

Scraping Teddy Analytics

David Jimenez Ribelles
Backend programmer

Agradecimientos

Principalmente quiero dar un agradecimiento al profesorado de FP Mislata, qué es de los mejores que he tenido, se ha notado que se preocupaban por nosotros sus alumnos y qué había muy buen rollo, pero lo más importante es que enseñan de maneras muy particulares y si no llevas el ritmo de la clase te apoyan.

También es importante nombrar a mis compañeros los cuales junto al profesorado hemos sido como una gran familia, compartiendo historias, ayudándonos en nuestras dudas y aprendiendo juntos.

También creo que es necesario por mi parte agradecer a [webscraper](#) sin esta página no podría haber realizado este proyecto ofrece sitios de prueba donde puedes scrapear diferentes tipos de páginas y una extensión que resulta bastante útil, de todas maneras he decidido hacer los mínimos scrapeos posibles para no causarles problemas con servidores.



Índice

Agradecimientos	1
Licencia	4
1 Marco de investigación	5
1.1 Temática Elegida	6
1.2 Contextualización	7
2 Organización del proyecto	8
2.1 Temporalización	8
2.1 Recursos	9
3 Aplicación práctica.	15
3.1 Introducción	15
3.1.2 Diseño	16
3.1.3 Implementación	17
3.2.3.1 Aplicación de Scraping	21
3.2.3.2 Aplicación SQL	25
3.2.3.3 Aplicación API	26
3.2.3.4 Aplicación Angular	30
4 Manual de Usuario	32
5 Valoración Personal	39
6 Fuentes Bibliográficas	40



Licencia

Este obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial 4.0 Internacional.



<https://creativecommons.org/licenses/by-nc/4.0/>

1 Marco de investigación

El objetivo de este proyecto es aprender java mediante el framework **Spring**, y desde ese framework usar la librería de código abierto **Jsoup** para leer documentos html y **JPA** usado para crear la api, también se ha usado **Angular** y **Bootstrap** para el FrontEnd.



He usado una base de datos relacional y como gestor he usado **HediSQL**, además para gestionar mis tareas he usado **Trello** que es una aplicación para trabajar con el método SCRUM para así llevar un seguimiento de mis avances y tener una mejor organización

1.1 Temática Elegida

La parte más extensa de la aplicación es la parte hecha con Java Spring, lo he decidido investigar ya que me gustaba Java y quería ver cómo funcionaba la extensión y usarlo en el Backend.

Lo más importante de la aplicación es el método que se usa para llenar los datos en la base de datos, he usado una técnica llamada scraping y para implementarla he usado una librería de Java llamada JSOUP la te proporciona bastantes herramientas muy útiles.

Sabiendo esto, he decidido hacer mi aplicación sobre una aplicación de administrador de una tienda online en la que puede comparar los precios de la competencia y modificarlos en base a los datos obtenidos por el scraping.

Pienso que el scraping tiene muchas más posibilidades pero considero que esta es una de las que más rentabilidad le pueden dar.

La página de la que se han obtenido los datos es una pagina de practica de scraping

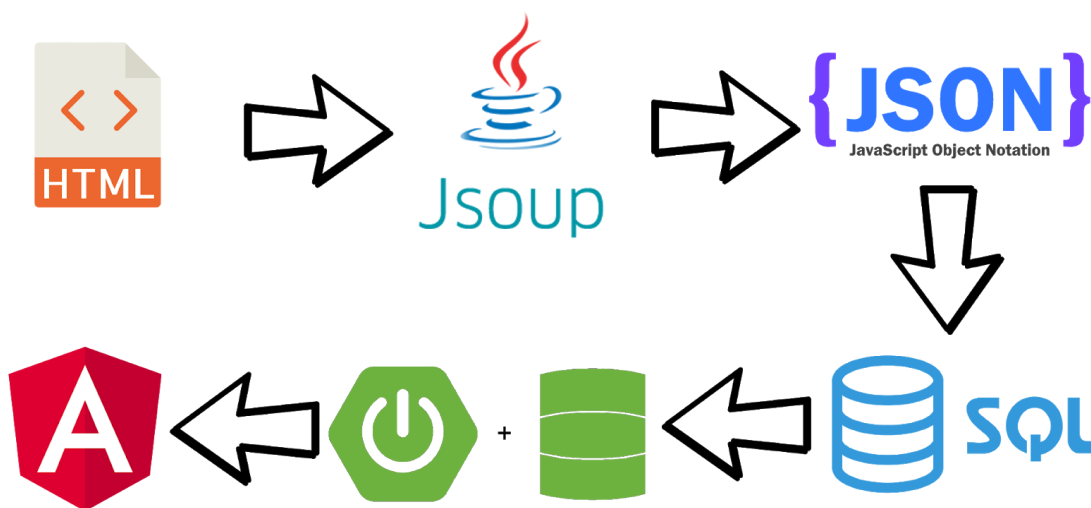
1.2 Contextualización

Se me ocurrió la idea al aprender que era el scraping, me encantó las posibilidades que existían con ese sistema y decidí crear una aplicación en la que se puedan apreciar su funcionalidad.

Este tipo de aplicación tiene varios usos, pero uno de los más importantes es recoger datos de otras empresas de la competencia para averiguar sus productos de manera automática en cada momento.

Lo más común es realizar una aplicación por cada proceso de los datos para que sea más fácil de actualizar y mantener, es decir tendríamos una para scrapear que lo guardaría en una Base de datos en formato JSON, otra para traducir el JSON a SQL, otra sería la API en JPA y por último la parte de Angular.

Cada vez que se ejecute el Scrapeo actualizará la Base de Datos y cambiará el contenido de nuestra página.



2 Organización del proyecto

2.1 Temporalización

El desarrollo del proyecto se podría estructurar en 3 fases.

Estudio

Cuando empecé en este proyecto no sabía nada sobre ninguna de las tecnologías usadas excepto Angular que habíamos dado en el curso así que tuve que estar dos semanas aprendiendo técnicas de scrapeo, estudiando las diferentes forma de realizar una API con Java y ampliando mis conocimientos en Angular.

Planificación

En esta fase empecé a pensar en la estructura que iba a tener el proyecto ya que tiene muchas partes quería tener todo claro antes de empezar para evitar imprevistos, tambien me hice un planning en Trello en el que iba marcando objetivos por día y los iba completando.

Gracias a esta manera de planificar las tareas he comprendido lo importante que es organizarse cualquier trabajo.

Desarrollo de la App

Creando el proyecto no tuve muchas dificultades gracias a los conocimientos adquiridos previamente y a lo estructurada que tenía la creación del proyecto.

Mientras realizaba la aplicación anotaba cosas para poder usarlas en la memoria, así recordaría las cosas que necesitaba comentar en la memoria que iba a realizar después.

Lo más complicado de la aplicación ha sido sin duda el diseño, pero después de muchas pruebas para ver como me gustaba más conseguir hacer uno bastante bonito.

