

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



CDTI Centro para el Desarrollo Tecnológico Industrial



UNIÓN EUROPEA

Fondo Europeo de Desarrollo Regional (FEDER)

Una manera de hacer Europa

ENTREGABLE E.2.2. ESPECIFICACIÓN DE REQUISITOS IDENTIFICANDO PERIFÉRICOS QUE DEBE SOPORTAR LA APLICACIÓN A DESARROLLAR

RESUMEN DEL DOCUMENTO

Requisitos técnicos de la aplicación. Módulos en los que se divide el proyecto y su descripción (frontend, backend, librería de bloques, aplicación para compilación y carga de programas por le puerto serie), descripción de las infraestructuras software necesarias a desarrollar así como sus requisitos básicos para construirlas.

February 3, 2016

Contents

1 Aspectos generales de la plataforma	3
1.1 Objetivos	3
1.2 Requisitos	4
1.3 Módulos de la plataforma	5
1.3.1 Módulo de programación visual mediante bloques.	5
1.3.2 Módulo de software para la compilación y programación de hardware	6
1.3.3 Infraestructura completa en la nube.	7
1.3.4 Infraestructura para test unitarios y end-to-end.	9
1.3.5 Metodologías en el desarrollo de software	10
2 Requisitos específicos de la plataforma	23
2.1 Flujos de navegación principales	24
2.2 Espacio de trabajo de usuario (Construye)	25
2.2.1 Espacio de trabajo con bloques	25
2.2.2 Zona de elección de bloques	26
2.2.3 Código autogenerado	26
2.2.4 Información del proyecto	27
2.3 Gestión de proyectos de usuario	27
2.3.1 Sistema de login y registro en la plataforma	30
2.3.2 Proyectos privados del usuario (Mis proyectos)	31
2.3.3 Espacio público y compartido de proyectos (Explora)	32
2.4 Software de programación de hardware (Serial uploader)	36
2.4.1 Casos de uso	41
2.5 Gestión de perfil de usuario	42
2.5.1 Perfil de usuario	42
2.5.2 Permisos de usuario	43
2.6 Otros aspectos de la plataforma	44
2.6.1 Wizard	44
2.6.2 Consola	45
2.6.3 Sistema de ayuda y reporte de incidencias	48

2.6.4	Visión y compatibilidades del producto	49
-------	--	----

1 Aspectos generales de la plataforma

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 microcontroladoras. 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.

1.1 Objetivos

Como objetivo final, el entorno de programación sirve al usuario para interconectar el mundo de la robótica y la electrónica junto al de su programación compleja, abstrayendo todo lo posible esta última parte. Esto, se consigue siguiendo una metodología similar a *Scratch* que utilice bloques visuales de código que a modo de puzzle consiga crear un programa para cargar en una placa.

En concreto, la herramienta está enfocada para enseñar a programar a niños de edades comprendidas entre 8 y 12 años que no tengan conocimientos de programación y quieran adentrarse en el mundo de la electrónica y robótica de una forma fácil, visual, divertida y aplicada al mundo real. Siguiendo esta filosofía, el software debe ser capaz de aportar un entendimiento de la lógica de la programación de una forma más visual e interactiva ya que es más fácil para ellos guiarse por la intuición con una sintaxis simple y significativa. También se quiere enseñar a aprender de los errores, al no funcionar algo como lo habían planeado, esto les enseña a crear otra estrategia para poder abordar el problema y conseguir la solución o el objetivo definido.

Desde el punto de vista de plataforma comunitaria, el aprendizaje en grupo resulta más efectivo, ayudando a entender el trabajo en equipo y a compartir tu trabajo en una comunidad en la que la enseñanza es recíproca.

Queremos desarrollar un software para enseñar a los niños a entender la lógica de la programación de una forma más visual e interactiva ya que es más fácil para ellos guiarse por la intuición con una sintaxis simple y significativa.

1.2 Requisitos

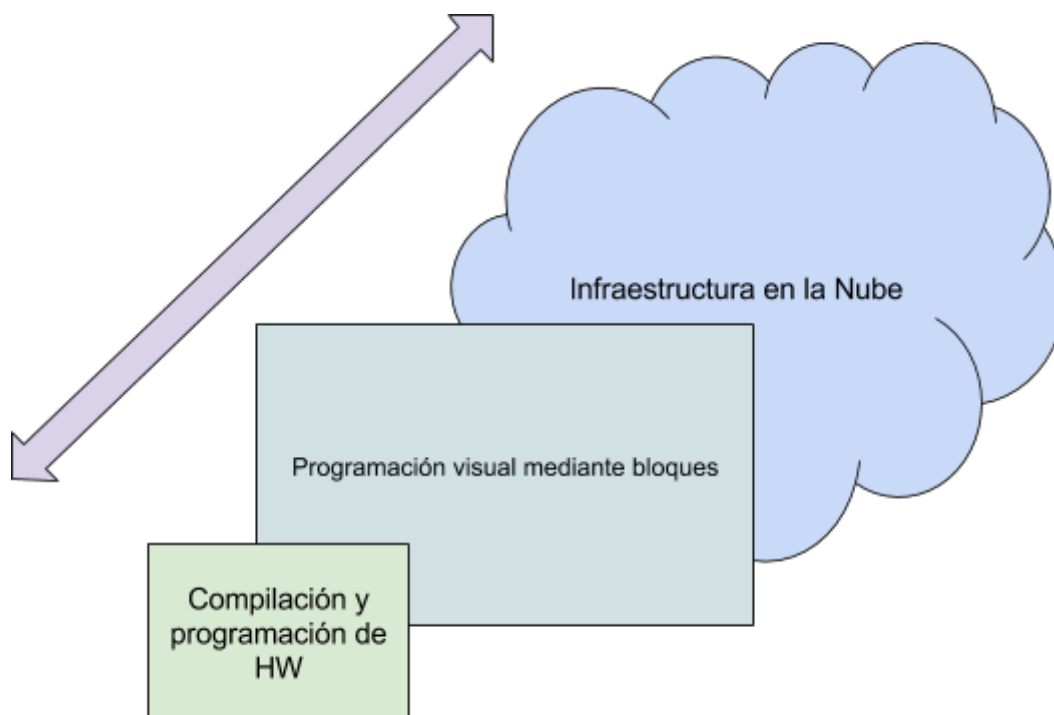
Así mismo, uno de los requisitos principales para el entorno de programación es usar plataformas modernas que permitan ‘romper la barrera de entrada’ al uso de este tipo de tecnologías. La plataforma debería tener tres características principales; un “suelo bajo” para poder programar fácilmente, unas “paredes amplias” para poder programar diferentes componentes y un “techo alto” para poder crear programas más complejos. También crear una comunidad donde los usuarios puedan compartir sus proyectos, dudas, necesidades a los demás y conseguir un apoyo mutuo y recíproco para conseguir avanzar en conocimiento y en inspiración.

Las necesidades principales son:

- Poder crear un programa de código mediante bloques visuales
- Tener una parte visual de código para ver la comparación con los bloques visuales
- Poder verificar ese programa
- Poder cargar el programa en la placa
- Tener una comunidad de usuarios para exponer y compartir proyectos
- Acceso a una parte de aprendizaje donde se obtengan conocimientos sobre programación y electrónica
- Poder gestionar mis proyectos
- Poder exportar un proyecto a código arduino

El software debe constar de tres módulos diferenciados de cara al usuario.

1. Módulo de programación visual mediante bloques visuales.
2. Módulo auxiliar de software para la compilación y programación de las placas microcontroladoras.
3. Infraestructura completa de gestión de proyectos de usuario en la nube.



1.3 Módulos de la plataforma

1.3.1 Módulo de programación visual mediante bloques.

La plataforma debe ser capaz de modelar, mediante bloques visuales, fragmentos de código válidos que configuren programas para ser cargados directamente en placas microcontroladoras. Estos bloques engloban y categorizan tanto fragmentos de lógica de programación como código necesario para configurar y modelar componentes electrónicos.

Como resultado final, al procesar la combinatoria de los bloques conectados, un fragmento de código resultante será compilado y cargado en el hardware directamente.

Categorización de bloques La categorización de bloques puede agruparse en tres niveles, en base a la complejidad que abstraen de cara al usuario:

- **Bloques que modelan componentes hardware.**

En primer lugar tendremos un nivel básico, constituido por bloques de alto nivel que encapsulan las tareas más comunes a realizar por un robot, haciendo muy sencilla la secuenciación de las mismas, destinado por tanto a usuarios sin conocimientos (p.e.: niños).

- **Bloques que permiten escritura avanzada de código.**

Por último se permitirá el acceso a programación a bajo nivel para aquellos usuarios que deseen controlar cada una de las variables que intervienen en las tareas realizadas por el robot. Este tipo de programación se realizará mediante código en un lenguaje diseñado al efecto, y estará destinado a personas con un conocimiento considerable de robótica y programación (p.e.: investigadores, estudiantes de doctorado, graduados en Ingeniería Industrial o Informática, etc.).

1.3.2 Módulo de software para la compilación y programación de hardware

Pieza fundamental de software encargada de comunicar el entorno de programación con el hardware para cargar los programas en las placas.

Mediante este módulo se puede controlar todo el flujo de compilación y programación de una placa hardware. Las funciones de este módulo incorporan los siguientes requisitos y funcionalidades:

- Proceso de compilación de código autogenerado previo a la programación en la placa.
- Flujo de programación completo y consistente.
- Control de conexiones.
- Detección automática de placa y puerto.
- Información al usuario de casos de error en la conexión.
- Monitor serie para mostrar información serializada entre el entorno y el hardware.
- Idempotencia de cada operación de programación. Control de apertura y cierre de los puertos.

Un proceso genérico de programación de hardware debería seguir una serie de pasos atómicos para poder finalizar el proceso correctamente de cara al usuario. Desde el entorno de programación se debe controlar todo el flujo con el módulo externo de carga de compilaciones en la placa.

1.3.3 Infraestructura completa en la nube.

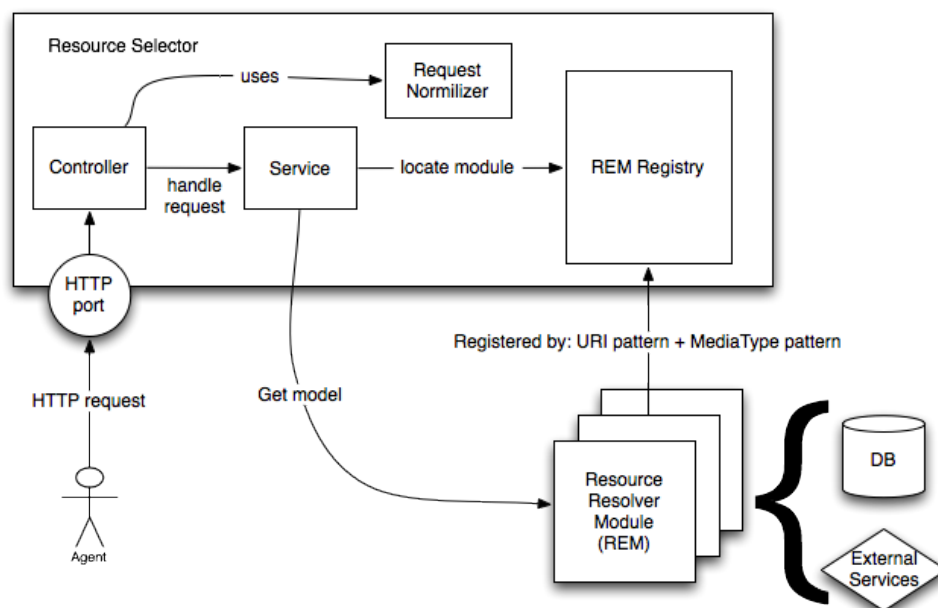
Este módulo viene representado en la aplicación como un espacio de trabajo completo, donde el usuario puede gestionar el ciclo de vida de sus proyectos y gestionarlos desde cualquier lugar mediante tecnologías web gestionadas a través de la Nube.

