Práctica: Skills

(Front-end)

Juanan Pereira, Xabi Garmendia

Sistemas Web. 2024-2025

Índice

1 Introducción **1**
1.1 Preparar el entorno 1
1.2 Autentificación 1
1.3 Ejercicios 1
1.4 Logra *Skills* (competencias) 3
1.5 Ejercicios 3

2 Implementación de la sección Front-end **4**
2.1 Pintando Hexagonos 4
2.2 Ejercicios 4

3 Medallas y Puntos **6**
3.1 Ejercicios 6

4 Competencias y Tareas **7**

5 Evidencias **9**
5.1 Ejercicios 9

6 Confirmación de competencia **10**

7 ¡Lo has logrado! **13**

1 Introducción

En la presente práctica, denominada **Skills**, se implementará una plataforma de aprendizaje gamificado que fomente el aprendizaje práctico y colaborativo de un tema en concreto. La aplicación web a desarrollar utiliza un sistema *skills* representado mediante diamantes, con diferentes niveles de dificultad y puntuaciones. Los usuarios complementan las tareas de cada *skill* para conseguir las capacidades y la puntuación, desbloqueando medallas o *badges*. Cada vez que se obtiene un *skill* (competencia) nuevo se va ganando puntos. Cada *badge* (medalla) se adquiere por una cantidad determinada de puntos. El objetivo: conseguir la última medalla, Jedi-Zalduna.

Tendremos la opción de visualizar el progreso de cada alumno en el sistema *Skills*, realizar la evaluación peer-to-peer[1] así como observar una una tabla de liderazgos.

Además, el administrador tendrá su propio panel para; crear nuevos *skills*, editar los existentes o para gestionar los usuarios y contrasenas.

1.1 Preparar el entorno

La preparación del entorno para el desarrollo de la aplicación *Skills* es fundamental. Lo primero que hay que hacer es instalar y configurar herramientas de desarrollo como NodeJS y Express framework. A continuación crearemos la estructura básica del proyecto utilizando Express y realizaremos los cambios necesarios en la organización de los archivos y carpetas.

Es importante definir bien el estado inicial del proyecto: personalizar la página de inicio que se verá en el navegador e importar correctamente los archivos JavaScript. Además, utilizaremos el control de versiones (a través de GitHub) para seguir la evolución del proyecto y facilitar la colaboración entre los miembros del grupo.

Se recomienda realizar todo lo relacionado con esta sección de manera precisa y manteniendo bien la atención, con el fin de crear unos cimientos robustos para el proyecto.

> STOP
>
> En esta práctica recibirás las instrucciones que debes seguir para programar la aplicación *Skills*. Deberás programar y probar las funciones que se solicitan en cada sección. Tendrás que subir una versión de tu código a un proyecto privado de GitHub (repo). Tantos como se desean.

1.2 Autentificación

Para poder entrar en el sistema *Skills*, el usuario debe autentificarse. Así, se deben programar las funcionalidades de login y logout. En medida de lo posible mediante el protocolo OAuth y haciendo uso de una cuenta de Google, Facebook o GitHub (sin necesidad de realizar registro). De momento no lo vamos a implementar.

1.3 Ejercicios

- Si no lo has hecho ya, instala en tu ordenador NodeJS (última versión LTS) (https://nodejs.org/) y express (`npm install -g express`).
- Trabaja los siguientes comandos desde una terminal:

[1] Peer-to-Peer review es un proceso de evaluación donde los participantes se hacen sus propias evaluaciones

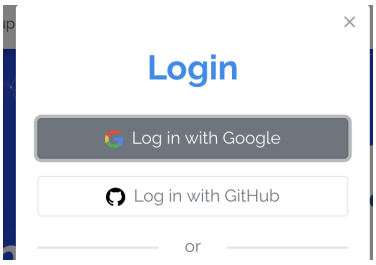


Figura 1: La autentificación OAuth se puede realizar, entre otros, con Google o GitHub

```
npm install cross-env -g
express --view=ejs skills
cd skills
npm install
cross-env "DEBUG=skills:*" npm start
```

- Abre/Carga la siguient URL en el navegador: http://localhost:3000 y confirma que se visualiza la página de defecto del framework Express (ver Figura 2).

Figura 2: Página por defecto de un proyecto Express

- Cambia el fichero views/index.ejs y en vez de visualizar "Express / Welcome to Express"haz que se visualice lo siguiente: "Skills / Sistemas Web (2024-2025)".
- Importa el siguiente script en el fichero views/index.ejs:

```
<script type="module" src="/js/main.js"></script>
```

- Renombra la carpeta public/javascripts/ a public/js y en vez de public/stylesheets indica public/css.
- En el fichero views/index.ejs, en vez de /stylesheets/style.css indica /css/styles.css.
- Crea el fichero, que estará vacío, public/js/main.js. En él indicarás el código principal de la aplicación mediante las instrucciones de las siguientes secciones.
- Sube tu código al GitHub correspondiente a tu grupo.