2.1 Recursos

En este proyecto he usado varios lenguajes de marcado, IDE, lenguajes de programación, frameworks, extensiones, sistemas gestores de base de datos, contenedores web y librerías

Lenguajes de marcado usados:

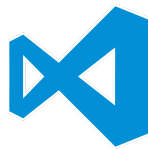


HTML es uno de los lenguajes de marcado más populares de la historia, se estructura mediante etiquetas y es uno de los lenguajes que es casi obligatorio aprender ya que se usa para casi todo.



CSS también es uno de los más usados, si aprendes HTML vas a sentir la necesidad de aprender CSS ya que se complementan, ahora mismo todas las páginas web usan CSS.

IDE usados:



Visual Studio Code es uno de los mejores IDE que he probado es potente a la par que ligero, ofrece muchas extensiones y es muy personalizable. Es compatible con casi todos los lenguajes de programación más usados actualmente y tiene una comunidad creando contenido continuamente.



Eclipse es un IDE gratuito de código abierto usado principalmente para Java, es muy potente y ofrece muchas opciones. Tiene un market en el que hay extensiones, temas y librerías para añadir desde el propio IDE.

Lenguajes de programación usados:



TypeScript es un lenguaje de programación de código abierto basado en JavaScript, en este lenguaje hay nueva sintaxis, interfaces, decoradores, tipado de datos, programación orientada a objetos. Al ser una variante de JavaScript con tantos añadidos transpila el código a JavaScript.



Java es un lenguaje de programación que genera software multiplataforma que pueden ser usados en la mayoría de sistemas operativos, trabajar en java requiere un Java Development Kit (JDK) que aporta diversas herramientas de desarrollo

Frameworks usados:



Angular es un framework de código abierto diseñado para crear aplicaciones web de una manera más completa y estructurada, modifica: etiquetas html, sintaxis de JavaScript y agrega funcionalidades a TypeScript. Se ha hecho muy famoso gracias al tiempo que ahorra al hacer una aplicación web compleja.

Extensiones usadas:



Spring Boot a diferencia de Spring no es un framework sino una extensión que aporta una configuración que ayuda a la creación de aplicaciones web mediante los IDE Eclipse y Netbeans.

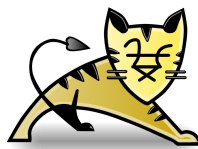
Sistemas gestores de bases de datos usados:



HeidiSQL es un sistemas gestor de base de datos de código abierto muy ligero, con muchas opciones usado para administrar las bases de datos

.

Contenedor web usado:



Tomcat es un contenedor web que soporta servlets y JSP, usa Jasper como compilador para compilar los JSP y convertirlos en servlets.

Librerías usadas:



Jsoup es una librería del lenguaje Java que se usa para obtener el contenido HTML de una página mediante su url y manipularlo mediante búsqueda con selectores y DOM.



Bootstrap es una colección muy grande de código útiles y reutilizables escritos en HTML, CSS y JavaScript, además es un framework que permite diseñar de manera mas facil y rapida.

3 Aplicación práctica.

3.1 Introducción

Mi aplicación esta creada en Angular y tiene una API hecha con Java y sobretodo está enfocada a las empresas como tiendas online o realmente para cualquier empresa que quiera medirse con su competencia.

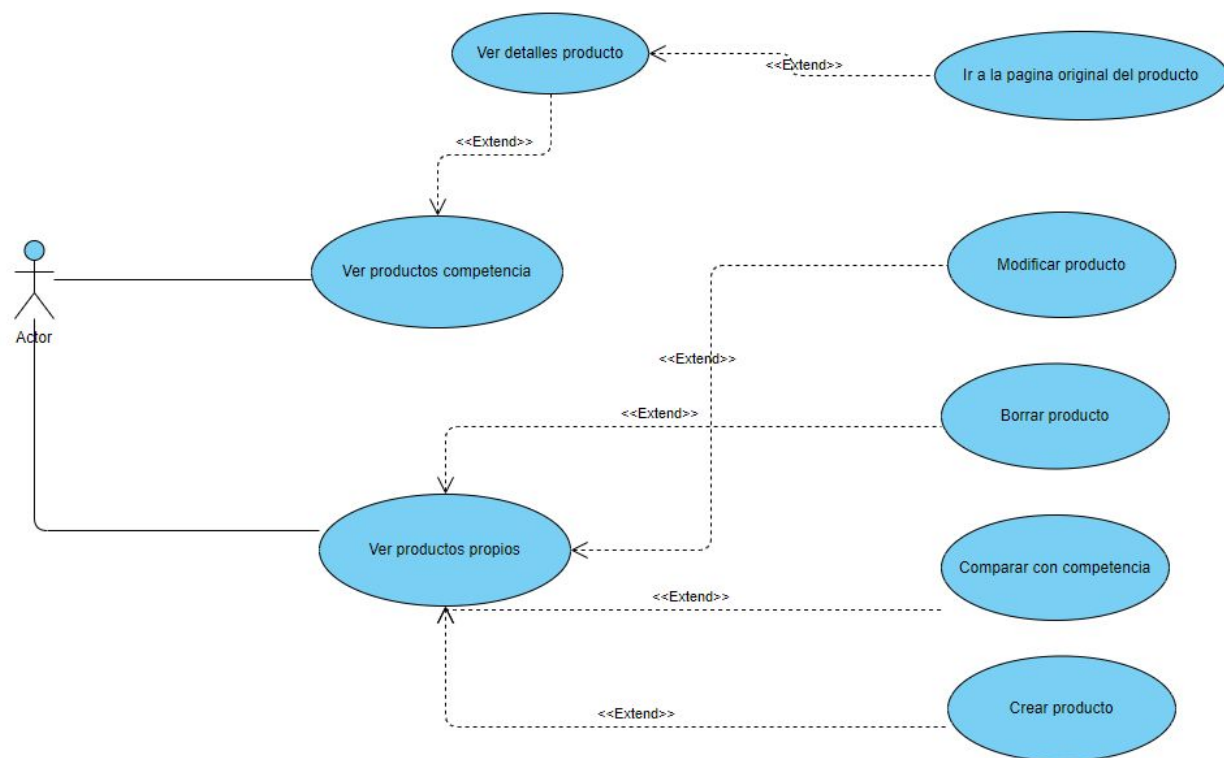
La aplicación de scrapeo está hecha de manera que de forma fácil pueda ser manipulada en caso de cambio de distribución del documento html.

Pensé en esta idea a raíz de descubrir lo que era el scraping, en el momento me pareció una de las mejores aplicaciones que se le puede dar al scraping.