El objetivo principal de aplicar este modelo es crear una comunidad de usuarios entorno a proyectos DIY con un componente social fuerte y establecer un marco de referencia dentro de las comunidades maker online.

La gestión de proyectos en la nube, permite compartir proyectos entre usuarios y la publicación de manera pública y divulgativa de este tipo de proyectos al resto de la comunidad.

La infraestructura implementada detrás de la plataforma debe poder gestionar:

- Cuentas y perfiles de usuarios, contemplando autorización y autenticación de usuario.
- Recursos de usuario, abarcando desde los proyectos de usuario hasta los recursos asociados con los proyectos, como imágenes y otros recursos multimedia. Estos contenidos deben ser expuestos a través de una API pública.



- Posibles servicios en la nube como compiladores de código online y servicio de notificaciones al usuario.

Infraestructura completa del servidor En todo producto completo, basado en tecnologías web, es importante diferenciar dos términos técnicos totalmente diferentes: el front-end y el back-end. Estos términos se refieren a la separación de intereses entre la capa de presentación y la capa de acceso a los datos necesarios para el producto.

El concepto de front-end es utilizado para definir la parte del software que interactúa con los usuarios. Mientras que el término back-end hace referencia a la gestión de los datos y recursos del sistema.

Separar en el producto estos dos términos ayuda a tener una mayor abstracción y un mejor mantenimiento por separado de las diferentes parte del sistema.

Este apartado será dedicado a detallar cómo debe ser y comportarse la infraestructura de la parte del back-end.

Se quiere utilizar una plataforma, que sea completa y fácil de usar, en la que apoyarse para el desarrollo y lanzamiento de la aplicación.

Plataforma Back-end como servicio (BaaS)

BaaS es un modelo relativamente reciente en la computación en la nube. Este es el modelo sobre el que debe estar construido la plataforma desarrollada y que provee de un sistema independiente que podría ser migrado a otros proveedores de servicio sin afectar al servicio (API) que expone al cliente.

La principal ventaja de estos sistemas radica en la utilización de microservicios, cada uno de estos con su propia API que pueden ser incorporados de forma atómica a una aplicación cliente.

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

La importancia de la calidad del software Es una de las partes más importantes del producto, es necesaria e imprescindible para garantizar su robustez y funcionamiento.

Históricamente la calidad del software era delegada a último lugar debido al uso de metodologías en cascada donde la calidad formaba parte al final del proceso, además de ello al no ser un proceso de desarrollo iterativo hacía que cada modificación del proyecto ya en curso supusiera una dilatación de tiempos que provocaba que la última parte del proceso, es decir la calidad, se viera mermada en tiempo provocando una disminución de la calidad en el producto final.

Debido a ello se propone utilizar metodologías ágiles en el desarrollo en las cuales juega un papel importante la calidad del software.

Características del desarrollo ágil en la calidad del software. Una característica importante de desarrollo ágil es que este se realiza en ciclos cortos de forma reiterativa en los cuales siempre está presente el proceso de calidad, de este modo se consigue que en cada iteración se realice el proceso de calidad.

1.3.5 Metodologías en el desarrollo de software

Dentro de la ingeniería de software existen diferentes metodologías de desarrollo en las cuales encontramos algunas en el que núcleo de ellas son las pruebas para una mayor calidad del software, como por ejemplo:

- **Desarrollo guiado por pruebas “TDD”**

El desarrollo guiado por pruebas de software o “TDD” Test-drive development como su nombre indica, es un proceso en el que el desarrollo es guiado desde las pruebas.

Para ello en primer lugar se escriben las pruebas “Test First Development” y en segundo lugar refactorizar el código “Refactoring”.

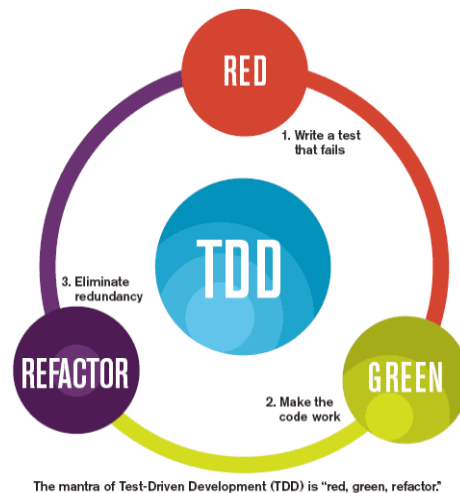
TDD tiene su enfoque principalmente en pruebas de código unitarias “Unit Testing” en el que se prueba una pequeña parte de código, una de sus principales ventajas es lograr un código limpio y funcional.

El propósito general es que los requisitos y especificaciones sean traducidas a test unitarios de cada una de las nuevas funcionalidades que se vayan a realizar.

Se ha de tener en cuentas que para lograr realizar el desarrollo guiado por pruebas, Bitbloq debe de ser lo suficiente flexible para permitir que sea probado automáticamente.

Ciclo de desarrollo conducido por pruebas

- Elegir un requisito de mayor importancia y de fácil implementación.
- Escribir un test que pruebe la funcionalidad deseada.
- Comprobar que **el test falla**.



- Dotar de una mínima funcionalidad al test para que este pase.
- Comprobar que el test ha pasado.
- Comprobar que todos los test automatizados no fallan.
- Actualización de las pruebas de requisito.

- **Desarrollo guiado por comportamiento “BDD”**

BDD o “Behaviour-Driven Development” es una evolución de TDD debido a que las pruebas siguen guiando el desarrollo, BBDD surge para crear pruebas que integren las reglas de negocio con el lenguaje de programación, centrándose en el comportamiento del software.

Además de ello mejora la comunicación entre los equipos de desarrollo y pruebas, aumentando el intercambio de conocimientos entre ellos.

BBDD se centra en un lenguaje común que integre la parte técnica y de negocio, y que desde ese lenguaje común arranque el testing y a partir de ahí el desarrollo.

En el lenguaje común se escriben las pruebas de aceptación usando las historias de usuario o funcionalidades “features” a implementar.

Las historias de usuario antes nombradas son escritas por la parte de negocio y son responsabilidad del product owner. Las features son escritas de forma entendible por el negocio usando un lenguaje denominado DSL (Domain-Specific Language) muy cercano al lenguaje natural..

El lenguaje DSL generalmente consta de 5 sentencias:

Feature: Descripción concisa de lo que se desea con el fin de lograr un valor añadido al negocio, como por ejemplo: “Quiero que los usuarios se puedan registrar”

Scenario: La situación en la que se va realizar la feature, como por ejemplo: “Un nuevo usuario se desea registrar “

Given: Dada cierta condición previa “el usuario con nombre x va a registrarse”

When: Sí la acción es favorable “el nombre del usuario no contiene caracteres extraños”

Then: Si la acción es favorable, se produce el objetivo “el usuario se logea”

Pruebas de verificación y calidad del software Para la verificación se realizan diferentes test para asegurar la calidad del software, entre ellas podemos destacar las siguientes:

Test unitario

Los test unitarios o pruebas unitarias sirven para comprobar una parte de código o función, en un primer momento se comprueba una parte del código o módulo para asegurar que funciona por separado. Lo principal es escribir pruebas unitarias por cada función o método de forma que cada una sea independiente de las otras.

Una prueba unitaria debe tener las siguientes características:

- Automatizable: No requiere una intervención manual.
- Completas: Debe cumplir toda la funcionalidad del código.

- Reutilizable: Es conveniente que se pueda utilizar en otras pruebas o bloques de código.
- Independientes: La ejecución de la prueba no debe afectar a las otras.

Test funcional “e2e”

Un test funcional, o test de fin a fin “end to end \rightarrow e2e” son test que prueban una funcionalidad completa.

En este caso lo que importa es que se cumpla una funcionalidad, no se comprueba el código, se comprueba a nivel de usuario que el software cumple la funcionalidad.

Un ejemplo de prueba funcional podría ser la funcionalidad de registro de un usuario, todo el proceso de principio a fin “end to end \rightarrow e2e”.

Pruebas de Integración

En las pruebas de integración se realizan las comprobaciones de la interacción entre dos unidades o más de software, es decir que todos los componentes de la aplicación funcionan correctamente en su conjunto.

Por ejemplo se realizaría la comprobación de la conexión con la bbdd, la conexión a los servidores de recursos, el envío de emails, etc.

Las pruebas de integración son dependientes del entorno en el que se ejecutan, por lo que su fallo puede deberse al proyecto o al entorno en el que se ejecuta.

Pruebas Manuales

Las pruebas manuales son una parte importante de las pruebas, todas las pruebas posibles se deben automatizar para una mayor agilidad, pero mientras que estas no sean automatizadas se debe comprobar manualmente, además hay pruebas que por su complejidad puede dar el caso de no poder ser automatizadas.

También se debe comprobar manualmente el diseño de la interfaz, es decir, los colores, tamaño de los componentes, usabilidad y adaptación a los diferentes tamaños en los que se visualiza.

Pruebas de sistema

Las pruebas de sistema engloban todo tipo de pruebas encargadas de probar todo el sistema de software completo e integrado desde el punto de vista de requisitos de aplicación, entre las pruebas del sistema se encuentran las siguientes:

- Pruebas de carga: Las pruebas de carga son las encargadas de comprobar el rendimiento del sistema, con ellas por ejemplo comprobamos la respuesta del sistema a un determinado número de peticiones.
- Pruebas de estrés: Las pruebas de estrés son también para comprobar el rendimiento del sistema, pero a diferencia de las pruebas de carga se centran en someter al software a situaciones extremas e intentar que el sistema caiga para ver su comportamiento, tiempo en recuperarse o de tratar errores.

Pruebas de regresión

Las pruebas de regresión son todo tipo de pruebas como las nombradas anteriormente. La idea de las pruebas de regresión es tener seleccionadas las pruebas más relevantes para que estas sean pasadas siempre que se realice cualquier modificación. A medida que un proyecto avanza la batería de pruebas se incrementa considerablemente y es imprescindible realizar una selección de las más relevantes para que estas, las pruebas de regresión, sean las que siempre se pasan con cualquier modificación o prueba de integración.

Proceso de certificación El proceso de certificación de la calidad del software seguirá una metodología iterativa en ciclos cortos de desarrollo.

