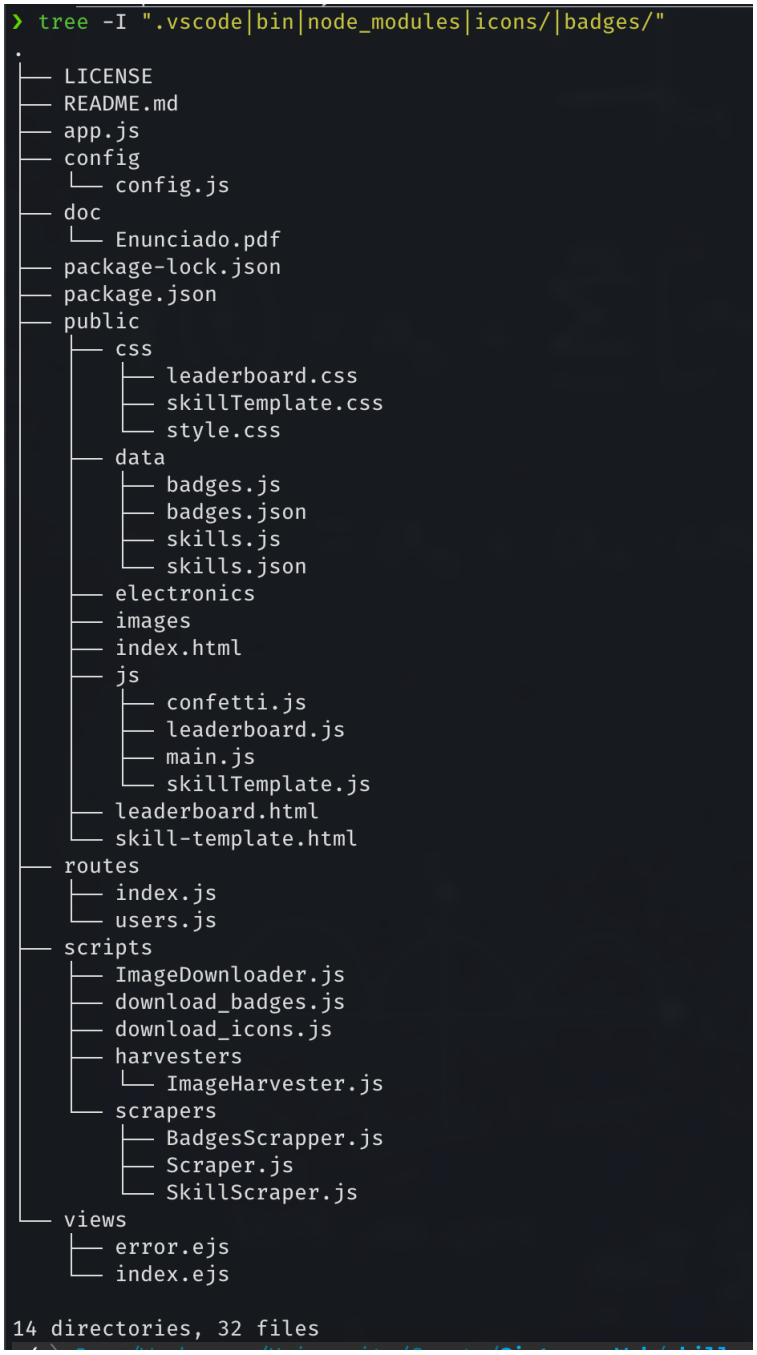
Skills Project - Frontend

Sistemas Web - proyecto

Aingeru García Blas
Joseba Gómez Rodriguez

Índice

1. Estructura del proyecto



Hemos organizado el proyecto en varias carpetas y módulos básicos para para realizar las tareas de scraping, recolección (harvesting) y exportación de datos, así como la lógica del Frontend para mostrar lo requerido por el enunciado. Veamos varios de los componentes clave:

config/

La carpeta `config/` tiene los archivos de configuración (uno por ahora, pero hemos pensado en la siguiente entrega), donde definimos rutas, endpoints y otros parámetros que utilizan los módulos de scraping y harvesting.

---

scripts/

scrapers/

Esta carpeta contiene el conjunto de clases que hemos diseñado para realizar tareas de scraping utilizando `cheerio`. La clase base es `Scraper`, de la cual heredan otras clases especializadas. Cada scraper tiene un enfoque particular para extraer datos específicos según el contexto. Por ejemplo, `SkillScraper` hereda de `Scraper` y extrae información sobre skills (como el texto, iconos asociados, etc.).

- Clase `Scraper`: se encarga de establecer la conexión al endpoint y de cargar el HTML. Utiliza librería `cheerio` como scraper.

- Clase `SkillScraper`: extiende `Scraper` para procesar datos específicos de skills. Implementa métodos como `harvestSkills`, que sirve para identificar y extraer datos de los elementos SVG (ID, texto, y links a iconos...etc). También hemos provisto de funciones para exportar las habilidades en formato JSON.