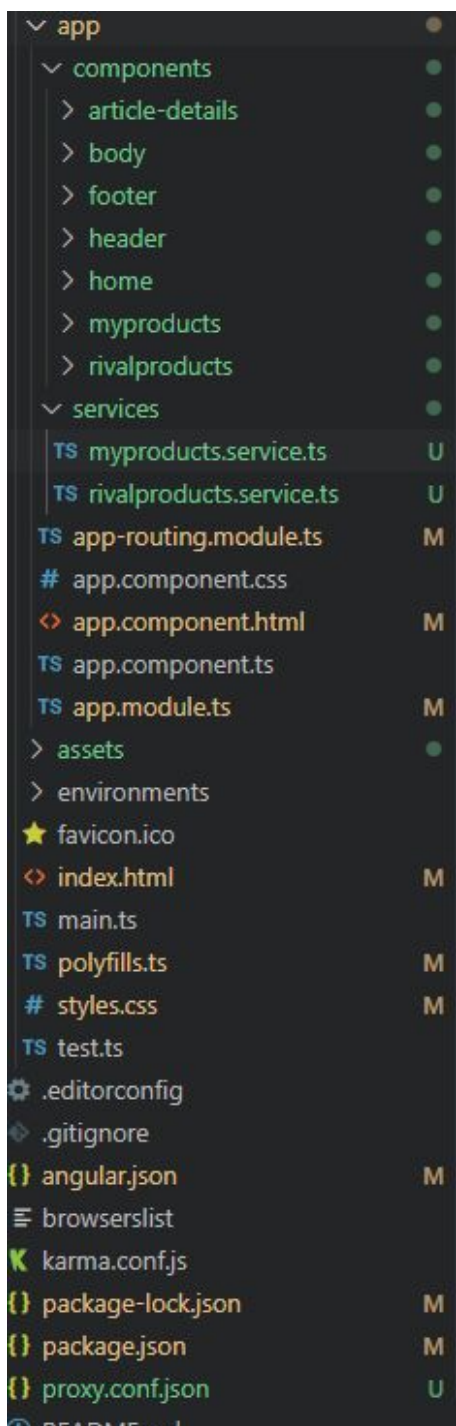
La parte de Angular es de simple uso no contiene ningún tipo de contenido complicado de entender consta con un CRUD para editar, crear y borrar los artículos y también tienes la opción de comparar el precio de un artículo tuyo con los precios de la competencia

Además tienes una lista de los productos de la competencia en la que puedes acceder y ver todos los datos que tienen en su página y un link que te redirige a su página.

3.1.2 Diseño



3.1.3 Implementación



Esta es la estructura de carpetas usadas en la parte de Angular

En esta parte se encuentran 7 componentes y 2 servicios.

Hay solamente dos imágenes en assets, una el favicon y la otra la imagen de la página principal.

Además cuenta con un proxy.conf.json el cual he usado para poder evitar el problema que suponía el CORS

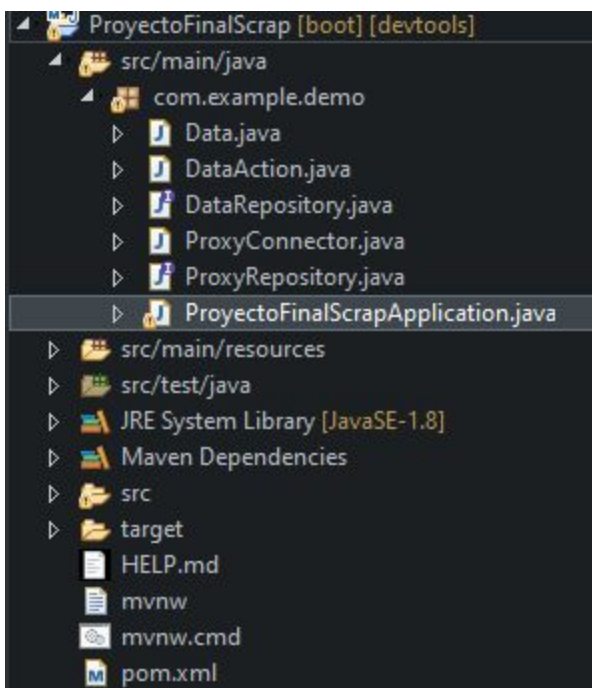
Aunque la parte de back también tiene definidos los permisos del CORS

El POM es un archivo xml donde se introducen las dependencias de la aplicación maven, hereda de un SuperPOM

En todas las aplicaciones hay una dependencia que es el connector de MYSQL

Estructura de carpetas de la aplicación de scrap y el POM

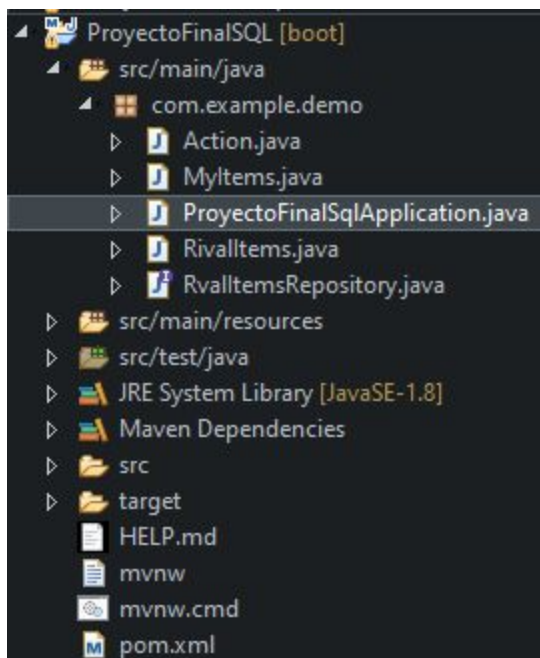
Las dependencias importadas en esta parte del proyecto han sido JSOUP para obtener y obtener los datos, JPA para la conexión con la Base de Datos y Jackson, una dependencia para poder transformar objeto a JSON



```
<dependencies>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
  </dependency>
  <dependency>
    <groupId>org.jsoup</groupId>
    <artifactId>jsoup</artifactId>
    <version>1.13.1</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    <exclusions>
      <exclusion>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
</dependencies>
```

Estructura de carpetas de la aplicación de SQL y el POM

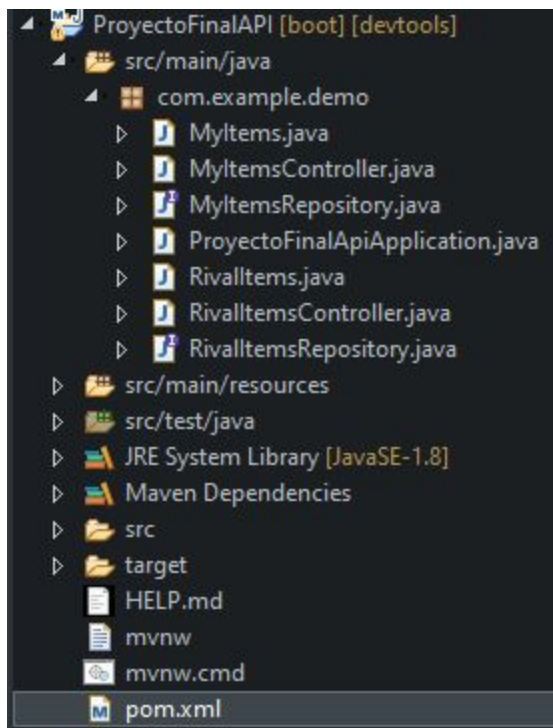
Las dependencias importadas en esta parte del proyecto han sido manipuladores o transformadores de JSON y JPA para la conexión con la Base de Datos



```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    <exclusions>
      <exclusion>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
</dependencies>
```