El proceso de certificación a seguir en líneas generales sería el siguiente, en primer lugar se realiza un análisis de requerimientos y especificaciones de las nuevas funcionalidades o mejoras a implementar, seguidamente se diseñan los casos de pruebas para posteriormente automatizar.

Cuando se cierra el ciclo de desarrollo para certificar su funcionamiento se crea una build de pruebas indicando la versión a certificar, con la build creada se ejecutan todos los casos y si estos son satisfactorios la certificación ha pasado correctamente.

Detalle de las fases del proceso de certificación:

Fase de Análisis de requerimientos y especificaciones

En primer lugar se tomarán las especificaciones de requisitos y estas se incluirán en la herramienta de gestión a utilizar.

El análisis de requerimientos se gestiona junto con el product owner o encargado del proyecto, con el se tomará nota de todos los requerimientos necesarios así como las funcionalidades que se han de implementar con dichos requerimientos.

En las especificaciones se debe detallar fielmente las necesidades y restricciones del sistema o funcionalidad a implementar.

Diseño de casos de prueba

Una vez realizado el análisis y requerimiento de las especificaciones se realizan los distintos casos de prueba para asegurar que las especificaciones se cumplen.

Ejecución de pruebas

Creados los casos de pruebas, estos se deben analizar y determinar si se deben automatizar en pruebas unitarias, funcionales, de sistema.

Cuando ya han sido automatizados se ejecutan para comprobar su correcto funcionamiento.



Plan de pruebas

El plan de pruebas es la forma de organizar las pruebas para cada entorno a probar, navegador y sistema operativo. De esta forma se realizan pruebas comunes o para un entorno específico o navegador específico.

Build

Dentro del proceso de certificación una buena forma de tener un control es creando una build, es decir crear una versión o nomenclatura por cada ciclo de desarrollo.

De esta forma se tiene un control de versiones.

Herramientas para su gestión Para llevar un seguimiento del proceso de calidad, es necesario disponer de diferentes herramientas. En este caso se hacen uso de

dos herramientas: Jira, Testlink. Gracias a Jira, se puede llevar un seguimiento de las tareas que se van completando y de las tareas que están pendientes de realizar, y gracias a Testlink, podemos hacer una gestión y seguimiento de los test que se van definiendo.

Issues

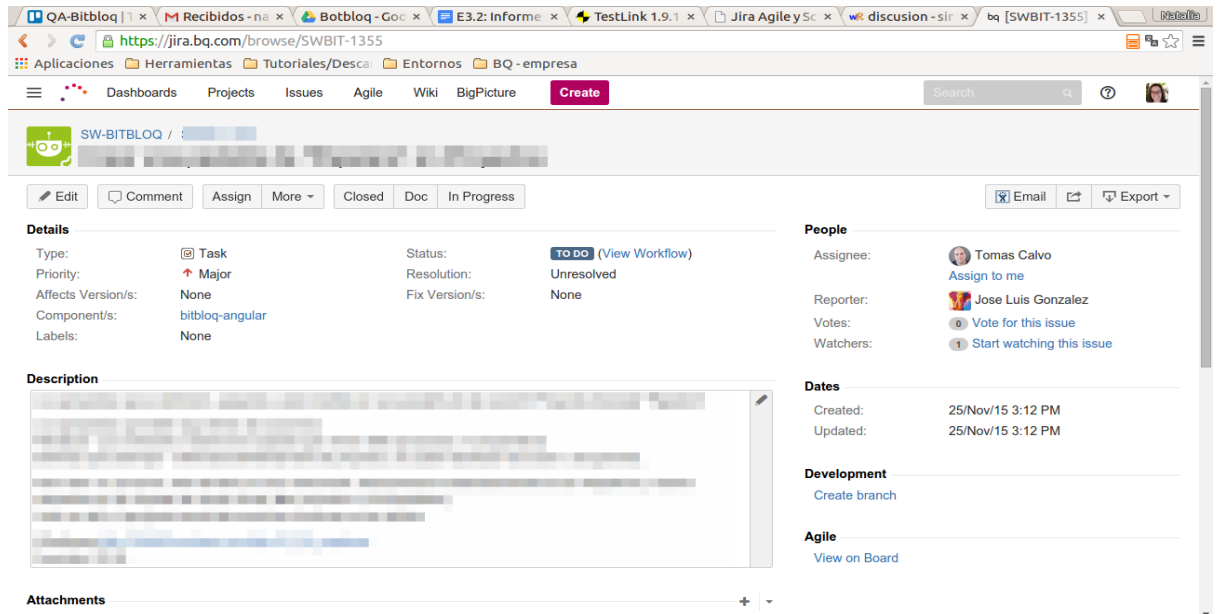
Comenzando por lo más básico, JIRA es una herramienta software para la gestión de proyectos, que permite el seguimiento de incidencias. Originalmente nació para gestionar incidencias, pero en los últimos años ha evolucionado, aunque la herencia originaria como gestor de incidencias se nota por toda la herramienta. En cualquier caso, es hoy, en gestión de proyectos, de las más populares, muy del estilo a otras como Redmine, pero de pago.

Un punto a favor de la herramienta es que se le pueden añadir complementos, como por ejemplo JIRA Agile, que permite la gestión de proyectos ágiles.

Dentro de la herramienta se pueden crear:

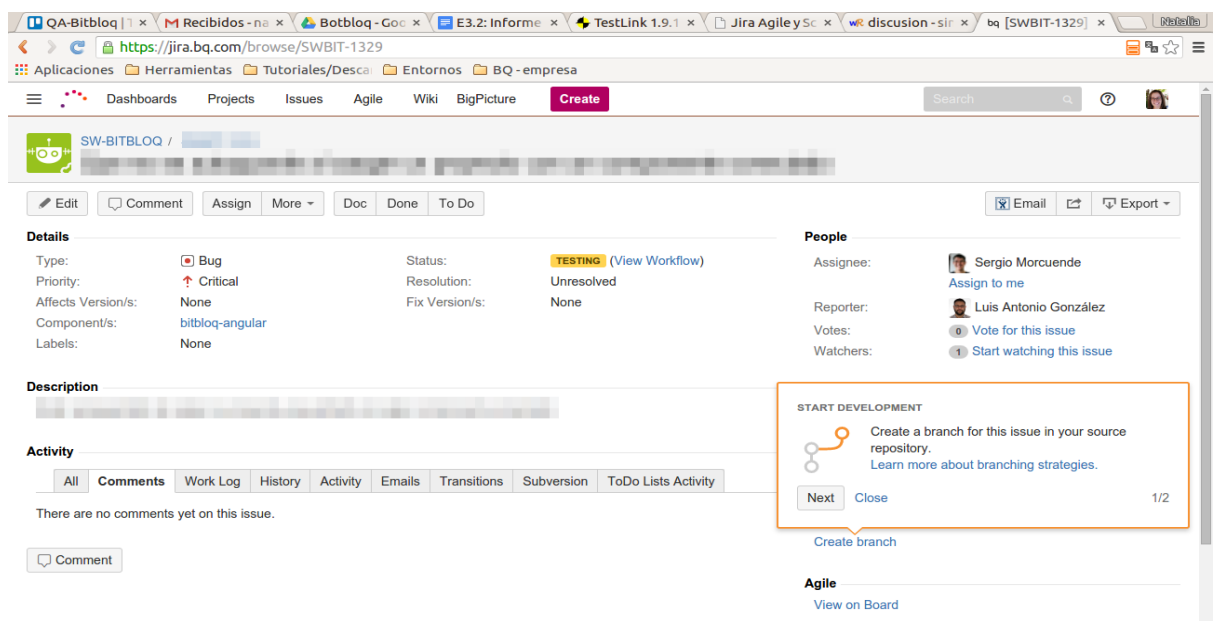
- Tarea

Una tarea recopila la información de un nuevo requisito, es decir, nueva funcionalidad que se quiere añadir a la aplicación. Dentro de la especificación tendrá que incorporar todos los detalles del requisito añadir. En la imagen de abajo, se puede ver un ejemplo de tarea.



- Bug

Un bug o error recopila la información de algo que no funciona correctamente dentro de la aplicación. Como en el caso de la tarea, en la especificación se debe detallar cómo reproducir el error y cómo debería funcionar correctamente. En la imagen de abajo, se puede ver un ejemplo de bug.



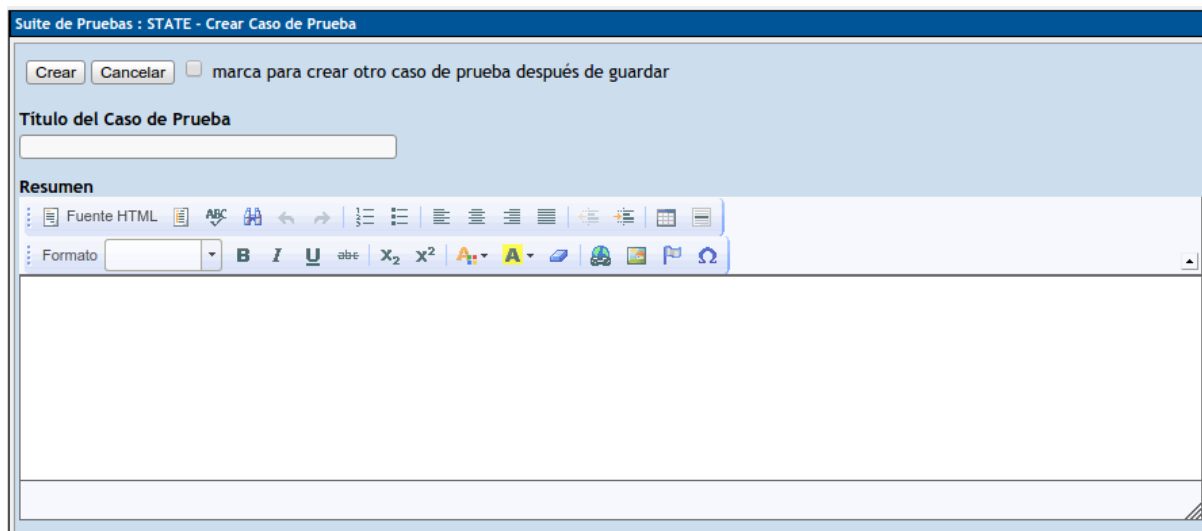
En las dos capturas realizadas del Jira se pueden observar elementos en común:

- El creador de la tarea o reporter
- Quien tiene asignada la tarea (Assignee), es decir la persona que realizará la tarea.
- La descripción de la tarea o del bug, donde se especificará la información necesaria.
- El tipo de tarea (tarea o bug)
- La prioridad de la tarea indica que importancia tiene la tarea y cuando debe resolverse.
- El estado en el que se encuentra, por ejemplo el estado Testing es que ya se ha realizado y se puede probar para comprobar su correcto funcionamiento.