1.4 Logra *Skills* (competencias)

Primeramente, con el fin de lograr una aplicación exitosa, necesitamos preparar adecuadamente el contenido. En concreto, vamos a preparar una lista de competencias (*skills*) relacinada con la electrónica. Es decir, un array de JSON.

```
[
     {
       "id": 1,
       "text": "Light up an\n\n\nLED for the first \n\n\ntime",
       "icon": "/electronics/icons/icon1.svg",
       "description": "The quintessential first step in electronics! Learn the
           basics of c    ircuits and components by lighting up an LED."
     },
     {
       "id": 2,
      "text": "Learn to\n\n\nuse wire strippers",
      "icon": "/electronics/icons/icon2.svg",
      "description": "Master the art of preparing wires for connections by
          learning to us    e wire strippers effectively and safely."
    },
 ....
```

Lograremos el texto y las imágenes desde los siguientes links: https://tinkererway.dev/web_skill_trees/electronics_skill_tree

1.5 Ejercicios

- Crea una carpeta denominada scripts/
- Programa , en dicha carpeta, un script denominado scraper.js. En dicho script, extraeremos los datos necesarios desde las URL mencionadas anteriormente. Extraer los siguientes datos por competencia:
  - Número identificador
  - Texto de la competencia
  - Icono de la competencia nombre.svg
- Programa el script download_icons.js para obtener los iconos de los presentes links. En concreto:
  - Se descarga desde internet 67 iconos de electónica en el formato SVG.
  - Éstos se almacenan en una carpeta denominada public/electronics/icons.
  - Cada icono se guarda con la siguiente nomenclatura; icon1.svg, icon2.svg, etc.
  - Imprime el resultado de cada descarga indicando éxito o no.
  - Al terminar el proceso se debe visualizar el mensaje "Se han descargado todos los iconos".

2 Implementación de la sección Front-end

En esta primera aproximción vamos a introducir algunas simplificaciones:

- No vamos a tener en cuenta la autentificación del usuario. O, mejor dicho, vamos a simular una autentificación simple.
- Vamos a simular por completo la parte del código del servidor.
- En esta primera parte de la práctica NO vamos a usar la librería React[2] o parecidas. Vamos a realizar todo el código de la parte Front-end con JavaScript (*Vanilla JS*[3]).

React y la parte del servidor se tratarán en la siguiente aproximación de la práctica.

2.1 Pintando Hexagonos

Cada *Skill* se pinta en el interior de un hexagono, tal y como se observa en la imagen. Por otro lado, el hexagono, será un SVG, que se almacenará en una capa denominada svg-wrapper. Todos ellos estarán localizados en una capa denominada svg-container.

A continuación, el contenido del primer SVG:

```
<div class="svg-wrapper" data-id="1" data-custom="false">
            <svg width="100" height="100" viewBox="0 0 100 100">
                <polygon points="50,5 95,27.5 95,72.5 50,95 5,72.5 5,27.5"
                    class="hexagon"/>
                <text x="50%" y="20%" text-anchor="middle" fill="black" font-
                    size="10">
                    <tspan x="50%" dy="1.2em" font-weight="bold">Light up an</
                        tspan>
                    <tspan x="50%" dy="1.2em" font-weight="bold">LED for the
                        first </tspan>
                    <tspan x="50%" dy="1.2em" font-weight="bold">time</tspan>
                </text>
                <image x="35%" y="60%" width="30" height="30" href="../../
                    web_skill_trees_resources/svg/electronics_icons/icon1.svg"/
                    >
            </svg>
        </div>
```

Tal y como se observa, en el interior de cada hexagono colocaremos texto (el título de cada *skill*, por ejemplo:'Light up a LED for the first time'). Si el título es muy extenso, lo dividiremos en varias líneas, ajustando a la anchura del hexagono. A continuación visualizaremos la imagen del *skill* (en el ejemplo, `icon1.svg`).

2.2 Ejercicios

- Utiliza el array JSON creado en los ejercicios anteriores, para crear el panel dinámico de todos los *skills* (mediante JavaScript).
- Descarga la siguiente página de estilo CSS, con el fin de usarlo en local. Empieza por el siguiente código HTML para conseguir lo que se ve en la Figura4 .