Estructura de carpetas de la aplicación de API y el POM

Las dependencias importadas en esta parte del proyecto han sido las mismas que en las demás y JPA para la conexión con la Base de Datos



```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
    <exclusions>
      <exclusion>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
</dependencies>
```


3.2.3.1 Aplicación de Scraping

Para empezar hay que generar un JSON con todos los datos

Este seria el metodo principal donde Se obtendrían los datos y haría todas las acciones de scraping

```
@Scheduled(cron = "0 0 23 ? * *")
public static void scrapPage() throws JsonProcessingException {

    Document page = getHtmlDocument(urlFiltros);
    Elements repositories = page.select("h4 > a.title");

    for (Element mainPage : repositories) {
        Map<Object, Object> allData = new HashMap<>();
        String itemUrl = null;
        String pageId = null;

        try {

            itemUrl = url + mainPage.attr("href");
            System.out.println(itemUrl);
            try {
                pageId = itemUrl.split("/")[7];
            } catch (IndexOutOfBoundsException e) {}

        } catch (NullPointerException e) {}

        Document selectedAd = getHtmlDocument(itemUrl);

        Map<String, Object> pageData = new HashMap<>();
        Map<String, Object> itemData = new HashMap<>();

        Map<String, String> itemSelectors = new HashMap<>();

        itemSelectors.put("price", "div.caption > h4.price");
        itemSelectors.put("description", "p.description");
        itemSelectors.put("swatch", "div.swatches > button");
        itemSelectors.put("ratings", "div.ratings > p");

        itemData.put("id", pageId);
        itemData.put("title", selectedAd.select("div.caption > h4").get(1).text());
        itemData.put("images", selectedAd.select("img.img-responsive").first().attr("src"));
        for (String selectedData : itemSelectors.keySet()) {
            try {

                String str = selectedAd.select(itemSelectors.get(selectedData)).text().replace("$", "");

                itemData.put(selectedData, str);

            } catch (NullPointerException e) {
                System.out.println("Falta el campo: " + selectedData);
            }
        }

        pageData.put("url", itemUrl);
        pageData.put("itemData", itemData);

        allData.put("article", pageData);

        ObjectMapper objM = new ObjectMapper();
        dataAction.store(new Data(objM.writeValueAsString(allData)));
    }
}
```

Antes de empezar a explicar del código hay que dejar claro lo que son las anotaciones, las anotaciones de Java son indicaciones que se le dan al compilador definiendo que hace el elemento al que se le ha aplicado

Para poder programar el actualizado de datos de la página para una hora determinada necesitamos varias anotaciones la primera es

```
@EnableScheduling
@SpringBootApplication
public class ProyectoFinalScrapApplication {
```

@EnableScheduling en la clase.

Esto nos permitirá usar diferentes anotaciones en los métodos para programar tiempos de espera, horarios a los que se tienen que usar o tiempo de espera entre métodos.

```
@Scheduled(cron = "0 0 23 ? * *")
public static void scrapPage() throws JsonProcessingException {
```

Usando el parámetro cron le indicamos con expresiones cron a que hora quieres que se use el método en formato (segundos minutos horas ? dia_semana mes)

Usando un método creado por nosotros y poniendo la url podremos obtener el documento html y con selectores podemos obtener todos los links de los artículos de la página.

Este es el link de la página scrapeada, en este caso ha sido la categoría de portátiles pero podría haberse hecho un scrapeado general

<https://webscraper.io/test-sites/e-commerce/allinone/computers/laptops>

```
Document page = getHtmlDocument(urlFiltros);
Elements repositories = page.select("h4 > a.title");
```

Los "Elements" contienen todas las etiquetas con los links de cada artículo se pueden recorrer con un bucle y se puede obtener su link, usar el método para guardar el documento html y coger los datos mediante selectores

```
for (Element mainPage : repositories) {
    Map<Object, Object> allData = new HashMap<>();
    String itemUrl = null;
    String pageId = null;

    try {

        itemUrl = url + mainPage.attr("href");
        System.out.println(itemUrl);
        try {
            pageId = itemUrl.split("/")[7];
        } catch (IndexOutOfBoundsException e) {
        }

    } catch (NullPointerException e) {}

    Document selectedAd = getHtmlDocument(itemUrl);
```

Es importante identificar la id de cada producto, en este caso en el link estaba la id, con esa información se pueden realizar peticiones a la base de datos de cada producto y así se evitaría una de las "barreras contra el scrap".

Dentro del bucle al final generamos un JSON desde un map con el ObjectMapper y lo guardamos en la BD


```
    ObjectMapper objM = new ObjectMapper();

    dataAction.store(new Data(objM.writeValueAsString(allData)));
}
```

Cuando practicas scraping para aprender no necesitas una gran cantidad de datos ni los usas para comercializar, pero en el caso de que necesites scrapear muchos datos de una página lo más probable es que te bloqueen por demasiadas peticiones desde el mismo sitio, con el mismo patrón o demasiado rápido, con este método se lanzan peticiones más lentas, a diferentes tiempos y desde muchos sitios así que es menos probable que te bloqueen la dirección

```
public static Document getHtmlDocument(String url) {
    Random random = new Random();

    ProxyConnector usedProxy = null;
    Document doc = null;

    try {
        usedProxy = proxyList.get(random.nextInt(proxyList.size()));
        try {
            TimeUnit.SECONDS.sleep((int) (usedProxy.getDelay()));
        } catch (InterruptedException e1) {
            e1.printStackTrace();
        }
        doc = Jsoup.connect(url).proxy(usedProxy.getUrl(), usedProxy.getPort()).userAgent("Mozilla/5.0")
            .timeout(24000).get();
    } catch (Exception ex) {
        System.out.println(ex);
    }

    return doc;
}
```

Los proxys son añadidos en la Base de Datos, hay muchas listas de proxy gratis.