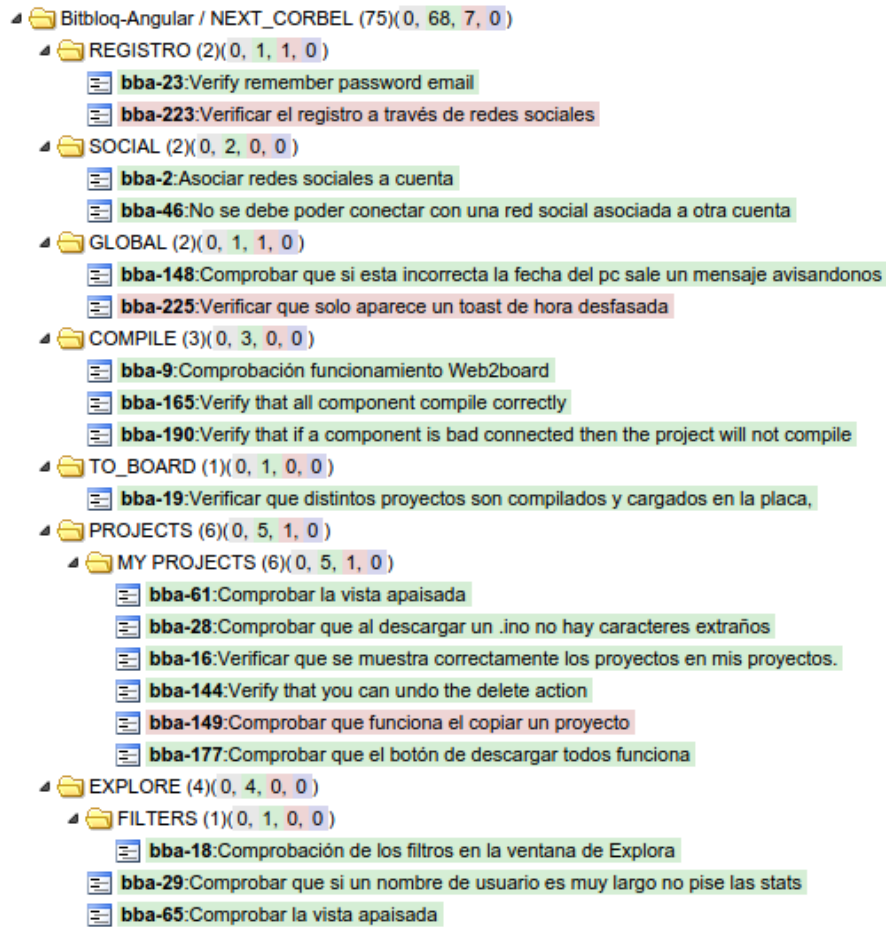
Seguimiento

Por otro lado, TestLink es una herramienta para el almacenamiento y gestión de test. También permite el almacenamiento y gestión de builds (un historial de los test que han pasado correctamente y de los que no han pasado para certificar el funcionamiento de la aplicación).

Para crear cualquier elemento en la herramienta, es necesario especificar un título o nombre que identifique al elemento, y una breve descripción que explique lo que hace el elemento, por ejemplo, en el caso de un test se deberá explicar los pasos necesarios para llevar a cabo la prueba. En la imagen inferior se puede ver la captura de Testlink para crear un test.



Otro parte útil de la herramienta es la gestión de builds, en la que puedes ver que test han pasado y cuáles no. Cada vez que se certifique el software se almacenará en una build diferente, quedando reflejado en cada build, qué funcionó correctamente y qué partes fallaron. En la siguiente captura se observan todos los test que han pasado en verde y los que no han pasado en rojo.



La calidad de un producto es más que un testeo de las condiciones de aceptación de historias de usuario, se debe de ser responsable de procesos y automatizaciones que den no solo valor añadido al producto si no una seguridad y escalabilidad.

Responsabilidades Estas serían las responsabilidades en los procesos de calidad:

1. **Certificación historias de usuario y bugs:** se deben certificar las condiciones de satisfacción de las user stories (US) y los bugs.
2. **Certificación por UX:** se deben identificar aquellas US que por su magnitud o impacto en la aplicación sean susceptibles de necesitar una verificación

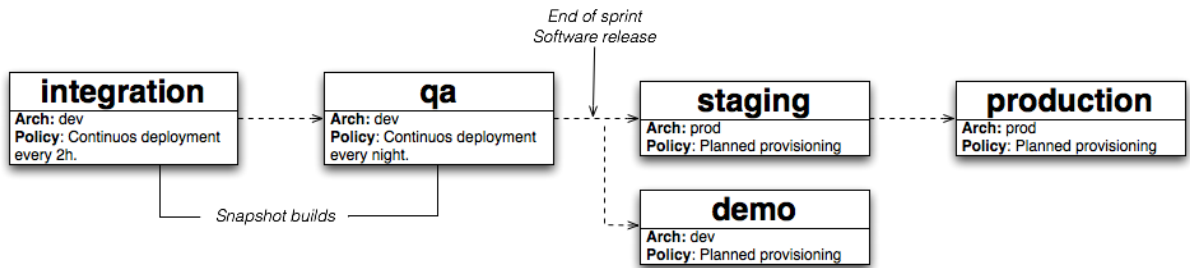
por UX. Esta verificación debe ser explícita y dejando constancia en la ficha de la US del sistema de gestión a utilizar.

3. **Estrategia de Unit Testing:** no todo es susceptible de ser testeado a través de Unit Tests por tanto se debe definir las partes que sí deben hacerse y un objetivo de coverage aceptable.
4. **Liderar Framework automatización:** se debe automatizar las pruebas, configurar Jenkins y controlar que no se certifica una US sin esta validación; pero esto no significa que el responsable deba automatizarlo todo por sí mismo, será su función delegar esto siempre acorde con la capacidad y plazos del equipo.
5. **Monkey Testing:** el departamento tiene a un equipo de testing del cual para el proyecto corresponden un número de horas semanales.

Será labor del responsable de calidad el proporcionar a este equipo las tareas a realizar y coordinación de los tiempo con el responsable del dpto.

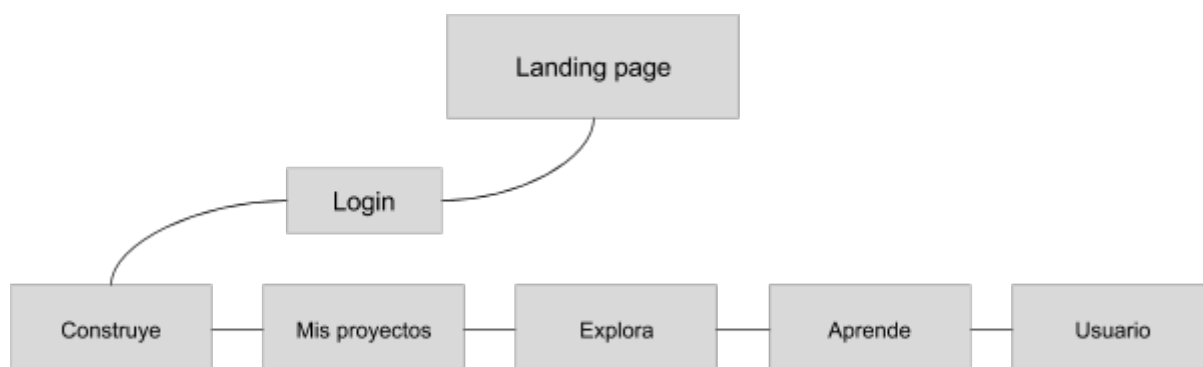
1. **Stress Tests:** se deben realizar pruebas de stress utilizando diferentes herramientas que nos garanticen una seguridad en cuanto a fiabilidad de la aplicación. Se deberán hacer registros e históricos de los datos para detectar cualquier variación que pueda suponer un potencial problema o defecto oculto.

Entornos de prueba Para realizar un proceso completo de certificación, los paquetes de software que van a desplegarse en producción, deben pasar una por una serie de entornos y en cada uno de los cuales deben cumplir una serie de requisitos y pasar un conjunto de pruebas específicas. A continuación, se detallan en el siguiente gráfico los entornos ideales que deberían existir en el proceso de certificación.



2 Requisitos específicos de la plataforma

La herramienta tiene un núcleo principal, el espacio donde se crea la programación y desde donde se tiene acceso a todas las partes que la componen.



Landing Page

Página descripción de la herramienta online con información detallada de características, necesidades y especificaciones técnicas para el correcto funcionamiento.

Login

Se puede acceder a la herramienta a través de un login disponible con email propio o mediante redes sociales como Facebook y Google+.

Construye (núcleo)

Núcleo de la herramienta, en esta parte es donde se desarrolla todo el proceso de programación con bloques visuales de código y donde se tiene acceso a todas las acciones disponibles de la herramienta.

Explora

Comunidad de usuarios y proyectos para poder compartir información entre ellos e incluso poder ver y modificar proyectos de otros usuarios para aportar en ellos y avanzar en el conocimiento y el desarrollo de la programación.

Aprende

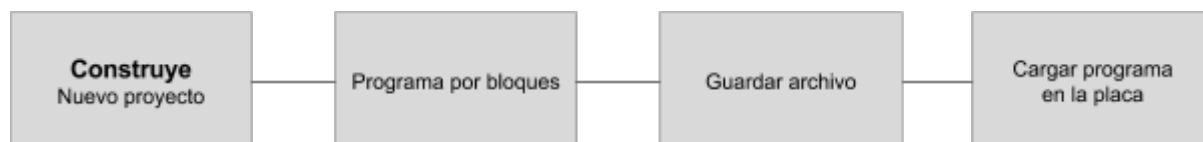
Esta sección lleva una parte del ecosistema de educación donde se enseña con contenido pedagógico todo lo referente a la herramienta, a programación y a la electrónica.

Usuario

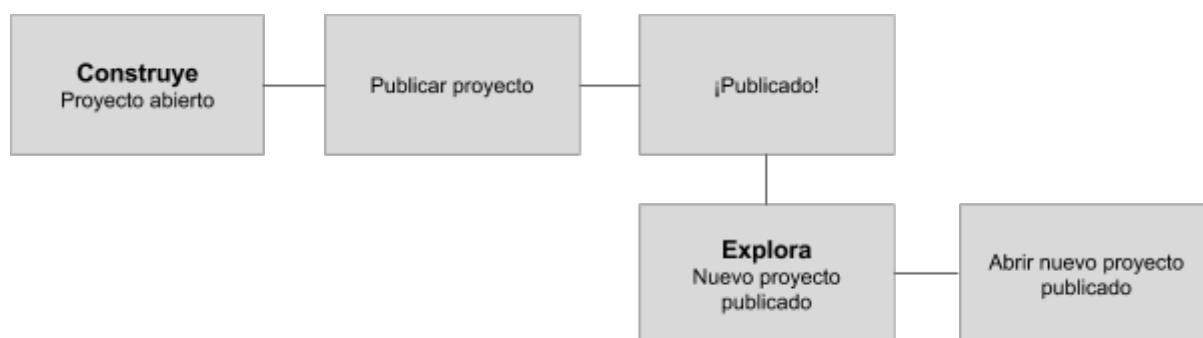
Configuración de la cuenta del usuario en la herramienta.