[2]https://react.dev/

[3]Vanilla JS es un término humorístico utilizado para referirse a JavaScript puro, es decir, JavaScript sin el uso de bibliotecas o frameworks externos, como React, Angular o jQuery. http://vanilla-js.com/

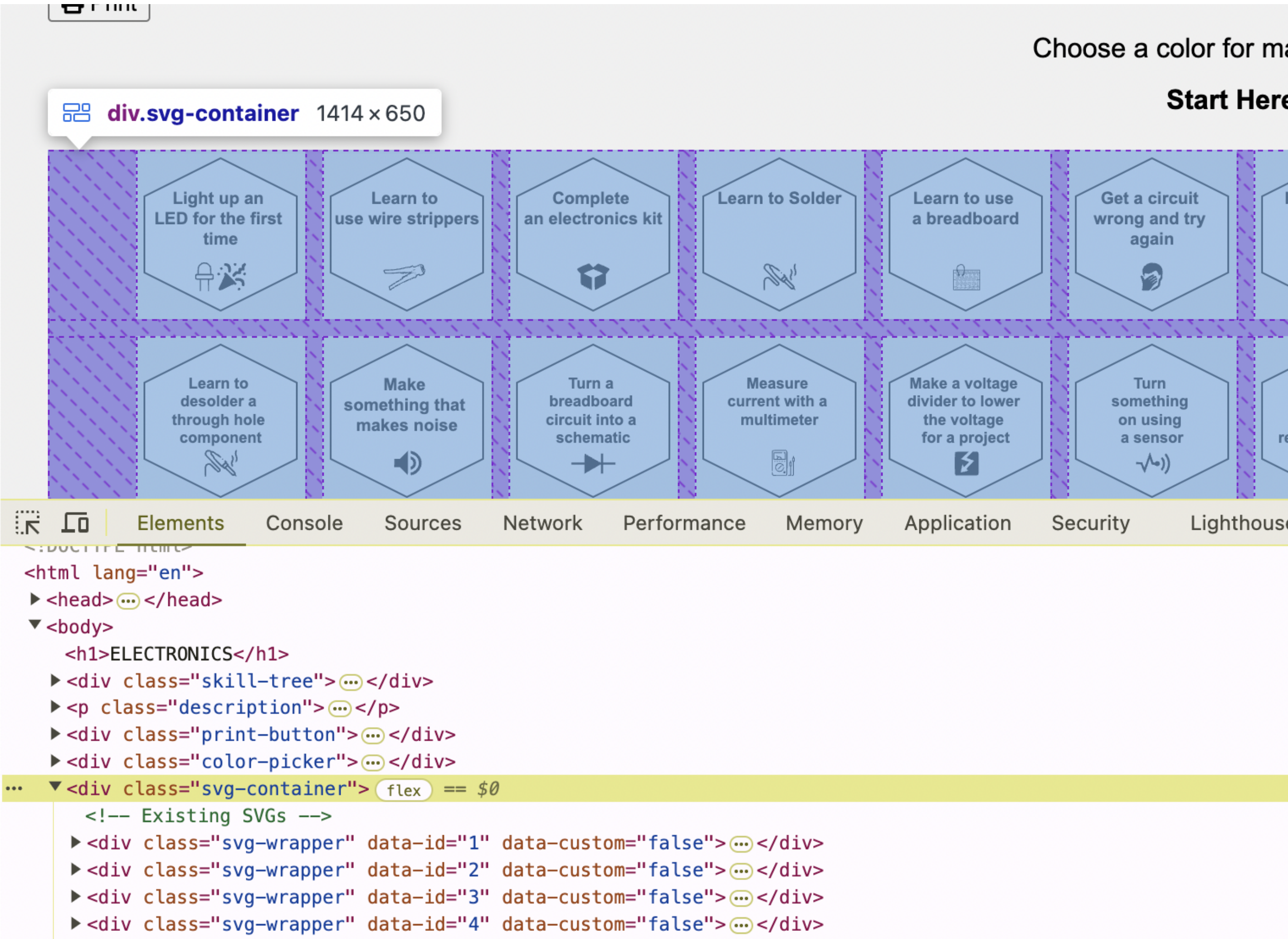


Figura 3: Cada hexagono será un svg

```
 <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Web-Skill Tree : Electronics</title>
    <link rel="stylesheet" href="styles.css">
    <script src="skills.js"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
       awesome/6.0.0-beta3/css/all.min.css">
</head>
<body>
    <h1>ELECTRONICS</h1>

    <div class="svg-container">
        <!-- Existing SVGs -->
    </div>
</body>
</html>
```