- Clase `BadgeScraper` Misma idea que SkillScraper, pero para badges. Con su export, capacidad de recolectar/harvestear de lo proporcionado por el scraper etc.

---

harvesters/

La carpeta `harvesters/` contiene clases que funcionan como recolectores de datos ya scrapeados. Por ejemplo:

- Clase `ImageHarvester`: se centra en la descarga de imágenes desde los endpoints, con URLs generadas en función de los datos scrapeados. Básicamente se encarga de almacenar los iconos en el directorio que corresponda (múltiples imágenes en paralelo). La parte interesante es que es configurable, y se adapta al scraper que le pasemos.

`Downloaders:`

- `download_icons.js`: Configura y ejecuta el flujo de scraping y descarga de iconos.
- `download_badges.js`: Lo mismo, pero para las badges.

Ejemplo de flujo de ejecución

Para un proceso de recolección/harvesteo de iconos, tendremos el siguiente flujo:

1. `ImageHarvester` se inicializa con el endpoint de iconos.
2. `setupScraper` configura `SkillScraper` para obtener las skills.
3. Por último, el método `harvestImages` descarga los iconos (en ruta especificada en `config`).

Documentos HTML

- `index.html` - Página de inicio y visualización de habilidades.
- `leaderboard.html` - Página que presenta una tabla de rangos.
- `skill-template.html` - Página de plantilla para mostrar información algo más detallada de los skills.

`index.html`

Árbol de skills principal, con elementos SVG, que generamos dinámicamente desde `main.js`. Su núcleo es un contenedor donde presentamos los SVG y una caja de descripción para visualizar información sobre cada skill cuando el usuario pasa el cursor sobre ella.

`leaderboard.html`

`leaderboard.html` muestra una tabla con explicaciones de rangos que hemos scrapeado de un `README.md` de la cuenta de Github del gran Obijuan, se muestran datos de rangos, puntos, iconos distintivos etc. Cada fila incluye información sobre el rango, el iconos, por ejemplo. Los datos de esta tabla los generamos dinámicamente en `leaderboard.js`.

skill-template.html

`skill-template.html` es una plantilla para visualizar detalles de cada skill individual. Muestra información de una skill específica; su descripción, icono, lista de tareas y recursos. También se permite que el usuario envíe evidencias de cumplimiento de la skill. El archivo incluye:

- `skillTemplate.js` para generar dinámicamente los contenidos.
- `confetti.js` efectos visuales de confeti cuando todas las tareas de una habilidad están completadas.

Archivos JavaScript

- `main.js` - Renderizado y control de skills en `index.html`.
- `leaderboard.js` - Generación de la tabla de rangos en `leaderboard.html`.
- `skillTemplate.js` - Tareas y evidencias en `skill-template.html`.
- `confetti.js` - Efecto visual.

main.js

Creación y visualización de las skills dentro de `index.html`. Al cargarse, monta los elementos SVG en base a los datos de skills (scrapeados + harvest) y aplica colores específicos (en función del progreso, etc.). Funciones principales:

- `buildSkills` - Crea el contenido SVG dinámicamente.
- `applyHexagonColors` - Aplica colores a cada skill según el estado almacenado en `localStorage`.

leaderboard.js

Maneja la carga y generación dinámica de la tabla de rangos en `leaderboard.html`. Extrae datos de los rangos y crea filas para mostrar los iconos, nombres y rangos de puntos (intervalos puestos aleatoriamente, por ahora). Función clave:

- `loadBadgeTable` - Genera cada fila en la tabla (a partir de los datos de rangos).

skillTemplate.js

`skillTemplate.js` se encarga de gestionar las tareas y la presentación de una skill en `skill-template.html`. Genera la interfaz donde los usuarios pueden ver tareas, agregar evidencia, etc. Funciones destacadas:

- `renderSkillTemplate` - genera la interfaz de la habilidad a partir de su ID y datos.
- `handleEvidenceSubmit` - enviar evidencia de cumplimiento.
- `updateHexagonColor` - Cambia el color del hexágono de la skill.

`confetti.js`

`confetti.js` implementa efecto de confeti cuando un usuario completa todas las tareas de una skill en `skill-template.html`.

2. Sobre la integración completa de datos

Debido a la falta de acceso a datos durante el proceso de scraping, hemos configurado el frontend del proyecto de manera que, cuando estos datos estén disponibles y contemos con una gestión de datos más granular y segura desde el backend, podamos integrar la lógica completa de manera adecuada. Ahora mismo, algunos elementos del proyecto están `hardcodeados` en esta entrega para simular la funcionalidad y permitir el desarrollo y visualización básica. Ya que en el enunciado aparecen varios de ellos y, sin embargo, aún no es posible gestionarlo.

31 de Octubre de 2024