BOYS

Ricardo Arturo Landinez Porras
Politécnico internacional
ricardo.landinez@pi.edu.co

Resumen — *Este desarrollo MVC, tiene como objetivo la visualización de una de las series más polémicas, irónicas y crudas de la fantasía humana, la cual intenta deconstruir el mito de los superhéroes, donde nos enseñan un mundo distópico, en el que estos existen y luchan contra la injusticia humana, más sin embargo todo es una fachada, donde nos muestran que a la final estos continúan siendo humanos, pero de la forma más aberrante, con temas de supremacía blanca, acosadores sexuales y hasta fanatismo religioso. A este punto, trataremos de mostrar de forma interactiva, los datos y mensaje subliminales, de cada una de las escenas y personajes más cuestionables en la serie, veremos comentarios y escenas del detrás de cámaras, tanto de nuestros actores favoritos, como lo de sus productores, donde tratan de explicarnos el detrás de toda esta cinematográfica más aclamada en la última década.*

Palabras claves:

Abstract *— This informatic project aims to visualize one of the most controversial, ironic, and raw series of human fantasy, which tries to deconstruct the myth of superheroes, where they teach us a dystopian world, where they exist and fight against human injustice. However, everything is a fake, this series shows us that at the end these people continue to be human, but in the most aberrant way, with airs of white supremacy, sexual harassers, and even religious fanaticism. At this point, we will try to interactively display, subliminal data and message of each of the most questionable scenes and characters in the series, we will see comments and scenes from behind the scenes, both from our favorite actors as well as its producers, where they try to explain the behind all this most acclaimed cinema in the last decade.*

Keywords:

I. INTRODUCCIÓN

Este proyecto consiste en la visualización de la serie The boys en un sistema modelo cliente-servidor, el cual se basara principalmente en los datos más iconitos de la serie, todo este proceso será programado a través Java (Netbeans IDE 14),en el cual se almacenara la data del cliente en un servicio de MySql, donde se insertara los accesos como los usuarios, el administrador e invitado, con lo cual al ingresar tendrá ciertos accesos dependiendo su rol dentro de la misma, podrá consultar datos relevantes de la serie, configurar datos de su perfil, también incluirá un pequeño juego para los usuarios registrados. Lo anteriormente expuesto, se visualizar
de una forma didáctica y sencilla para el usuario.

DEFINICIONES

Para este Proyecto se requiere lo siguiente:

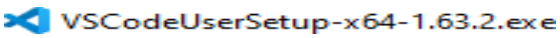
Aplicativo	Versiona miento	Requerimie ntosmínimos
Visual Studio Code	1.71	Windo ws, 8.9.10,3 2bit vession
MySQL Community Server	8.0.30	Windows (x86, 64-bit),
Payara		Oracle JDK8
Docker	Ink	Windows

Proceso de instalación Instalación VISUAL STUDIO CODE

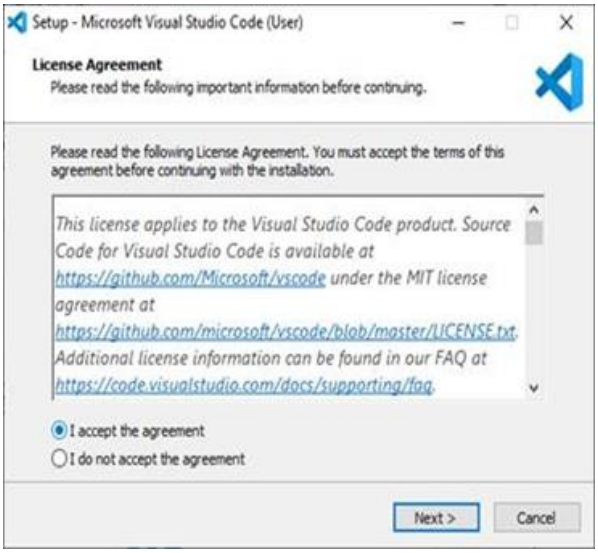
Figura (1)

Paso 1: Ve a la página de Microsoft Visual Studio Code en Academic Software y haz clic en el botón 'Descargar Visual Studio Code' para descargar el archivo de instalación.