3.2.3.2 Aplicación SQL

```
@SpringBootApplication
public class ProyectoFinalSqlApplication {

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/proyectofinalscrap";

    static final String USER = "root";
    static final String PASS = "";

    public static List<String> allData = new ArrayList<>();

    public static void main(String[] args) throws JsonMappingException, JsonProcessingException {
        ConfigurableApplicationContext context = SpringApplication.run(ProyectoFinalSqlApplication.class, args);
        Action action = context.getBean(Action.class);
        getJsonData();
        fillSchema(action);
    }

    public static void fillSchema(Action action) throws JsonMappingException, JsonProcessingException {
        ObjectMapper obj = new ObjectMapper();
        for (String json : allData) {
            JsonNode jsonNode = obj.readTree(json).get("article").get("itemData");
            RivalItems rivalitem = new RivalItems(jsonNode.get("id").asInt(), jsonNode.get("title").asText(),
                jsonNode.get("description").asText(), jsonNode.get("price").asDouble(),
                jsonNode.get("ratings").asText(), jsonNode.get("swatch").asText(), jsonNode.get("images").asText(),
                obj.readTree(json).get("article").get("url").asText());

            action.store(rivalitem);
            try {

            } catch (NullPointerException e) {
                System.out.println(e);
            }
        }
    }

    public static void getJsonData() {
        try {

            Connection conn = DriverManager.getConnection(DB_URL, USER, PASS);

            Statement stmt = conn.createStatement();

            ResultSet rs = stmt.executeQuery("SELECT content FROM data");

            while (rs.next()) {

                allData.add(rs.getString("content"));

            }
        } catch (SQLException se) {
            se.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Esta es la aplicación que obtiene los json de la base de datos y los traduce a sql los guarda como objetos rivalItems ya que hemos scrapeado a la "competencia"

3.2.3.3 Aplicación API

El método de conectar la Base de datos que se puede usar en Java me ha sorprendido, es muy eficiente y limpio, funciona de la siguiente forma:

Introducimos el link de la Base de datos en el application.properties junto con el usuario y la contraseña

Creamos el modelo de la tabla que quieres administrar

Creamos un repositorio que va a ser el que haga todas las funcionalidades de una API

Y después se crea un controlador para introducir todas las rutas de la API, en el caso de no estar haciendo una Api yo prefiero usar una clase Action que contiene todas las acciones con los repositorios

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="rival_items")
public class RivalItems {

    @Id
    private int id;

    @Column(name="title")
    private String title;

    @Column(name="description")
    private String description;

    @Column(name="price")
    private double price;

    @Column(name="ratings")
    private String ratings;

    @Column(name="swatch")
    private String swatch;

    @Column(name="images")
    private String images;

    @Column(name="url")
    private String url;

    public RivalItems() {
        super();
    }
}
```

Aquí se puede apreciar todo lo necesario para crear un modelo funcional con el sistema de persistence de Java

Las clases deben tener la anotación *@Entity* y *@Table* con el parámetro name que definirá el nombre de la tabla, si es el mismo que el de la clase no hace falta ponerlo.

Cada atributo tiene la anotación *@Column* con el parámetro nombre y funciona igual que el de *@Table*.

Es necesario tener un constructor vacío para que la aplicación lo use

El que siempre debe estar es el *@Id* que definirá la ID del modelo

Este ejemplo de id es válido en una tabla con id sin Autoincremental en int

En el caso de tener Autoincremental hay que poner el tipo de datos en Long y usar la anotación `@GeneratedValue`, esta anotación contiene varios parámetros pero si usas los que salen en la foto te generará el Autoincremental

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
```

Para poder usar un repositorio para nuestro modelo necesitamos anotar que es un repositorio con `@Repository` y hacer extend de `JpaRepository`.

Para definir a qué modelo le hace su función se le especifica en los parámetros, el primero sería el objeto al que le va a hacer la función y el segundo será el tipo de datos de la ID, es decir en caso de que fuera autoincremental sería Long

```
@Repository
public interface RivalItemsRepository extends JpaRepository<RivalItems, Integer> {
    List<RivalItems> findById(int id);
}
```

Si queremos algún tipo de filtro o algún método extra hay que añadirlo dentro de la interfaz, los métodos tienen que ser sintácticamente correctos, no puede tener un nombre inventado por ejemplo si quisiera obtener todos los objetos con un nombre que introduzca pondría:
`List<RivalItems> findByName(String);`

Esta es la clase controladora, voy a explicarlo las partes más importantes ya que contiene bastante información

```
@RequestMapping("/api/rivalItems")
@RestController
public class RivalItemsController {

    @Autowired
    RivalItemsRepository repo;

    @GetMapping("")
    public List<RivalItems> index() {
        return repo.findAll();
    }

    @GetMapping("/{id}")
    public RivalItems show(@PathVariable String id) {
        return repo.findById(Integer.parseInt(id)).get(0);
    }

    @PutMapping("/{id}")
    public RivalItems update(@PathVariable String id, @RequestBody Map<String, String> body) {
        return repo.save(new RivalItems(Integer.parseInt(id), body.get("title"), body.get("description"),
            Double.valueOf(body.get("price")), body.get("ratings"), body.get("swatch"), body.get("images"),
            body.get("url")));
    }

    @PostMapping("")
    public RivalItems create(@RequestBody Map<String, String> body) {
        return repo.save(new RivalItems(Integer.parseInt(body.get("id")), body.get("title"), body.get("description"),
            Double.valueOf(body.get("price")), body.get("ratings"), body.get("swatch"), body.get("images"),
            body.get("url")));
    }

    @DeleteMapping("/{id}")
    public String delete(@PathVariable String id) {
        repo.deleteById(Integer.parseInt(id));
        return "Deleted";
    }
}
```

Estas son las anotaciones que se necesitan para crear un controller para nuestra API Rest.

El `@RequestMapping` define la ruta que va a tener la clase .

El `@RestController` indica qué tipo de clase es y cuál va a ser su función, en este caso hacer la función de escuchar las peticiones.

```
@RequestMapping("/api/rivalItems")
@RestController
public class RivalItemsController {
```

En esta parte es donde enlazamos el repositorio con el controller, y para eso usamos la anotación `@Autowired` que nos crea la instancia y nos enlaza la clase controladora a la interfaz de repositorio.

```
@Autowired  
RivalItemsRepository repo;
```

Cada método tiene una anotación arriba que define el método que se usa y en el parámetro se introduce la ruta que quieres que tenga ese método.

La función de `index()` es obtener todos los datos que hay en la Base de datos usando un método del repositorio.

```
@GetMapping("")  
public List<RivalItems> index() {  
    return repo.findAll();  
}
```

Este método obtiene el objeto con la id introducida.

En el parámetro de la anotación ahora tiene una ruta, se pueden definir parámetros de la ruta entre llaves para después en el parámetro del método usando la anotación `@PathVariable` se pueda obtener y usar

```
@GetMapping("/{id}")  
public RivalItems show(@PathVariable String id) {  
    return repo.findById(Integer.parseInt(id)).get(0);  
}
```

Este es el método Put.

Aquí está lo explicado previamente más una anotación llamada *@RequestBody* en los parámetros del método, eso contiene el json del body en forma de Map