2.1 Flujos de navegación principales

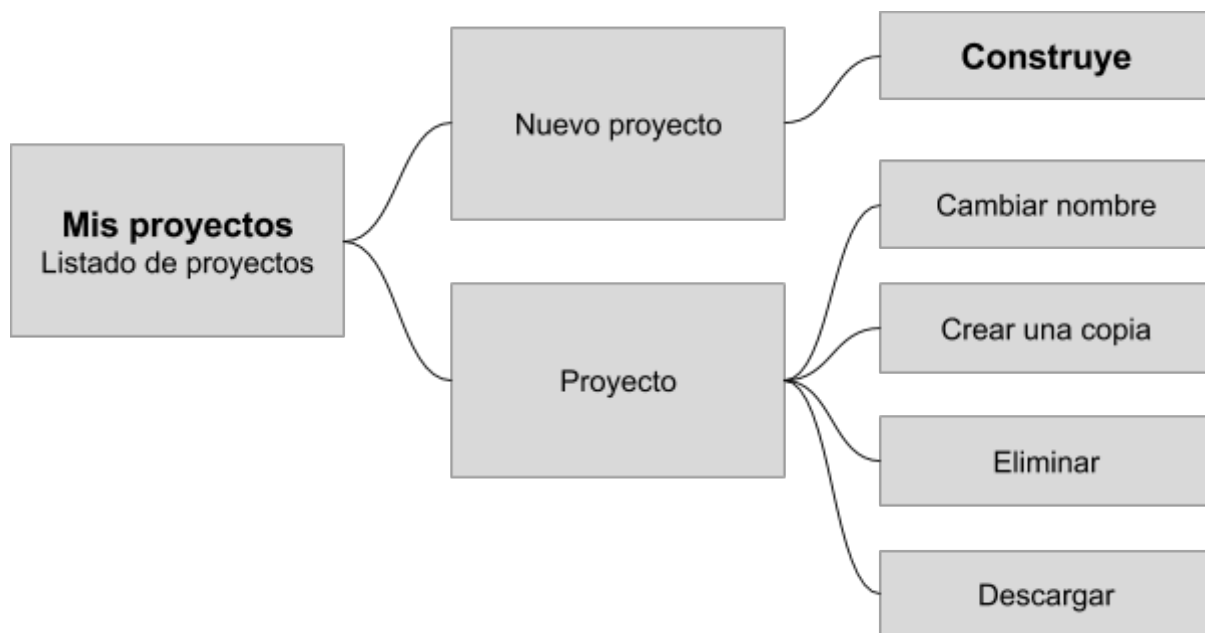
- Flujo principal de la herramienta, creación de un programa y carga en la placa



- Publicar proyecto en la comunidad



- Gestión de proyectos



Como requisitos específicos de la plataforma, se necesitan fijar una serie de funcionalidades básicas y concretas, que se detallan a continuación.

2.2 Espacio de trabajo de usuario (Construye)

El espacio de trabajo de la plataforma es el principal componente sobre el que se desarrolla toda la actividad del usuario.

Las principales funciones y elementos de este componente son detallados a continuación:

2.2.1 Espacio de trabajo con bloques

Se puede considerar como la parte principal de la plataforma, las principales acciones del usuario giran en torno a este espacio de trabajo, las acciones que el usuario puede realizar bajo este módulo son las siguientes:

- Se podrán juntar y separar bloques, comentarlos, seleccionar pines... , y en general, crear estructuras complejas anidando y concatenando bloques.

- Además se deben de poder realizar acciones de edición sobre los elementos de la pantalla.
 - Zoom -> Agranda o disminuye los bloques del espacio de trabajo.
 - Copiar -> Inhabilitado.
 - Hacer/Deshacer -> Deshace la última acción realizada sobre el espacio de trabajo.
- De manera local debe existir la posibilidad de interactuar con documentos estáticos del propio ordenador del usuario.
 - Cargar proyecto -> Te abre una ventana para que elijas la ubicación desde la cual vas a importar el proyecto.
 - Descargar -> Descarga un archivo .xml con el código. La descripción y los tags irán dentro del xml.
 - Exportar código arduino -> Descarga un archivo .ino con el código.

2.2.2 Zona de elección de bloques

Como componente asociado al espacio de trabajo, este elemento se debe comportar a modo de ‘caja de herramientas’ y facilitar al usuario un mecanismo de selección de cualquier tipo de bloques que haya agregado a la plataforma con el fin de agregarlo a su proyecto.

Los bloques deberán categorizarse visualmente para permitir al usuario una rápida selección de elementos para añadir al espacio de trabajo.

2.2.3 Código autogenerado

En todo momento, el usuario debe ser capaz de visualizar el código nativo o de bajo nivel que se va autogenerando al construir estructuras con los bloques. Los requisitos o funcionalidades que debe tener esta vista de usuario son:

- El código no se puede modificar. Es una vista de solo lectura.

- Zoom -> Se podrá hacer zoom sobre la letra del código, con valores contenidos entre un máximo y un mínimo.
- Copiar código -> En caso de que el usuario quiera copiar este código, se le facilita una acción simple de copiado al portapapeles.

2.2.4 Información del proyecto

Todos los campos que categorizan e identifican a un proyecto, pueden ser editados en esta parte de la plataforma. Los campos que un usuario debe poder modificar de un proyecto, son listados a continuación:

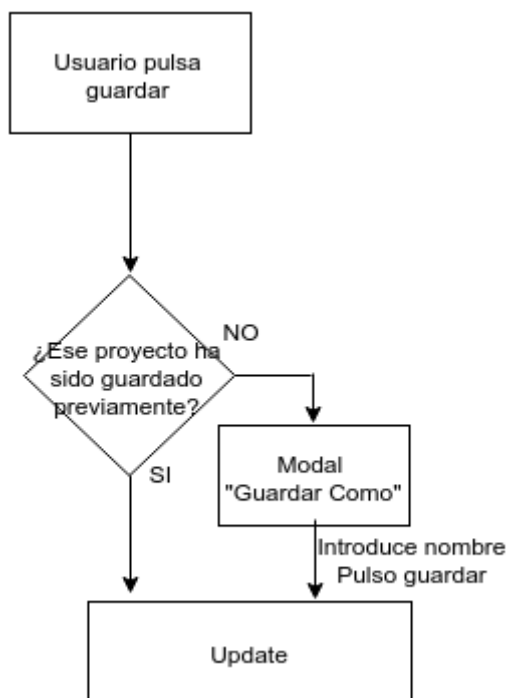
- Nombre del proyecto.
- Descripción básica del proyecto.
- Imagen asociada al proyecto.
- Url de youtube asociada a un posible video de demostración del funcionamiento del prototipo hardware.
- Etiquetado de proyecto -> Se pueden añadir o quitar, excepto las asignadas a bloques, que vienen definidas automáticamente por cada tipo de bloque que exista en el espacio de trabajo. Nombre, descripción, imagen, tags. Se guardan únicamente al pulsar en "Actualizar datos".

2.3 Gestión de proyectos de usuario

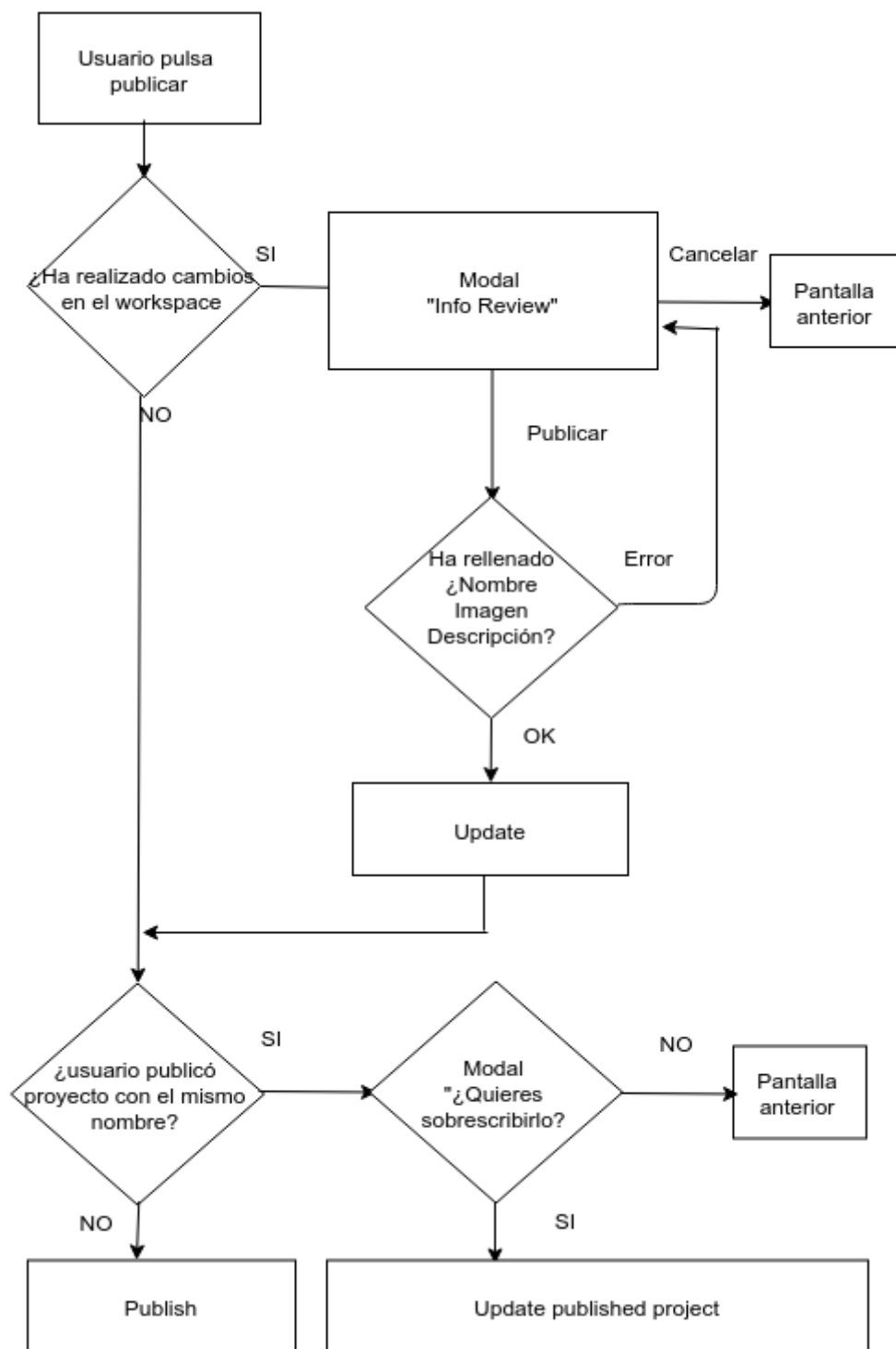
A través del entorno de programación se ofrece al usuario una solución completa de gestión de su perfil y sus proyectos, así como, un facilitar un ecosistema para compartir de manera fácil proyectos entre usuarios.