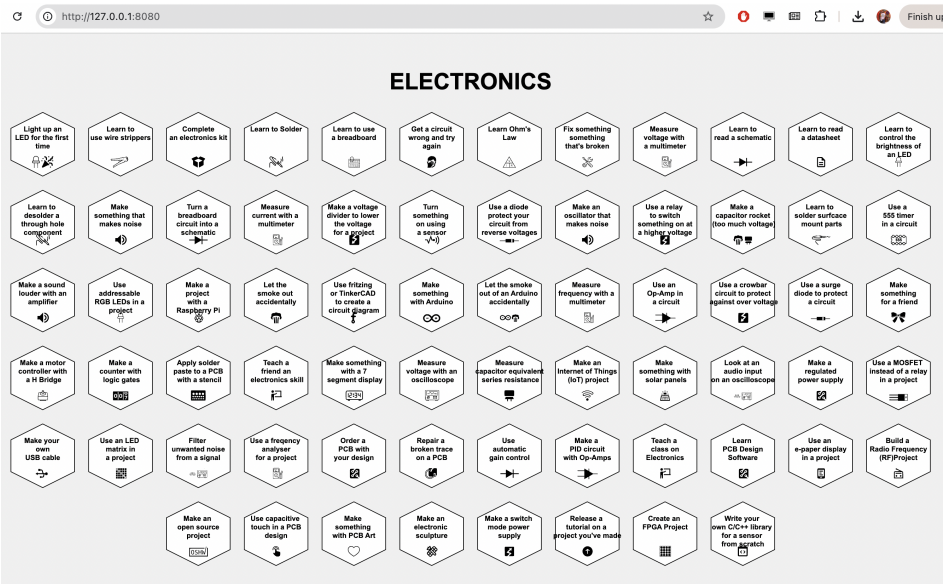


Figura 4: Panel de hexagono logrado dinámicamente

3 Medallas y Puntos

El estudiante logrará puntos (bitpuntos) a medida que va logrando competencias o *skills*. Con esos puntos logrará medallas.

En la presente aplicación se hace uso de dichas medallas. En concreto, medallas relacionadas con la saga StarWars.

https://github.com/Obijuan/digital-electronics-with-open-FPGAs-tutorial/wiki#listado-de-rangos

Por ejemplo, la medalla relacionada con "Observador"se obtiene con 0-9 puntos:

```
{"rango": "Observador", "bitpoints_min": 0,
    "bitpoints_max": 9, "png": "00-observador-min.png"},
```

Sin embargo, para conseguir "Aspirante a Cadete", se necesitan 10-19 puntos.

```
{"rango": "Aspirante a Cadete", "bitpoints_min": 10,
    "bitpoints_max": 19, "png": "01-Aspirante-cadete-min.png"},
```

3.1 Ejercicios

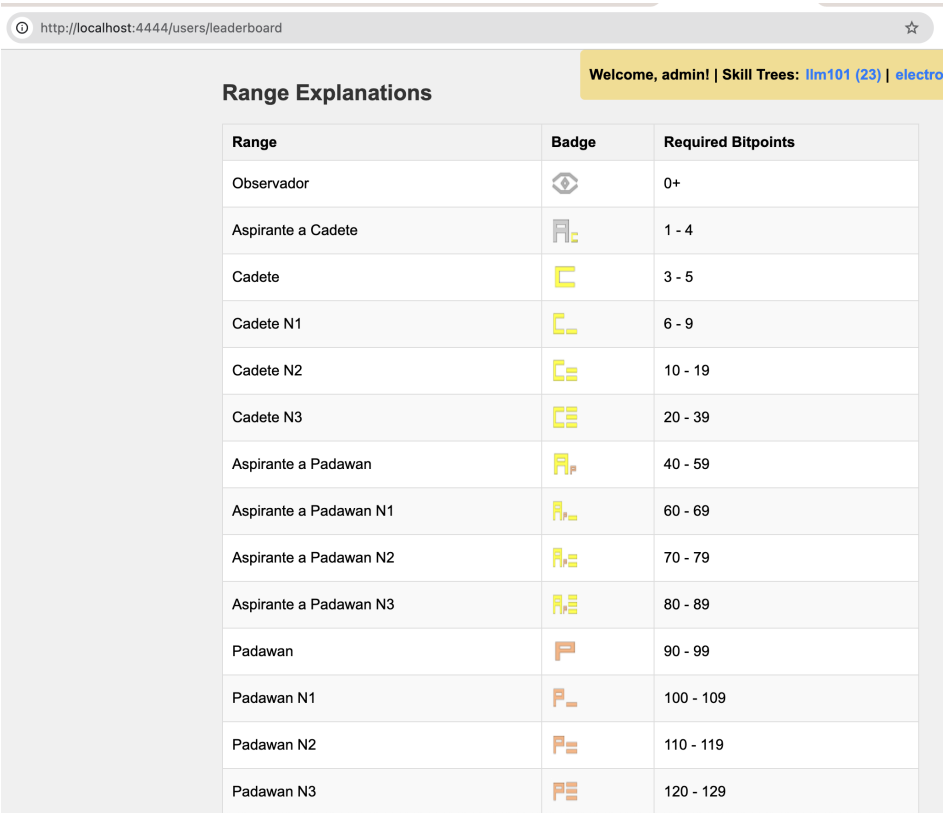
Programa un script NodeJS que revisa la web anterior (parsear) y extrae todas las medallas. En concreto, el script debe:

1. Revisar la web anterior y crear un array JSON, con la siguiente estructura:

```
let badges = [
 {"rango": "Observador", "bitpoints_min": 0,
        "bitpoints_max": 9, "png": "00-observador-min.png"},
    {"rango": "Aspirante a Cadete", "bitpoints_min": 10,
        "bitpoints_max": 19, "png": "01-Aspirante-cadete-min.png"},
 ...

]
```

NOTA: de momento establece los rangos de los bitpuntos en 10.



http://localhost:4444/users/leaderboard

Welcome, admin! | Skill Trees: llm101 (23) | electro

Range Explanations

Range	Badge	Required Bitpoints
Observador		0+
Aspirante a Cadete		1 - 4
Cadete		3 - 5
Cadete N1		6 - 9
Cadete N2		10 - 19
Cadete N3		20 - 39
Aspirante a Padawan		40 - 59
Aspirante a Padawan N1		60 - 69
Aspirante a Padawan N2		70 - 79
Aspirante a Padawan N3		80 - 89
Padawan		90 - 99
Padawan N1		100 - 109
Padawan N2		110 - 119
Padawan N3		120 - 129

Figura 5: Para lograr medallas se necesitan puntos.

2. Descarga las imágenes de las medallas que aparecen en la página y almacena en una carpeta denominada badges/ .

   Por ejemplo, la medalla relacionada con "Observador"la encontraras aquí:

   Medalla observador

   Sin embargo, lo que te interesa es la versión pequena. Es decir, Medalla observardor pequena

3. El script almacenalo bajo el nombre `get-badges.js` en la carpeta `scripts/`.

4. Crea una página ,leaderboard.html, que muestra las medallas y los puntos necesarios por medalla (ver Figura5.)

4 Competencias y Tareas

Si nos pones sobre un *skill* con el ratón, el hexagono relacionado se agaranda y muestra dos iconos (ver Figura6): un lápiz y un cuaderno. Además, en la parte inferior de la página, en el marco de una línea amarilla aparecerá la descripción de dicho *skill*.

En caso de pulsar sobre el cuaderno, se visualizará la página visible en Figura7 .

Ahí mismo, aparecen las especificaciones de una competencia. Es decir:

1. Titulo de la competencia

2. Breve descripción

3. Icono de la competencia

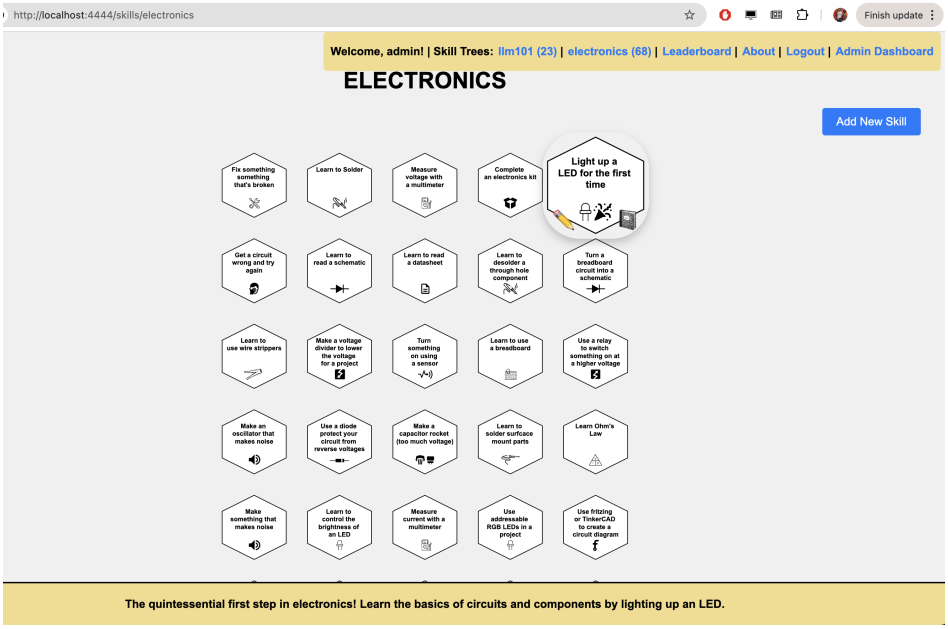


Figura 6: Deben aparecer símbolos para editar los datos de un *skill*y ver las tareas del mismo cuando se mueve el ratón sobre un *skill*