```
@PutMapping("/{id}")
public RivalItems update(@PathVariable String id, @RequestBody Map<String, String> body) {

    return repo.save(new RivalItems(Integer.parseInt(id), body.get("title"), body.get("description"),
        Double.valueOf(body.get("price")), body.get("ratings"), body.get("swatch"), body.get("images"),
        body.get("url")));
}
```

3.2.3.4 Aplicación Angular

Este sería mi app.module.ts donde declaro todos los imports

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HeaderComponent } from './components/header/header.component';
import { BodyComponent } from './components/body/body.component';
import { HomeComponent } from './components/home/home.component';
import { ArticleDetailsComponent } from './components/article-details/article-details.component';
import { MyproductsComponent } from './components/myproducts/myproducts.component';
import { RivalproductsComponent } from './components/rivalproducts/rivalproducts.component';
import { HttpClientModule } from '@angular/common/http';
import { FooterComponent } from './components/footer/footer.component';
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    BodyComponent,
    HomeComponent,
    ArticleDetailsComponent,
    MyproductsComponent,
    RivalproductsComponent,
    FooterComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    NgbModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

Aquí es donde se incluyen las rutas, pero para ello es necesario importar el componente, si la ruta contiene ":" lo siguiente será una variable que se puede manipular más tarde

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './components/home/home.component';
import { ArticleDetailsComponent } from './components/article-details/article-details.component';
import { MyproductsComponent } from './components/myproducts/myproducts.component';
import { RivalproductsComponent } from './components/rivalproducts/rivalproducts.component';
import { BodyComponent } from './components/body/body.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'rivalproducts', component: RivalproductsComponent },
  { path: 'rivalproducts/:id', component: ArticleDetailsComponent },
  { path: 'myproducts', component: MyproductsComponent },
  { path: 'myproducts/:id', component: BodyComponent }
];
```

Los servicios son los encargados de obtener los datos ya sea mediante peticiones o con un array mockeado, en este caso están usando llamadas a la API creada por mi

Estos serían los dos servicios usados para realizar las peticiones.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class MyproductsService {
  url = '/api/myItems/';

  constructor(private http: HttpClient) {}

  getmyProducts() {
    return this.http.get(this.url);
  }

  getmyProduct(id) {
    return this.http.get(this.url+id);
  }

  deletemyProduct(id) {
    return this.http.delete(this.url+id);
  }

  insertmyProduct(myproduct) {
    return this.http.post(this.url, myproduct);
  }

  putmyProduct(id, myproduct) {
    return this.http.put(this.url+id, myproduct);
  }
}
```

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class RivalproductsService {
  url = '/api/rivalItems';

  constructor(private http: HttpClient) {}

  getRivalProducts() {
    return this.http.get(this.url);
  }

  getRivalProduct(id) {
    return this.http.get(this.url+"/"+id);
  }
}
```

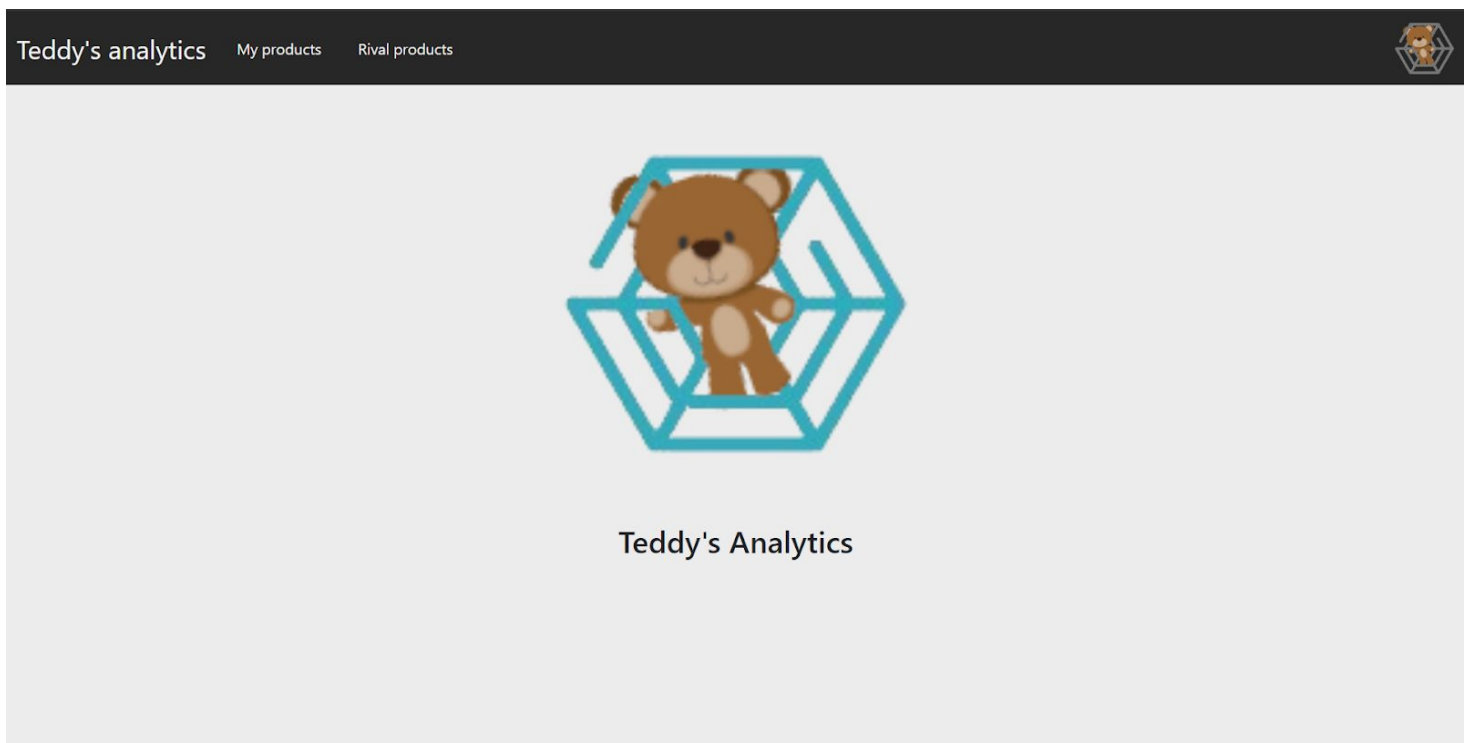
Manual de Usuario

Teddy Analytics



4 Organización del proyecto

La primera pantalla de la aplicación es el inicio donde está el logo, en la parte superior está la cabecera donde tienes tres opciones, ver tus productos, ver los productos de la competencia o clicar en el logo y te redirige a la página scrapeada.




Esta aplicación es un CRUD con una funcionalidad extra la cual seria los datos de la competencia sin necesidad de hacerlo de manera manual.

Este es el CRUD de los productos del propietario con la opción "Compare".