Las acciones que un usuario puede gestionar, tratando el concepto de proyecto como unidad atómica de información en la plataforma, se exponen a continuación:

- Crear y guardar nuevos proyectos en la Nube, estos proyectos pueden ser accedidos desde cualquier otro ordenador en cualquier momento. Un flujo genérico de guardado de proyectos, sería el siguiente:



- Abrir y crear copias de otros proyectos.
- Publicar un proyecto, tanto para ser compartidos con usuarios concretos como proyectos públicos abiertos a la comunidad. La acción de publicar un proyecto hará que se cree una instantánea pública del proyecto guardado, el cual no será editable, pero sí podrá ser borrado por el mismo usuario. En el siguiente esquema, se define el proceso genérico de publicación de proyecto.



Como partes de la plataforma necesarias, se deben reseñar mecanismos para

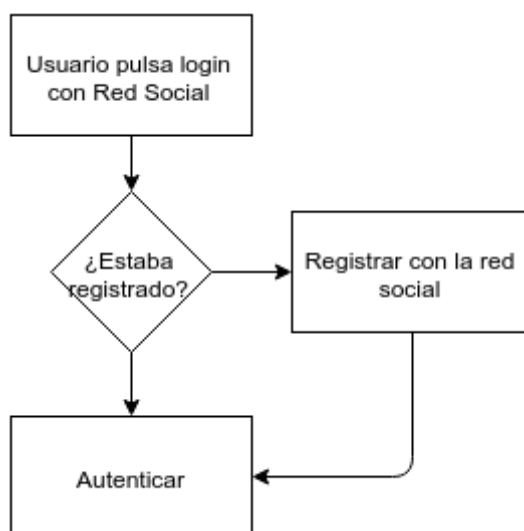
garantizar acceso, registro y gestión del ‘espacio de usuario’, para ello se enumeran en los siguientes puntos, los requisitos que debe garantizar la plataforma:

- Sistema de login y registro en la plataforma
- Proyectos privados del usuario (Mis proyectos)
- Espacio público y compartido de proyectos (Explora)

2.3.1 Sistema de login y registro en la plataforma

Cuando un usuario quiere registrarse y acceder al sistema puede hacerlo de dos maneras diferentes:

- Forma manual: el usuario debe escribir manualmente todos los datos necesarios como son su nombre de usuario, su correo electrónico, su contraseña, fecha de cumpleaños y aceptar los términos y condiciones del sistema.
- Con redes sociales: el comportamiento de esta forma de acceder al sistema sigue el siguiente flujo:



Si un usuario registrado se le ha olvidado la contraseña para acceder a Bitbloq puede recuperarla siguiendo el siguiente flujo:



Desde el punto de vista del ámbito social de un proyecto, se establecen dos tipos de gestión de los proyectos desarrollados con la plataforma:

2.3.2 Proyectos privados del usuario (Mis proyectos)

Hacen referencia a proyectos que un usuario tiene asociados como propia creación y que, a priori, son privados y por tanto otro usuario que acceda a la plataforma no es capaz de visualizarlo. Desde este espacio un usuario puede tener completo control sobre el ciclo de vida de sus proyectos.

Para la primera fase esta sección está dividida, en una gestión de carpetas, que darán acceso al contenido de las mismas. Estas carpetas son:

- Mis proyectos: (Aquí aparecerán tanto los que he guardado el usuario, como los que ha pulsado "Añadir a mis proyectos" sobre proyectos públicos).
- Proyectos publicados (por el usuario).

- Recientes: Aquí aparecen los últimos 5 proyectos modificados (misma información que en el wizard, sin el "último proyecto").

Tabla de acciones cuando está seleccionado un proyecto es la siguiente:

	Cambiar nombre	Crear copia	Eliminar	Abrir	Exportar código arduino	Descargar
Mis proyectos	X	X	X	X	X	X
Proyectos publicados		X	X	X	X	X
Recientes				X	X	X

La opción de "Nuevo proyecto" estará habilitada sólo cuando no hay ningún proyecto seleccionado y si no está en la carpeta "Proyectos publicados". Además, para selección múltiple sólo debe estar habilitada la opción de Eliminar.

2.3.3 Espacio público y compartido de proyectos (Explora)

Este espacio recoge los proyectos que han sido publicados por el usuario para que el resto de usuarios puedan acceder a su contenido. Deben existir dos ámbitos a la hora de compartir proyectos con otros usuarios:

- Proyectos públicos
- Proyectos compartidos

La plataforma debe proveer de los mecanismos básicos de búsqueda y filtrado de proyectos publicados por los usuarios en la plataforma. Las operaciones sobre las colecciones de proyectos públicos se enumeran a continuación, además, todas las operaciones de filtrado son mutuamente excluyentes:

- Ordenar por:

- Recientes.
- Más vistos
- Más añadidos.
- Buscar proyectos públicos:
 - Nombre de proyecto.
 - Descripción de proyecto
- Filtrado por tag:
 - Aparecerá un listado fijo de etiquetas y el usuario podrá seleccionarlo. Entre ellas el usuario podrá seleccionar ver tan solo los proyectos propios de BQ

Un proyecto publicado deberá tener una URL capaz de redirigir a la plataforma y mostrar en el espacio de trabajo el proyecto referenciado.



Este espacio debe cumplir las siguientes condiciones:

- Los proyectos podrán guardarse con el mismo nombre.
- En el modal de "guardar como": el botón de "guardar" estará deshabilitado si el espacio del nombre está vacío.
- Tras guardar, se debe actualizar un espacio de notificaciones de guardado que se encuentra en la interfaz de usuario.
- Cuando un usuario está viendo los proyectos públicos debe ser capaz de acceder al detalle de cada uno de ellos.

- Si el usuario pulsa sobre un proyecto de una lista de proyectos, se debe visualizar una ficha con la información asociada al proyecto. Solo una ficha de proyecto puede estar visible a la vez. El usuario podrá realizar una serie de acciones sobre ese proyecto en concreto.
- Cuando un usuario accede a una ficha de proyecto público, debe tener la opción de añadir el mismo a su espacio de proyectos.
- La opción de "Nuevo proyecto" estará habilitada sólo cuando no hay ningún proyecto seleccionado y si no está en la carpeta "Proyectos publicados".

Casos de uso 1. Buscar proyectos

1. El usuario hace foco en campo del buscador y empieza a introducir caracteres.
2. Cuando el usuario introduce múltiplos de 3 caracteres se realiza una búsqueda automática o pulsa en el icono de "lupa" o el "enter" del teclado, se lanza una búsqueda.
3. La búsqueda no será sensible a caracteres especiales y busca todos los proyectos cuyo nombre tiene una coincidencia con el string introducido con el usuario.
4. En caso de haber más resultados de los que caben en las dimensiones de la pantalla habrá scroll infinito.

Casos de uso 3. Filtrado por tag

1. Usuario pulsa sobre un tag.
2. El listado se muestra filtra por ese tag.
3. Usuario pulsa un segundo tag.
4. El listado se filtra por ambos tags.
5. Usuario pulsa el último tag que está actualmente filtrado y marcado se deja de aplicar el filtrado(se aplica búsqueda u ordenación si estuvieran aplicadas).
6. Si pulsa en "todos los tag" se quedan marcados todos.

Casos de uso 4. Ver proyecto público

1. El usuario pulsa sobre un proyecto de la lista y entre la fila en la que se encuentra el proyecto que pulsa y la posterior se visualiza la ficha.
2. Si con la ficha de un proyecto desplegada pulso en otro proyecto, se cierra el anterior y se abre el que haya pulsado.
3. El usuario realiza una de las siguientes acciones:

- **Abrir proyecto:** Cuando un usuario pulsa "Abrir proyecto" se crea un proyecto copiando la descripción que fuera en el proyecto público, pero no se guarda.

Es como un "Crear proyecto" pero rellenando algunos campos de descripción en caso de haberlos.

Ya en el espacio de trabajo, si el usuario pulsa guardar, aparecerá "guardar como" a todos los efectos es como un proyecto recién creado que no ha sido guardado nunca.

Tendrá el asterisco rojo correspondiente y no habrá una referencia a cuando fue guardado. Habrá un contador de "Abrir proyecto" que se verá reflejado cada vez pulsa un usuario sobre la opción en ese proyecto.

- **Añadir a mis proyectos:** Cuando un usuario pulsa añadir a mis proyectos se guarda el proyecto público como si fuera el usuario. Pero sin abrirlo, ni mostrarlo en el espacio de trabajo.

Este proyecto al ser del usuario le aparecerá en "Mis proyectos" y en "Abrir".

Tras añadir el proyecto, cambiará el literal : "Proyecto añadido", el cual no tendrá acción.

Cuando el usuario va a "Mis proyectos" o "Abrir" tendrá una visualización para que se sepa que se agregó.

Habrá un contador de "Añadir a mis proyectos" que se verá reflejado cada vez pulsa un usuario sobre la opción en ese proyecto.

- **Descargar:** Esta opción descarga el proyecto formato .xml. Mismo comportamiento que en "Mis proyectos" y en el espacio de trabajo cuando se pulsa en descargar.
- **Me gusta:** El me gusta es un contador que sobre un proyecto público se va acumulando.

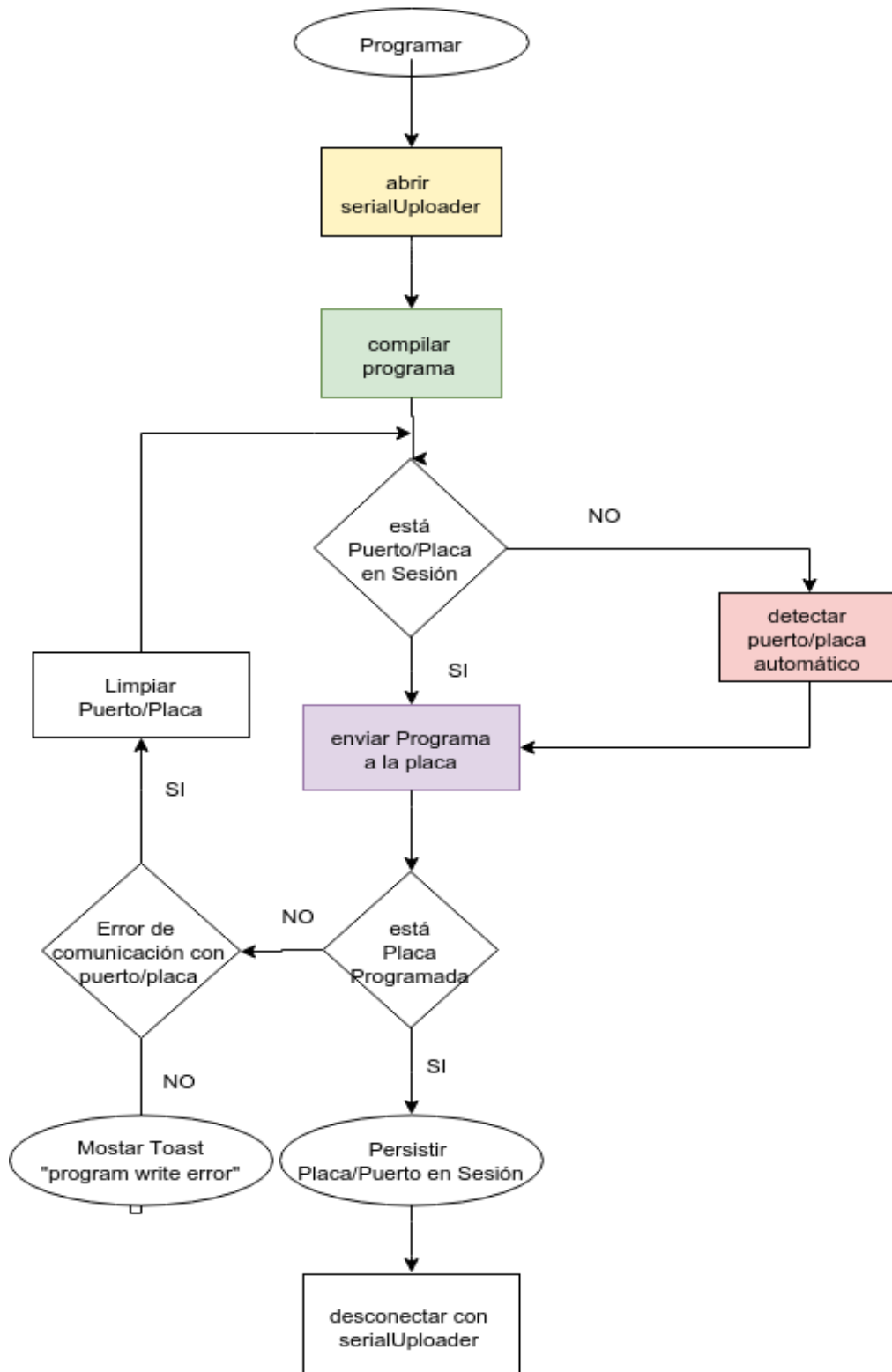
Cuando el usuario pulsa sobre me gusta el botón cambiará y permitirá la opción de quitar el "me gusta".

