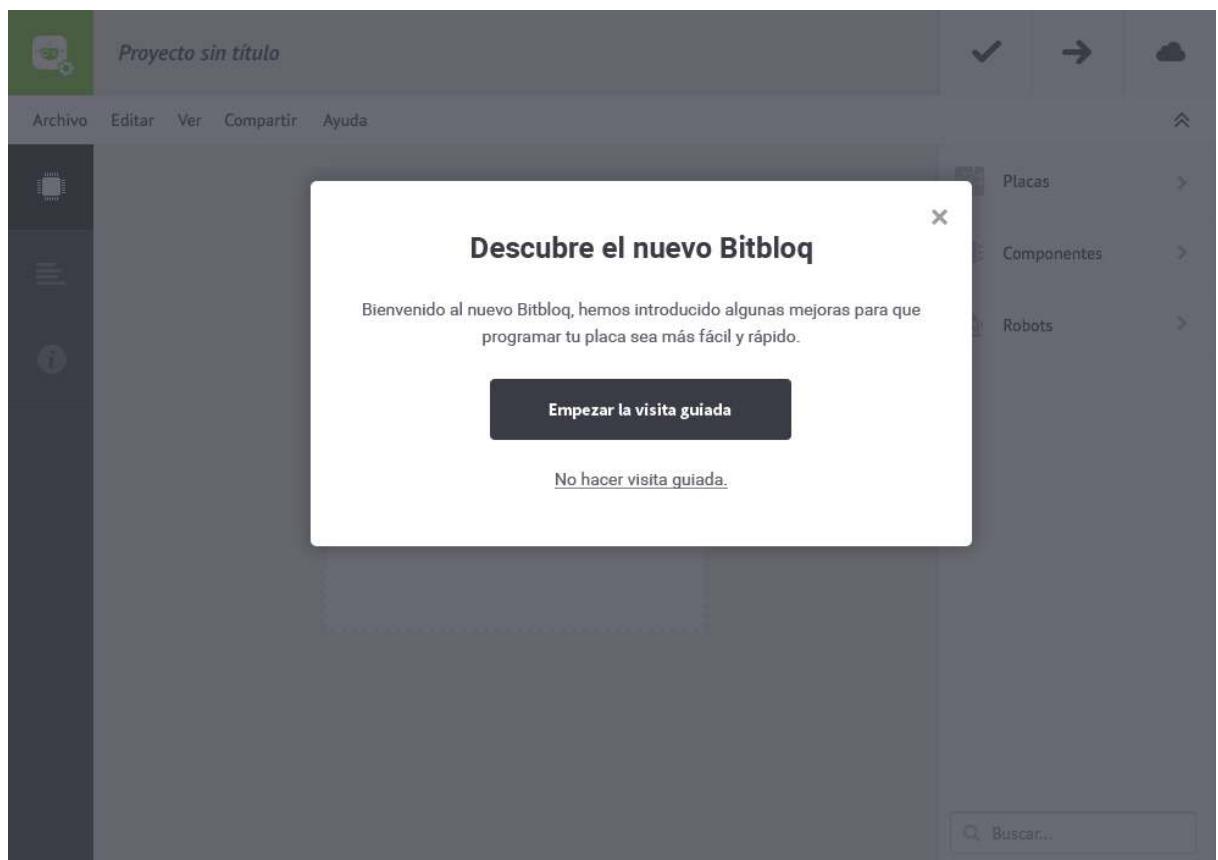
The Bitbloq landing page features a green header with the Bitbloq logo and a warning message: "¡Atención! Bitbloq sólo es 100% compatible con el navegador de escritorio Chrome. Si entras desde este navegador es posible que haya cosas que no funcionen correctamente." Below this, a link "Continuar de todos modos" is visible. The main content area shows a desktop monitor, a tablet, and a smartphone all displaying the Google Chrome logo, indicating cross-device compatibility.

Descarga Google Chrome

Un navegador web rápido y gratuito para tu ordenador, tu teléfono y tu tablet.