Teddy's analytics

My products

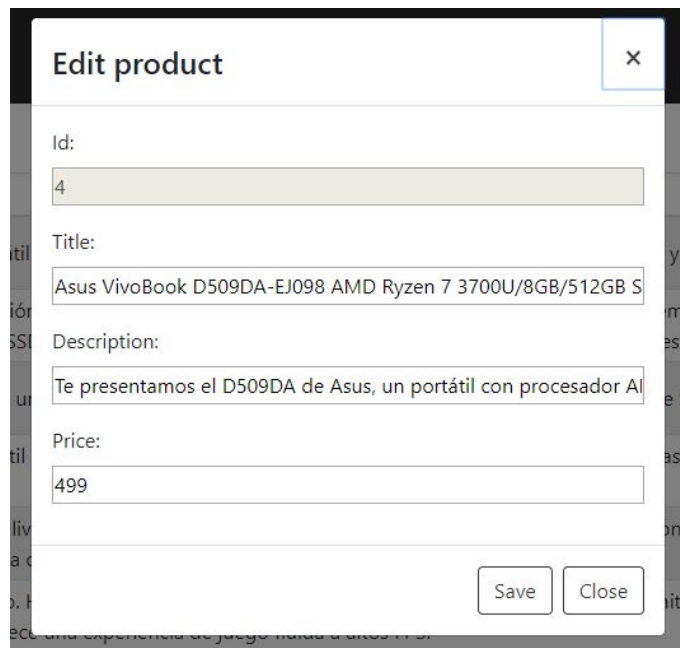
Rival products



Nuevo				
Id	Title	Description	Price	Actions
4	Asus VivoBook D509DA-EJ098 AMD Ryzen 7 3700U/8GB/512GB SSD/15.6"	Te presentamos el D509DA de Asus, un portátil con procesador AMD Ryzen 7, 8GB de RAM, 512GB SSD de disco duro y pantalla de 15.6".	499€	Modify Compare Delete
5	Lenovo IdeaPad S145-15IIL Intel Core i5-1035G1/8 GB/512GB SSD/15.6"	Con procesadores Intel Core de 10.ª generación, el IdeaPad S145 se ha diseñado para seguirte el ritmo, independientemente de la tarea. También incluye una gama de opciones de almacenamiento seguras, como una unidad SSD híbrida con unidad de disco duro, lo que garantiza tiempos de respuesta incluso más rápidos.	599€	Modify Compare Delete
6	MSI GF75 Thin 9SC-277XES Intel Core i7-9750H/16GB/512GB SSD/GTX1650/17.3"	Te presentamos el portátil GF75 Thin de MSI, un portátil gaming con procesador Intel Core i7, 16GB de RAM, 512GB de SSD y gráfica Nvidia GeForce GTX 1650.	999€	Modify Compare Delete
7	HP Notebook 255 G7 AMD A4-9125/4GB/1TB/15.6"	Afronta todas tus tareas diarias con un portátil asequible que viene equipado con todas las características que necesitas. Obtén toda la potencia que deseas con el portátil HP Notebook 255 G7.	269€	Modify Compare Delete
8	Lenovo Ideapad 330S-15AST AMD A9-9425/4GB/128GB SSD/15.6"	Conoce la Ideapad 330s. Más delgada y más liviana con bisel estrechos para una visualización más amplia, cuenta con procesadores de hasta última generación y un rendimiento acelerado de gráficos. Impulsa cualquier tarea con facilidad. El lector de huellas digitales no está incluido.	399€	Modify Compare Delete
9	Asus Rog Strix G531GT-BQ165 Intel Core i7-9750H/16GB/512GB SSD/GTX1650/15.6"	La victoria se reduce a fracciones de segundo. Hasta con 144 Hz de refresco, esta formidable pantalla ultrafina te permite atrapar cada momento de la batalla. El refresco rápido utiliza por completo el rendimiento de la GPU y ofrece una experiencia de juego fluida a altos FPS.	899€	Modify Compare Delete

El botón de Modificar abre una modal relleno los datos y tienes la opción de guardar o cerrar sin guardar.

Al añadir se abre el mismo modal pero sin relleno los datos.



The image shows a modal window titled "Edit product" with a close button (X) in the top right corner. The form contains the following fields:

- Id:** A text input field containing the number "4".
- Title:** A text input field containing "Asus VivoBook D509DA-EJ098 AMD Ryzen 7 3700U/8GB/512GB S".
- Description:** A text input field containing "Te presentamos el D509DA de Asus, un portátil con procesador AI".
- Price:** A text input field containing "499".

At the bottom right of the form are two buttons: "Save" and "Close".

























El comparador filtra los precios de tu producto seleccionado y todos los de la competencia.

Los productos de la competencia con un **menor precio** tienen tanto un marco rojo como el precio en rojo avisando que puede que quieras corregir eso.

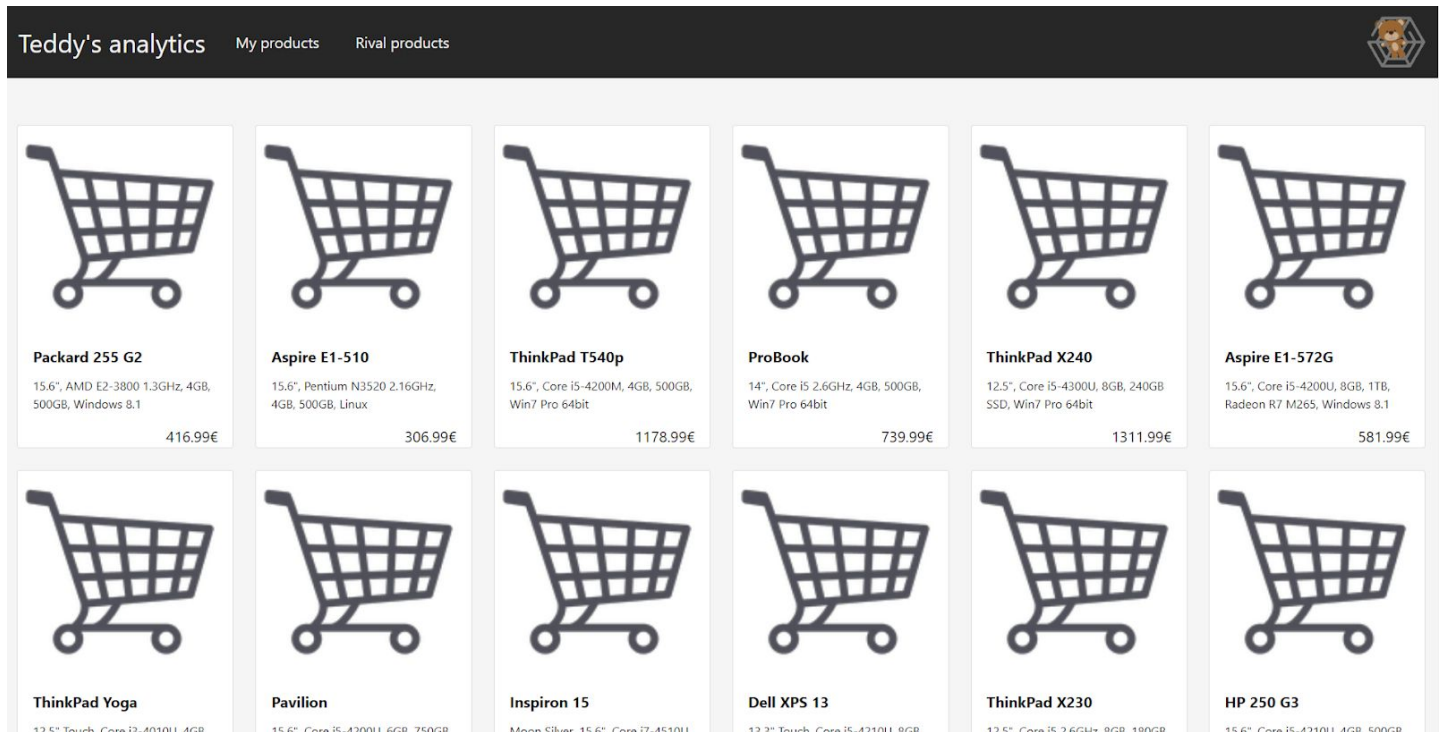
Los productos de la competencia con un **mayor precio** tienen tanto un marco verde como el precio en verde avisando que tu precio es el más barato de la competencia.