Habrà un contador de "Me gusta" que se verá reflejado por cada me gusta que hace un usuario sobre un proyecto.

2.4 Software de programación de hardware (Serial uploader)

Las principales etapas dentro del proceso de programación, se enumeran a continuación y se reflejan gráficamente en el grafo siguiente:

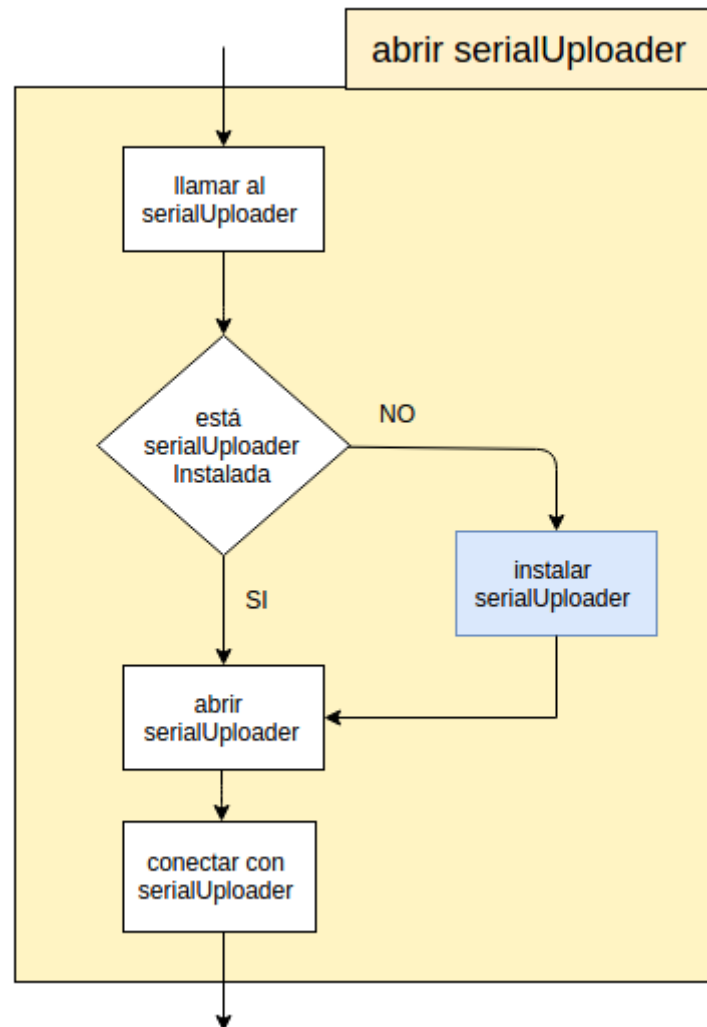
1. Abrir puerto de comunicación entre el entorno de programación y el módulo de carga.
2. Enviar orden de compilación de código autogenerado.
3. Proceso de detección de placa microcontroladora.
4. Carga de binarios compilados en la placa detectada.
5. Desconectar puerto de comunicación



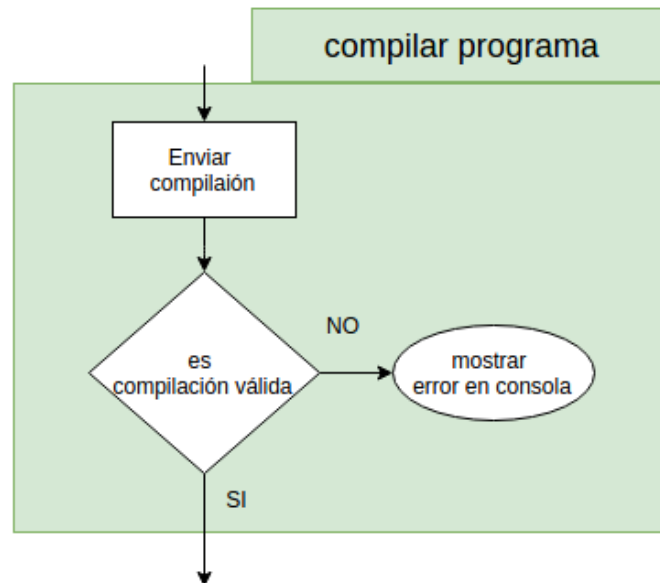
37

A continuación, se pasan a detallar los procesos más complejos del diagrama anterior.

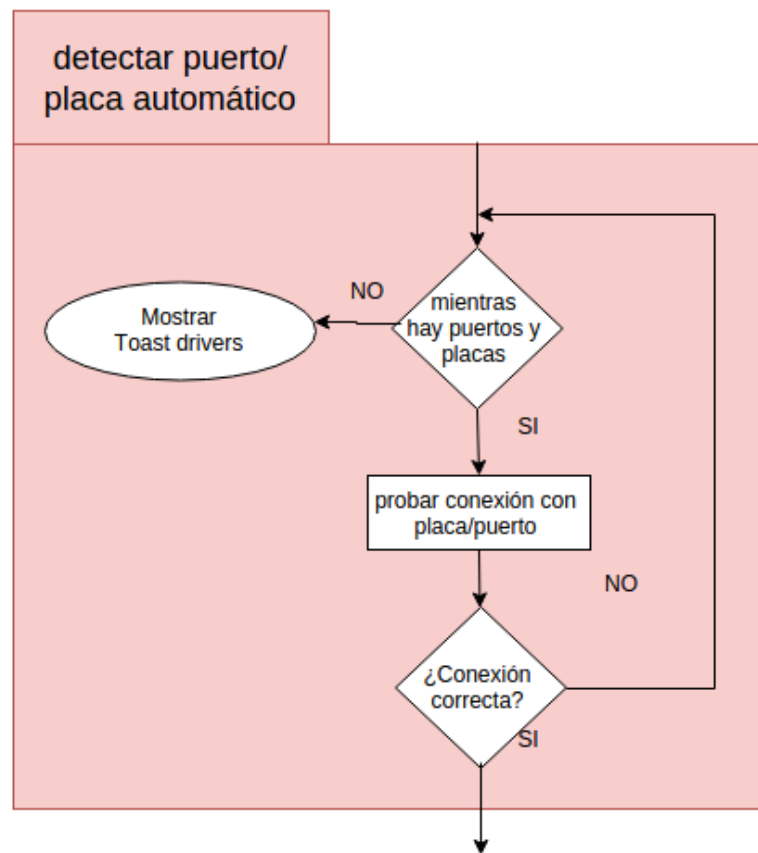
- Abrir serialUploader: En el momento que un usuario quiere programar o compilar y programa debe abrirse el serialUploader.



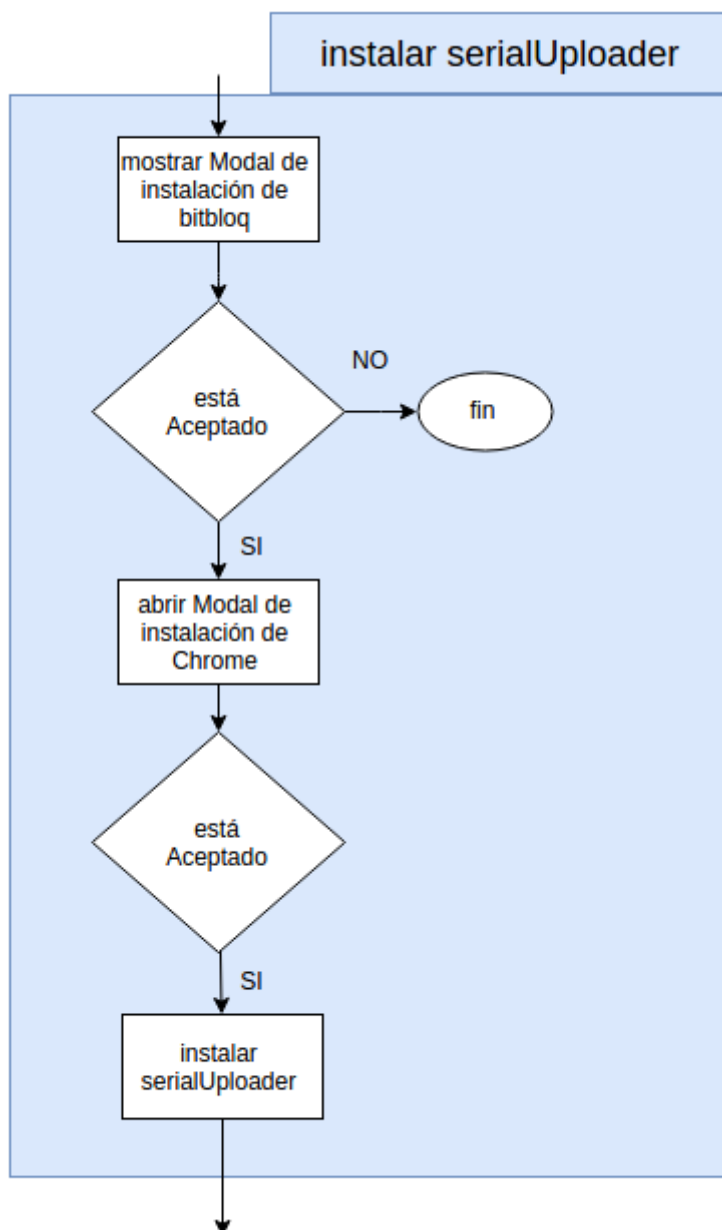
- Compilar programa: Una vez que se ha abierto el serialUploader se realiza la acción de compilar o de programar. Dichas acciones tienen el mismo flujo, por lo tanto, tan solo mostraremos el correspondiente a compilar programa.