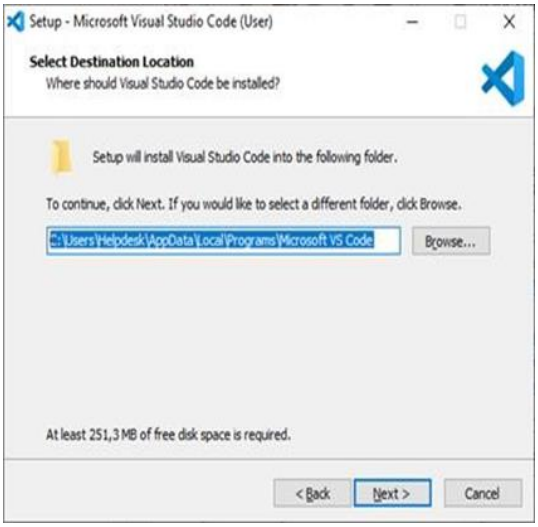
Paso 2: Abre el archivo de instalación .exe en tu carpeta de descargas para iniciar la instalación.



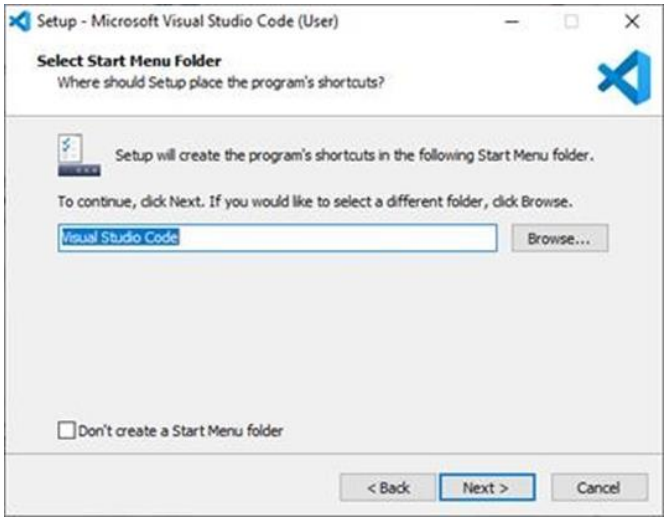
Paso 3: Lee y acepta el acuerdo de licencia. Haz clic en Next para continuar.



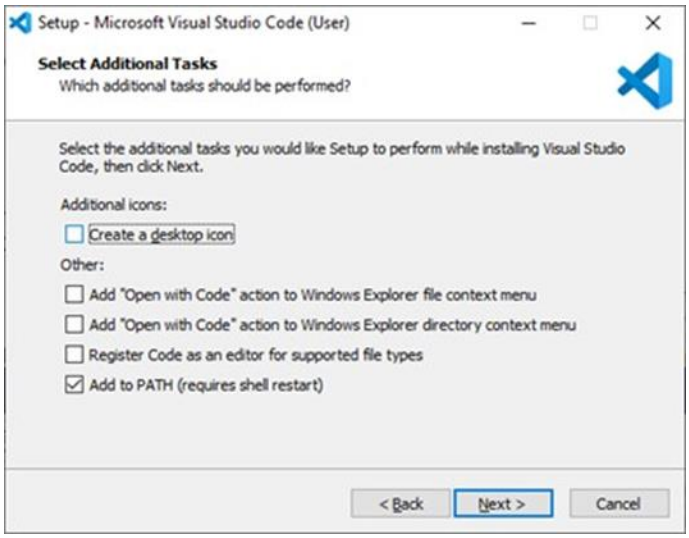
Paso 4: Puedes cambiar la ubicación de la carpeta de instalación o mantener la configuración predeterminada. Haz clic en Next para continuar.



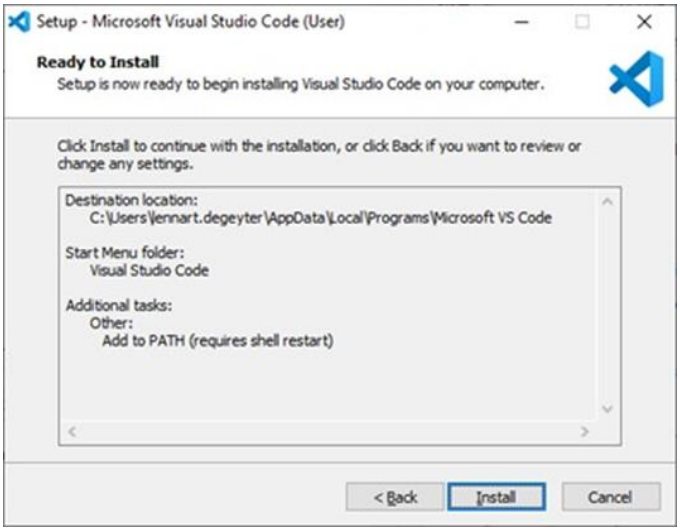
Paso 5: Elige si deseas cambiar el nombre de la carpeta de accesos directos en el menú Inicio o si no deseas instalar accesos directos en absoluto. Haz clic en Next.