No hay competencias directas sino compara un producto con todos porque hay demasiadas especificaciones para cada artículo y requeriría una Base de Datos demasiado grande



 <p>ThinkPad T540p 15.6", Core i5-4200M, 4GB, 500GB, Win7 Pro 64bit 1178.99€</p>	 <p>ProBook 14", Core i5 2.6GHz, 4GB, 500GB, Win7 Pro 64bit 739.99€</p>	 <p>ThinkPad X240 12.5", Core i5-4300U, 8GB, 240GB SSD, Win7 Pro 64bit 1311.99€</p>	 <p>Aspire E1-572G 15.6", Core i5-4200U, 8GB, 1TB, Radeon R7 M265, Windows 8.1 581.99€</p>	 <p>ThinkPad Yoga 12.5" Touch, Core i3-4010U, 4GB, 500GB + 16GB SSD Cache, 1033.99€</p>	 <p>Pavilion 15.6", Core i5-4200U, 6GB, 750GB, Windows 8.1 609.99€</p>
 <p>Inspiron 15 Moon Silver, 15.6", Core i7-4510U 1362.24€</p>	 <p>Dell XPS 13 13.3" Touch, Core i5-4210U, 8GB 1366.32€</p>	 <p>ThinkPad X230 12.5", Core i5 2.6GHz, 8GB, 180GB 1381.13€</p>	 <p>HP 250 G3 15.6", Core i5-4210U, 4GB, 500GB 1362.24€</p>	 <p>ThinkPad Yoga 12.5" Touch, Core i5 4200U, 8GB 1366.32€</p>	 <p>HP 350 G1 15.6", Core i5-4200U, 4GB, 750GB 1381.13€</p>
 <p>Asus ROG Strix GL702VM-GC146T Asus ROG Strix GL702VM-GC146T, 17.3" FHD, Core i7-7700HQ, 8GB, 1TB + 128GB SSD, GeForce GTX 1060 3GB, Windows 10 Home, Eng kbd 1399€</p>	 <p>Packard 255 G2 15.6", AMD E2-3800 1.3GHz, 4GB, 500GB, Windows 8.1 416.99€</p>	 <p>Aspire E1-510 15.6", Pentium N3520 2.16GHz, 4GB, 500GB, Linux 306.99€</p>	 <p>Asus VivoBook Max Asus VivoBook Max X541NA-GQ041 Black Chocolate, 15.6" HD, Pentium N4200 1.1GHz, 4GB, 500GB, Windows 10 Home 399€</p>	 <p>Dell Vostro 15 Dell Vostro 15 (3568) Black, 15.6" FHD, Core i5-7200U, 4GB, 128GB SSD, Radeon R5 M420 2GB, Linux 488.78€</p>	 <p>Asus VivoBook X441NA-GA190 Asus VivoBook X441NA-GA190 Chocolate Black, 14", Celeron N3450, 4GB, 128GB SSD, Endless OS, ENG kbd 295.99€</p>
 <p>Prestigio SmartBook 133S</p>	 <p>Prestigio SmartBook 133S</p>	 <p>Lenovo V110-15IAP</p>	 <p>Lenovo V110-15IAP</p>	 <p>Hewlett Packard 250 G6</p>	 <p>Acer Aspire 3 A315-31</p>

La parte de los productos de la competencia es lo mismo pero sin ningún tipo de filtro ni orden:



En cualquier producto de la competencia le puedes clicar y te lleva a sus características.

Y dentro del producto si haces clic en el título del producto te lleva al artículo original.

(Todas las imágenes son las mismas por la propia página de prueba)



Aspire E1-510

15.6", Pentium N3520 2.16GHz, 4GB, 500GB, Linux

Id	517
Swatch	128 256 512 1024
Ratings	2 reviews

306.99€

5 Valoración personal

Este proyecto me ha servido para aprender sobre las tecnologías de Java y Angular, aunque Angular había hecho un poco en clase he notado mejoría ya que he experimentado más técnicas y métodos más avanzados de implementar mis ideas, sobre todo porque he podido juntar Angular con bootstrap y eso me ha interesado mucho, me ha parecido mucho más fácil crear el diseño de la web..

Me ha gustado el proyecto ya que considero que la parte “fuerte” de la web es el scrapeado con fines educativos, el cómo se manejan los datos y como evitar ciertas medidas de protección anti-scraping.

Un problema es que al no hay mucha información al respecto ya que el scraping es una técnica de la que no hay una gran fuente de información, pero lo bueno es que he podido generar mis propios métodos de hacer las cosas y trabajar con el ingenio.

También quiero destacar la organización que he tenido en este proyecto que creo que ha sido una de las mejores que he tenido, el método SCRUM qué hicimos en clase sirvió para aprender como usarlo pero en clase no le veía utilidad ya que si había buena comunicación no nos hacía falta, pero la realidad es que me ha sorprendido su utilidad y doy gracias al instituto por enseñar este tipo de mecánicas de trabajo.

Del scraping lo que he aprendido es que es demasiado peligroso ya que en cuestión de horas podrías obtener todos los datos en masa de cualquier página, incluso los personales ya que por mucho que se intente

evitar siempre hay una manera de saltar las defensas y obtener los datos, exceptuando los login, el sistema de inicio de sesión detiene cualquier scrapeo.

Finalizando con el tema del scraping creo que es una técnica que está bien aprender porque saber de muchas cosas es bueno pero qué es una técnica que puede molestar mucho a las páginas a las que scraperas así que en mi opinión debería hacerse solo para estudiarlo y aprenderlo.

Bibliografía

Documentación de bootstrap

Mark Otto, a. (2020). Bootstrap. <https://getbootstrap.com/>

Documentación Angular

DevDocs — Angular documentation. <https://devdocs.io/angular/>

Documentación Java

Baeldung. 2020. *Java, Spring And Web Development Tutorials*.: <https://www.baeldung.com/>

Diversas aportaciones

Stack Overflow. 2020. *Stack Overflow - Where Developers Learn, Share, & Build Careers*.: <https://stackoverflow.com/>



Scrap help

Webscraper.io. 2020. *Web Scraper - The #1 Web Scraping Extension*. <https://webscraper.io/>

Github projects

GitHub. 2020. *Build Software Better, Together*. <https://github.com/>

NPM

Npmjs.com. 2020. *Npm | Build Amazing Things*. <https://www.npmjs.com/>