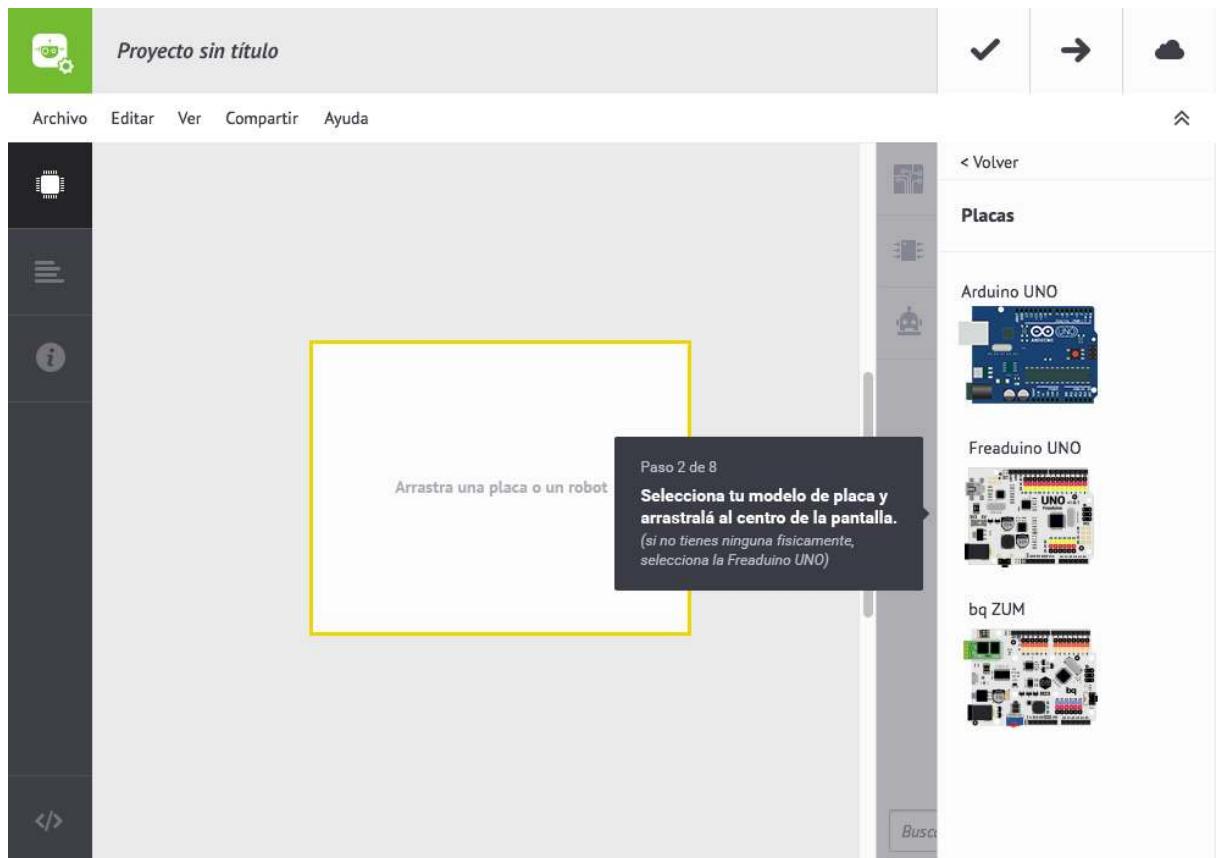
[Descargar Google Chrome](#)

Por último aparecen los diseños del tour:



243

Proyecto Financiado por el Centro de Desarrollo Tecnológico Industrial (CDTI).
Expediente IDI-20150289
Cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER) a través del Programa
Operativo Plurirregional de Crecimiento Inteligente 2014-2020



5.8 Anexo VIII. Tareas realizadas por el equipo de desarrollo

Ver E.2.3 Anexo VIII. Tareas realizadas por el equipo de desarrollo

5.9 Anexo IX. 100 proyectos de robótica con Bitbloq y Arduino

A finales de 2015 Ernesto Martínez de Carvajal Hedrich, escritor entre otras muchas facetas edita y publica el libro “100 PROYECTOS DE ROBÓTICA CON BITBLOQ Y ARDUINO”, un libro cargado de explicaciones sobre sensores, actuadores y robótica en general y decide utilizar la plataforma de Bitbloq para crear 100 proyectos cargados de detalle e ingenio para sacarle el mayor partido a los componentes.

http://www.todostuslibros.com/libros/100-proyectos-de-robotica-con-bitbloq-y-arduino_978-84-608-4317-7