Paso 6: Selecciona las tareas adicionales, por ej. crear un icono en el escritorio o añadir opciones al menú contextual de Windows Explorer. Haz click en next.



Paso 7: Haz clic en Install para iniciar la instalación.

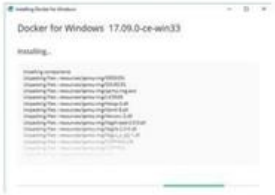


Paso 8: El programa está instalado y listo para usar. Haz clic en Finish para finalizar la instalación y lanzar el programa.

Instalando Docker

El proceso de instalación de Docker en Windows 10, es bastante simple, descargarlo desde aquí: Descargar Docker

Una vez descargado, ejecutamos el archivo y se lanzará el proceso de instalación como vemos en la imagen:



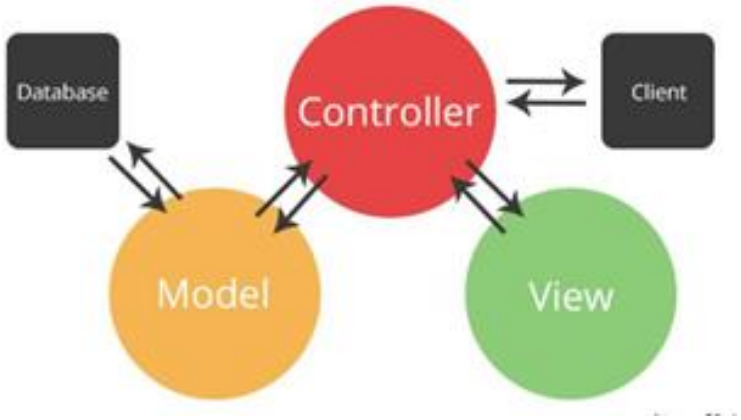
Instalar Docker

Una vez descargado, ejecutamos el archivo y se lanzará el proceso de instalación como vemos en la imagen:

Cuando finalice el proceso, aparecerá la ventana de instalación correcta, pulsaremos el botón de "Close and logout" para terminar de configurar Docker en nuestro Windows 10. En algunos casos, esto provocará que se reinicie el sistema Windows para terminar la configuración:

Después de reiniciar nuestro Windows 10 ya tendremos Docker instalado en nuestro PC.

Diagrama del proyecto MVC.



Clases (Actor hasta el momento).

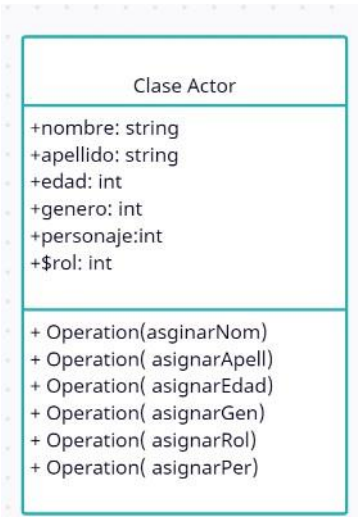
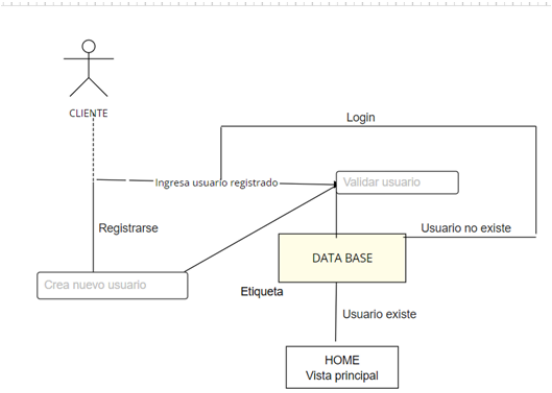


Diagrama de secuencia.



Diccionario de datos:

1. Registrar: sección de registro para nuevos usuarios, en el cual, a través de un usuario y contraseña, se creará en una base de datos un nuevo usuario.
2. Login: luego del registro se ingresará el login el cual ingresaran los datos pertinentes y si estos existen entraran al home de la página, donde se podrá visualizar una vista de información frente a ciertos actores.
3. Data Base : información donde se encontrara los datos registrados de cada usuario que se logea.
4.Home : Pagina principal donde se encontrara información acerca de la serie escogida.

Code.