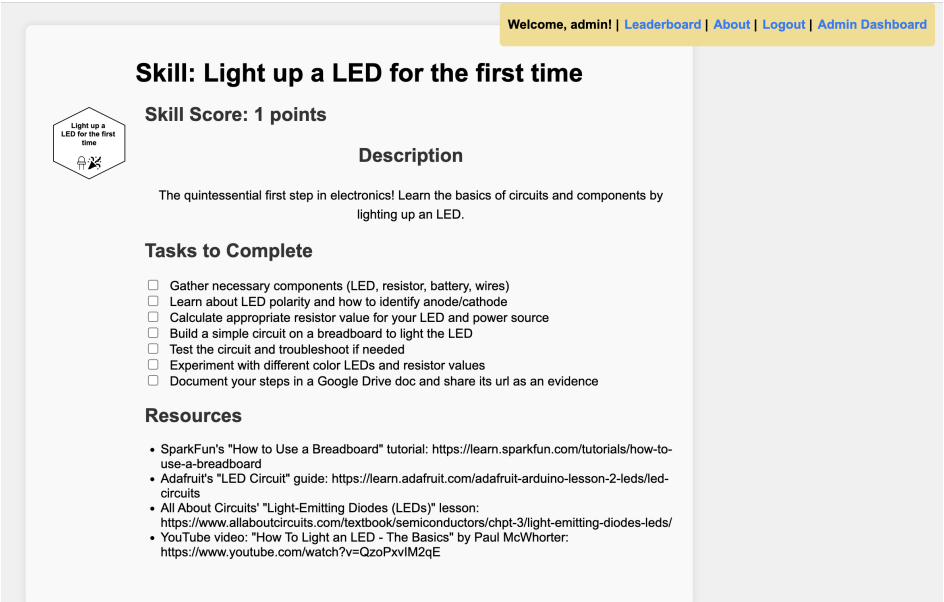


Figura 7: Especificaciones de la competencia

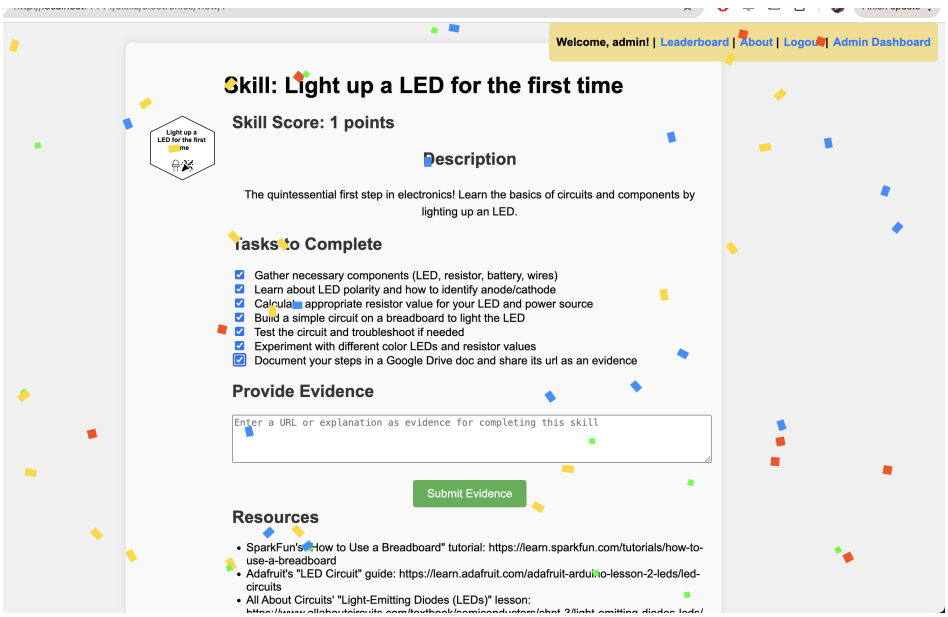


Figura 8: Al rellenar todas las tareas aparecerá una animación de confetti junto con un formulario de aportación de evidencia de que se han realizado tareas

4. Tareas que se deben realizar para lograr la competencia

5. Recuersos relacionados con la competencia (documentos de ayuda para realizar las tareas, URLs, vídeos, etc.)

Al lado de cada tarea se encuentra un checkbox. El usuario a medida que va realizando tareas, pulsará sobre el checkbox. Cuando cumpla todas las tareas de un *skill*, se visualizará una animación de confeti junto con un formulario que aporta evidencia de la realización de las tareas (ver Figura8). En ella, en general, el alumno escribirá una URL de un documento donde se explica cómo ha rellenado las tareas (y lo pulsaré en el botón 'Submit Evidence' para guardar la evidencia).