- Detectar puerto o placa automático: Tras compilar un programa se debe detectar la placa a la que se debe subir el programa. Esto debe ser una acción transparente al usuario.



- Instalar serialUploader: Para poder realizar todo el proceso anterior el serialUploader debe estar instalado. El flujo que se debe seguir cuando un usuario no lo tiene instalado es el siguiente:



2.4.1 Casos de uso

- Cuando el usuario inicie un proceso de programación o compilado, la plataforma intentará detectar el software específico para que la operación sea transparente para el usuario. En el caso de que no sea detectado este módulo de

software, se debe garantizar al usuario un flujo de asistencia, dentro de la plataforma, con instrucciones precisas sobre como instalar este módulo de software concreto. Una vez, el usuario ha conseguido instalar el software, la operación se debe repetir, esta vez, se debe iniciar un proceso transparente al usuario de compilación/programación.

2.5 Gestión de perfil de usuario

El perfil del usuario es la parte de la web donde el usuario podrá configurar y modificar sus datos. El usuario también podrá asociar su cuenta a redes sociales como facebook y google plus.

Además, el usuario puede modificar sus datos de acceso a la plataforma de dos formas diferentes:

- En el caso de autenticarse con redes sociales podrá pulsar en: Establecer contraseña dentro del perfil de usuario. El funcionamiento será el mismo que recordar contraseña(enviará correo para resetearla).
- En el caso de autenticar con Corbel el usuario puede cambiar la contraseña escribiendo dos veces la nueva contraseña, sin mostrar la actual.

2.5.1 Perfil de usuario

El perfil de los usuario tendrá los siguientes datos.

- Avatar -> Se establecerá uno por defecto.
- Email -> Los alumnos no dispondrán de este dato.
- Nombre -> No obligatorio.
- Apellidos -> No obligatorio.
- Alias -> Por defecto es el username propios de Corbel
- Idioma

2.5.2 Permisos de usuario

Rol	Invitado	Autenticado
Crear nuevo proyecto	(Hay uno creado en blanco al entrar excepto si hubiera uno anteriormente en local)	X
Cargar proyecto local		X
Descargar proyecto		X
Abrir proyecto Cloud		X
Guardar proyecto Cloud		X
Exportar código arduino		X
Publicar un proyecto		X
Cambiar nombre		X
Crear copia		X
Añadir bloques		X
Ver código	X	X
Ver información del proyecto	X	X
Ver "Explorar"	X	X
Zoom/ hacer/ deshacer	X	X
Profile		X

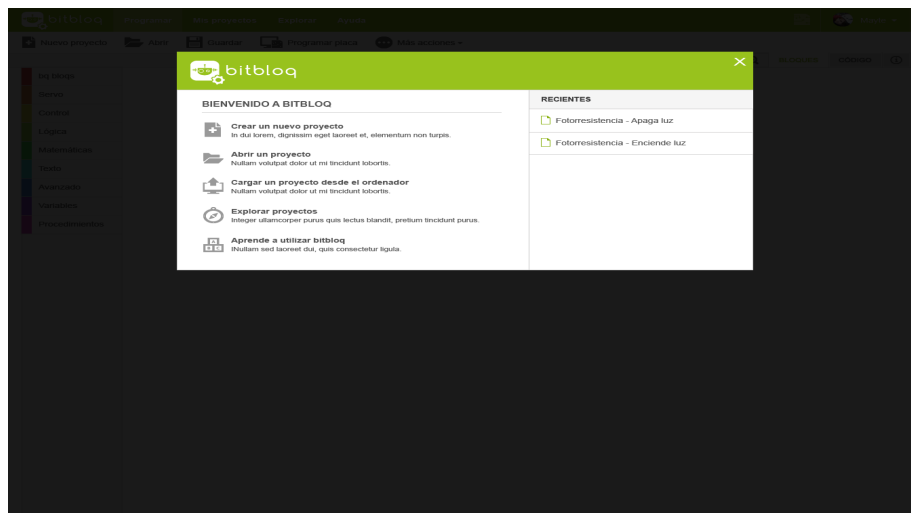
2.6 Otros aspectos de la plataforma

2.6.1 Wizard

El wizard va a proporcionar al usuario un acceso rápido a las opciones más comunes que se suelen realizar un usuario:

- Crear nuevo proyecto -> Acciona la funcionalidad de Nuevo proyecto.
- Abrir un proyecto -> Va a la pestaña de "Mis proyectos".

- Cargar proyecto desde el ordenador -> Acciona la funcionalidad de "Cargar proyecto".
- Explorar proyecto -> Va a la pestaña de explorar.
- Aprende a utilizar bitbloq -> Misma acción que pulsar en help.
- Recientes: Aparecen los 5 últimos proyectos que el usuario modificó después aparecerá un "Última sesión" seguido de la fecha y hora de esa sesión que se guardó en local(en caso de haberla).
- Cerrar wizard -> Cuando se pulsa en la "X" o fuera de la zona del wizard -> Tras realizar esta opción aparecerá el último proyecto hubiese uno.



2.6.2 Consola

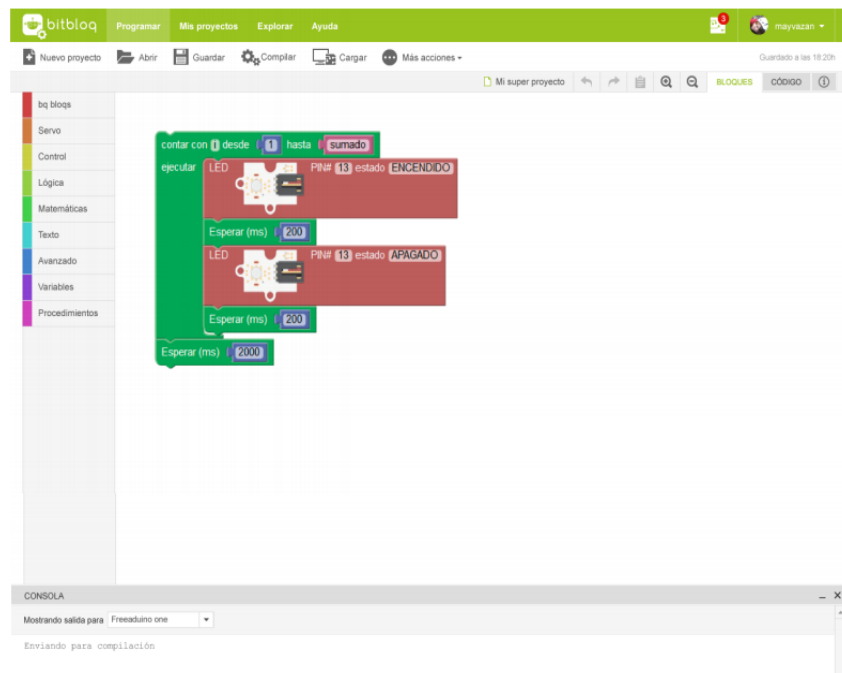
La plataforma dispondrá de un sistema de feedback visual con el módulo de programación de hardware. El monitor serie es una pieza fundamental para leer datos de las placas microcontroladoras y debe estar disponible de cara al usuario desde la interfaz gráfica. Las operaciones que se pueden realizar con este módulo de software, se enumeran a continuación:

- Operaciones básicas:

- Un campo de texto para escribir un mensaje
 - Un botón para enviar el mensaje.
 - Un espacio donde se vayan añadiendo mensajes que envía la placa y los que manda el usuario.
 - Puerto -> Puedo monitorizar una placa después de tenerla programada y puede variar el puerto.
 - Actualizar puertos -> Para poder listarlos.
 - Baud rate -> Para elegir una tasa de transmisión -> Valores fijos.
- Operaciones avanzadas:
 - Copiar log al portapapeles.
 - Limpiar log.
 - Desconectar -> Para dejar de monitorizarlo.
 - Habilitar/Deshabilitar Autoscroll

La consola estará ubicada en la parte inferior de la pantalla y recogerá toda las comunicaciones y errores que ocurren durante las acciones de la compilación y cargar en la placa.

- La consola estará en la parte inferior de la pantalla en su forma minimizada. Cuando el usuario pulsa sobre: Compilar o Ejecutar la consola se despliega.
- Cuando el usuario pulsa sobre: Compilar o Ejecutar la consola se despliega.



Caso de uso En la consola se muestran los mensajes correspondientes a los siguientes casos:

- Cuando el código ha sido cargado
- Cuando la subida de código a la placa a fallado
- Cuando el código ha sido compilado
- Cuando la compilación del código es fallida
- Cuando la compilación está en progreso
- Cuando la compilación está en la cola de tareas
- Cuando la placa no está conectada
- Cuando la placa está conectada

Visualmente la consola debe cumplir lo siguiente:

- El texto de los mensajes de error deben ir en color rojo.
- Salen los mensajes en la consola según van sucediendo eventos.
- No se puede hacer resize de la consola.

Camino funcional:

1. El usuario pulsa compilar o programar.
2. Expandir la consola. Esta no debería aparecer inicialmente abierta porque siempre está por defecto minimizada si estamos en "Construye"
3. Hacer limpieza de consola.
4. Se va pintando en la pantalla los mensajes, manteniendo el foco en el último mensaje. Usuario minimiza la consola, pulsando en el icono de minimizar
5. Usuario maximiza la consola, pulsando en el icono de maximizar. Usuario pulsa sobre un "Mis proyectos" o "Explora" y la consola desaparece. Usuario vuelve a "Construye" y la consola aparece minimizada (siempre que se vuelve aparece así).

2.6.3 Sistema de ayuda y reporte de incidencias

Mediante un sistema sencillo en la plataforma se debe poder recoger feedback por parte del usuario, ya sea en forma de comentarios acerca de la plataforma, como reportes de fallos e incidencias.




Un apunte importante en este apartado es dotar a la plataforma de enlaces a otras plataformas comunitarias, así como a su contenido, relacionados con el mundo ‘maker’ y de desarrollo de productos de robótica educativa. Uno de los fines últimos de la plataforma debe servir de nexo con las comunidades de desarrollo de este tipo de tecnologías.

2.6.4 Visión y compatibilidades 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 la 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.

A continuación se usará la siguiente nomenclatura:

Símbolo	Compatibilidad
	100%
	Parcial(no programa)
	Sólo muestra la landing

Sólo se certifica con la última versión de cada navegador.

Navegador	Soportado
Chrome	
Chromium	
Firefox	
IE	
Resto	

Siempre se debe certificar con las últimas versiones. Lo cual no significa que no sea compatible con otros SO, sin embargo se garantiza al 100% su compatibilidad.

SO	Compatible
Windows 7	✓
MAC Yosemite	✓
Linux (Ubuntu)	✓

Se ofrecerá un modal que saldrá al iniciar sesión y al entrar como usuario anónimo cuando el SO del cual se está accediendo a bitbloq, no esté en la lista de SO compatibles. Dicha listado de SO será dinámico y en caso de añadir/quitar un SO a la lista deberá actualizarse automáticamente.