Docker: dentro de la carpeta Docker del proyecto encontraremos la imagen que se monta para tener acceso a nginx y se construye una nueva imagen donde quedara alojada la página.

```yaml
version: '3'
services:
  web:
    image: nginx
    ports:
      - "8084:80"
    volumes:
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
      - ../html:/var/www/html
    networks:
      - web
  app:
    build: .
    image: boyz-app
    working_dir: /var/www/html
    volumes:
      - ../html:/var/www/html
    networks:
      - web
networks:
  web:
```

Carpeta Assets

Encontraremos en el archivo css los estilos frente a los inputs y button de la página, los cuales son replicados en las vistas de login,home y register, se exportan cierto tipo de fonts para ciertas etiquetas, como <p/>, <h1/>, etc.

```css
.form-wrapper {
  width: 100%;
  height: 100vh;
  background: url("../img/fondo2.jpg") no-repeat center center/cover;
  display: flex;
  justify-content: center;
  align-items: center;
}
#home {
  background: url("../img/home.jpg") no-repeat center center/cover;

}

hr {
    width: 100%;
    height: .5px;
    background: #eee;
    border: none;
}

.form-wrapper form {
  width: 400px;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  padding: 50px;
  background: #333;
  border-radius: 3px;
}
```

```css
}


.home-appbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 50px;
    background: #333;

}

.home-appbar .menu {
    display: flex;
    justify-content: space-between;
    align-items: center;

}

.home-appbar .menu a {
    color: #fff;
    font-size: 1.2rem;
    font-weight: 600;
    text-decoration: none;
    font-family: "Roboto", sans-serif;
    display: block;
    padding: 10px 20px;
    margin: 0 5px;
}
```

En la carpeta js, encontramos los enrutamientos dependiendo si genera acceso o no a la plataforma y sus respectivas validaciones

```
import request from "./requests.js";

const formLogin = document.querySelector("#form-login");
formLogin.addEventListener("submit", (e) => {
  e.preventDefault();

  const { usuario, clave } = e.target;

  request("../../scripts/login.php", "POST", {
    usuario: usuario.value,
    clave: clave.value,
  })
    .then((res) => {
      console.log(res);
      location.href = "../../home.php";
    })
    .catch((err) => alert(err.error));
});

```

Vista del home en html.el cual contiene el sesión start , cuando se conecta correctamente con el usuario registrado en base de datos.

```
<?php
session_start();

if (!isset($_SESSION['user'])) {
    header('Location: index.php');
    exit;
}
?>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./assets/css/styles.css">
    <title>Home</title>
</head>

<body>
    <div class="home-appbar">
        <h2>Bienvenido, <?php echo $_SESSION['user']['name']; ?></h2>
        <div class="menu">
            <a href="#">Sipnosis</a>
            <a href="#">Personajes</a>
            <a href="scripts/logout.php">Salir</a>
        </div>
    </div>
</body>
```

En el register, encontramos las validaciones en javascript de cómo se recibe el dato y la comparación que hay de una contraseña nueva con la confirmación de esta.

En la carpeta Resources, encontramos la conexión a base de datos.

Index, donde generamos la información del login, donde encontraremos los inputs para ingresar y la opción del botón de regístrate.

Registrar html, encontraremos las etiquetas para generar la vista.

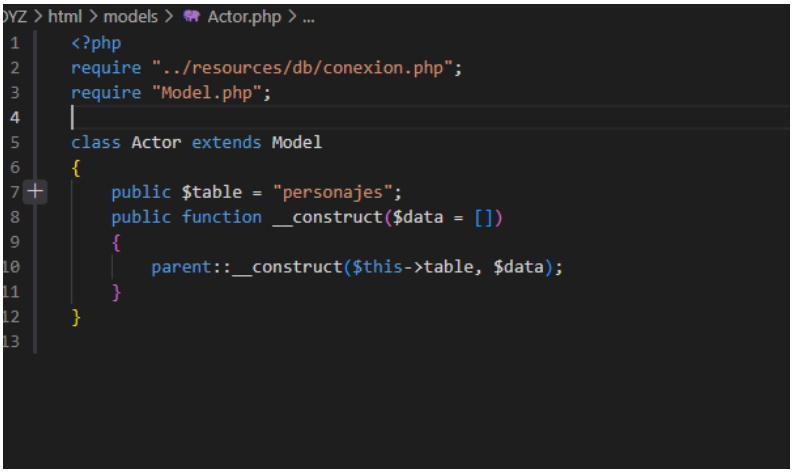
```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./assets/css/styles.css">
    <title>Registrarme</title>
</head>

<body>
    <div class="form-wrapper">
        <form id="form-register">
            <h1>Registrarme</h1>
            <hr>
            <input type="text" name="usuario" placeholder="Usuario" required>
            <input type="text" name="nombre" placeholder="Nombre" required>
            <input type="password" name="clave" placeholder="Clave" required>
            <input type="password" name="clave2" placeholder="Repetir Clave" required>
            <div class="spacer"></div>
            <button>Registrarme</button>
            <p>¿Ya tienes una cuenta? <a href="index.php">Inicia sesión</a></p>
        </form>
    </div>
    <script src="assets/js/register.js" type="module"></script>
</body>

</html>
```