5 Evidencias

Las evidencias recibirán una evaluación recíproca. Ejemplo 9. Como se observa en la imagen, un usuario ha enviado una evidencia relacionada con un *skill* y que estará disponible para su evaluación para aquellos que hayan superado ese *skill* (o para los administradores). El evaluador puede analizar y aceptar o anular la evidencia (ver Figura10).

5.1 Ejercicios

Por el momento (en front-end) solo se requiere la preparación de las pantallas que aparecen en las imágenes anteriores (mockup-up, no enviarán nada al servidor). Es decir:

1. Debe aparecer un lápiz y un cuaderno cuando movemos el ratón sobre un *skill* (Figura6)

2. También, el formulario para enviar la evidencia y el efecto confeti (Figura8).

3. Prepara la página que visualiza las tareas y su decripción por skill. (Figura7).

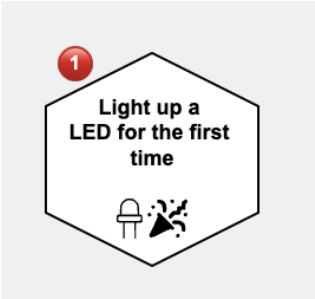


Figura 9: Las evidencias se deben aceptar

4. elabora un círculo rojo de las evidencias que un *skill* tiene pendientes de analizar. Se pintará para analizar el número de evidencias que hay dentro del círculo(por ejemplo, en la Figura10 sólo hay 1)

6 Confirmación de competencia

Para acreditar una capacidad se necesita el visto bueno de los 3 alumnos (o de un solo admin). Una vez comprobado un *skill*, su hexagono se visualizará en verde (ver Figura11)).

En general, para conocer el significado de los círculos e iconos ver la Figura12.

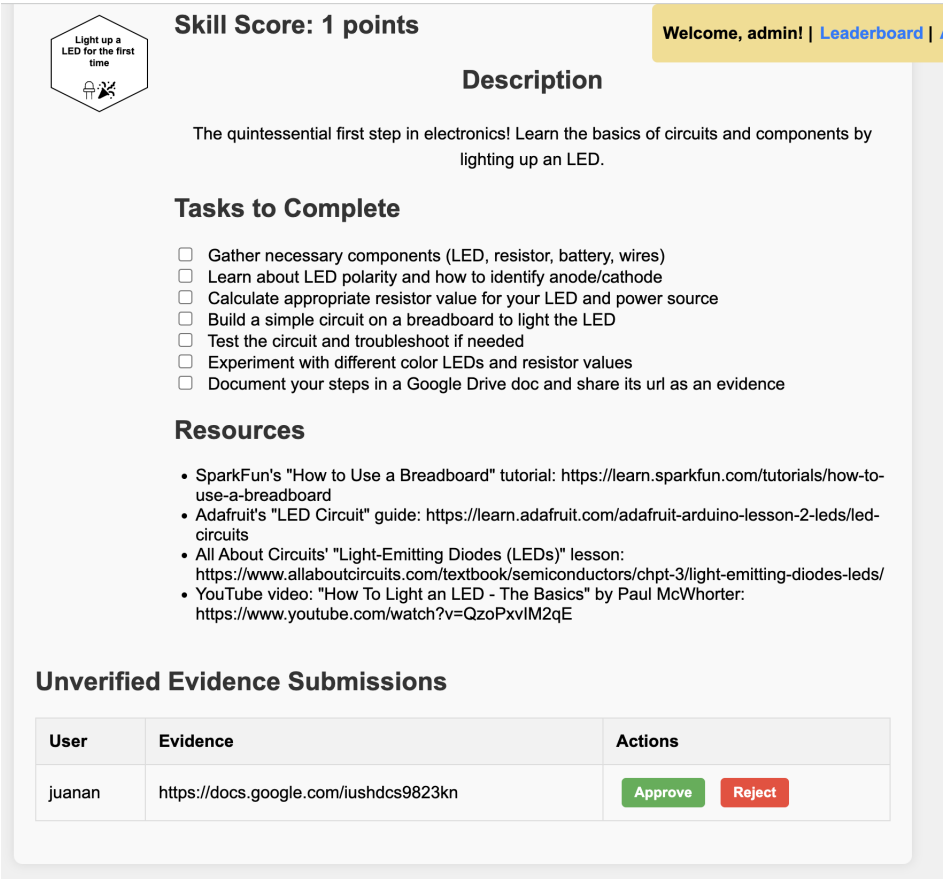


Figura 10: El evaluador puede aceptar o rechazar una evidencia tras su analisis.