Una de las clses aplicadas fue la clase actor, la cual directamente se traer por una clase abstrac model.



```php
<?php
require "../resources/db/conexion.php";
require "Model.php";

class Actor extends Model
{
    public $table = "personajes";
    public function __construct($data = [])
    {
        parent::__construct($this->table, $data);
    }
}
```

```php
<?php

require "../resources/db/conexion.php";


abstract class Model {
    private $keys = [];
    private $values = [];
    private $data;


    public function __construct($table, $data)
    {
        $this->table = $table;
        $this->data = $data;
        if (count($data) > 0) {
            $this->keys = array_keys($data);
            $this->values = array_values($data);
        }
    }


    public function get_all() {
        global $mysqli;
        $sql = "SELECT * FROM {$this->table}";
        $result = $mysqli->query($sql);
```

En el siguiente print vemos la imagen de Docker que se usó para levantar los servicios de php,phpmyadmin,mysql Y el nginx. Este archivo se encontrara en la carpeta de Docker/Docker-compose

```yaml
app:
  build: .
  image: boyz-app
  working_dir: /var/www/html
  volumes:
    - ../html:/var/www/html
  environment:
    DB_HOST: db
    DB_USER: $DB_USER
    DB_PASSWORD: $DB_PASSWORD
    DB_DATABASE: $DB_NAME
  networks:
    - web
db:
  image: mysql
  ports:
    - "3306:3306"
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: $DB_NAME
    MYSQL_USER: $DB_USER
    MYSQL_PASSWORD: $DB_PASSWORD
  volumes:
    - ./mysql/data:/var/lib/mysql
    - ./mysql/init:/docker-entrypoint-initdb.d
```

Esta carpeta de Docker, también contendrá los tablas de BD que se generan dependiendo el proceso que se realice en la app, ejemplo insertar nuevos personajes.

```
--

DROP TABLE IF EXISTS `personajes`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `personajes` (
  `id` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(45) NOT NULL,
  `apellido` varchar(45) NOT NULL,
  `edad` int NOT NULL,
  `genero` varchar(45) NOT NULL,
  `temporada` varchar(45) NOT NULL,
  `personaje` varchar(45) NOT NULL,
  `rol` varchar(45) NOT NULL,
  `imagen` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `personajes`
--

--

LOCK TABLES `personajes` WRITE;
/*!40000 ALTER TABLE `personajes` DISABLE KEYS */;
INSERT INTO `personajes` VALUES (1,'Anthony','Stark',46,'Masculino','1,2,3','Homelander','Antagonista','homelanc
/*!40000 ALTER TABLE `personajes` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `users`
--

DROP TABLE IF EXISTS `users`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `usuario` varchar(45) NOT NULL,
  `clave` varchar(255) NOT NULL,
  `nombre` varchar(45) NOT NULL,
  `admin` tinyint(1) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
```

Aquí podemos visualizar en la carpeta de scripts, se indica que serán métodos post y get para generar la creación de los nuevos personajes en la sección del administrador.

```php
<?php

require "../resources/db/conexion.php";
require "../resources/response.php";
require "../resources/functions.php";
require "../models/Actor.php";

allowMethods(["POST", "GET"]);

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $data = getData();
    $actor = new Actor($data);

    $found = $actor->getByField("personaje");
    if ($found) {
        sendResponse("El personaje ya existe!", 400);
    }

    $result = $actor->create();
    if (!$result) {
        sendResponse("Error al crear el personaje: $mysqli->error", 400);
    }

    sendResponse("Personaje creado correctamente!", 200);
}
```

Scrip y sentencia de búsqueda para traer los personaje que seleccionemos en el select de la app.

```php
<?php

require "../resources/db/conexion.php";
require "../resources/response.php";
require "../resources/functions.php";

allowMethods();
$data=getData();

$personaje=$data["personaje"];

$result=$mysqli->query("SELECT * FROM personajes WHERE personaje='$personaje'");
$perso=$result->fetch_assoc();

if(!$perso) sendResponse('Personaje no encontrado', 404);

// $actor = new Actor();
// $actor->asignarNombre($user["n"])

sendResponse($perso, 200);
```