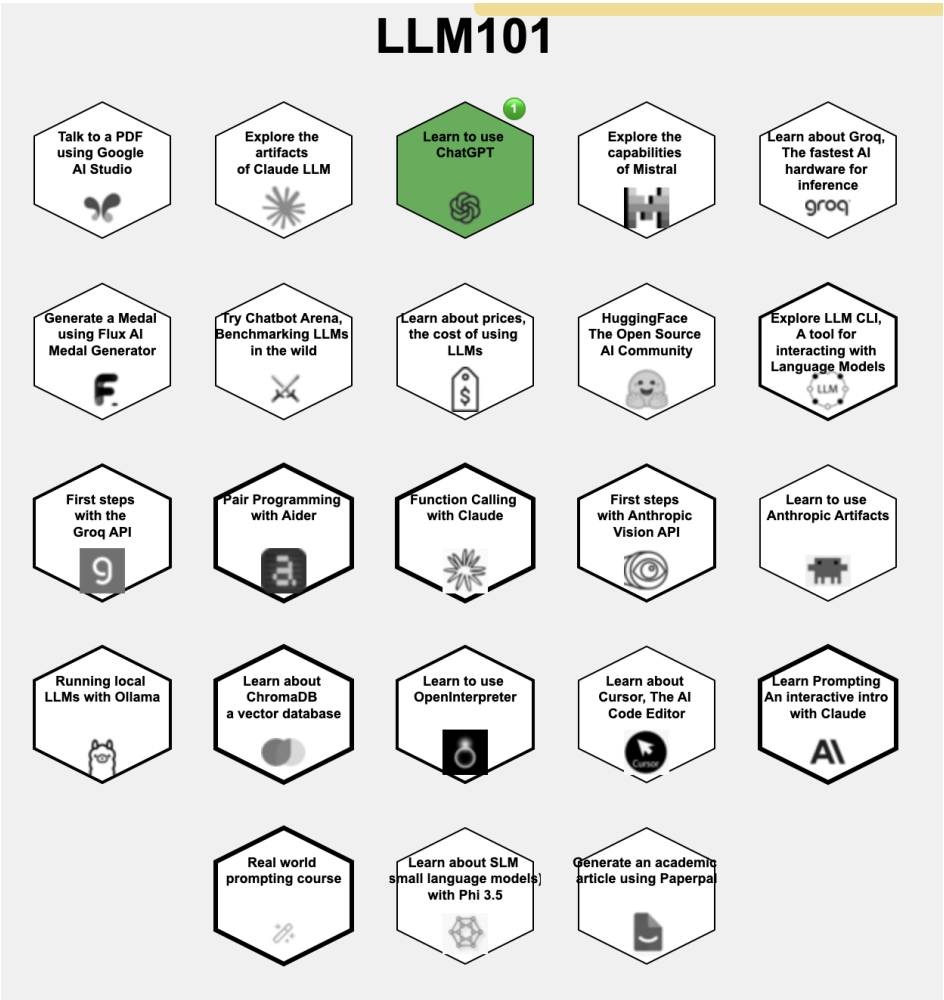


Figura 11: Para confirmar una competencia se necesita el visto bueno de 3 alumnos, o 1 admin.

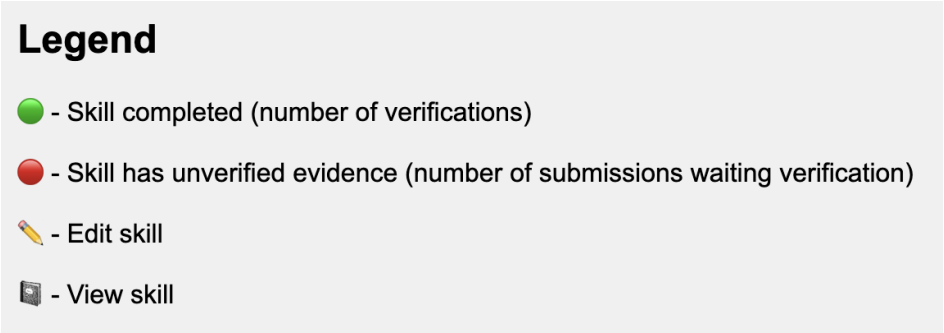


Figura 12: Significado de cada círculo e icono

7 ¡Lo has logrado!

Zorionak

Si has llegado hasta aquí, Zorionak! Has realizado un bonito trabajo con la parte de Front-end. Todavía queda trabajo por delante: las funcionalidades que hemos dejado de lado en el front-end (por ejemplo, la autenticación), la programación para el back-end, la integración entre las dos... y otras sorpresas.