

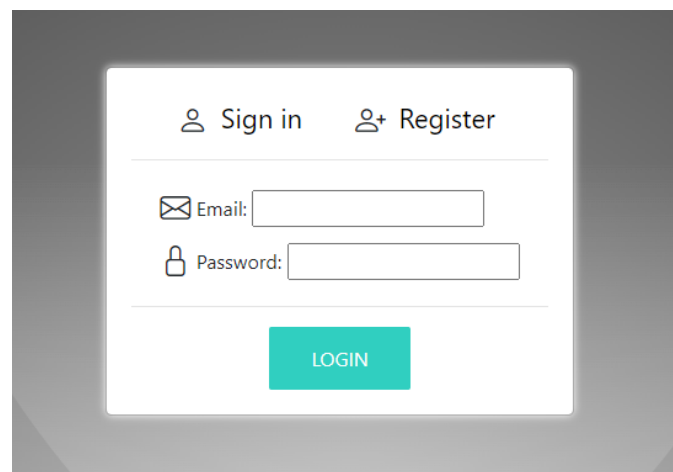
# Ibex 35 – Cotizaciones de la bolsa

## Descripción del reto

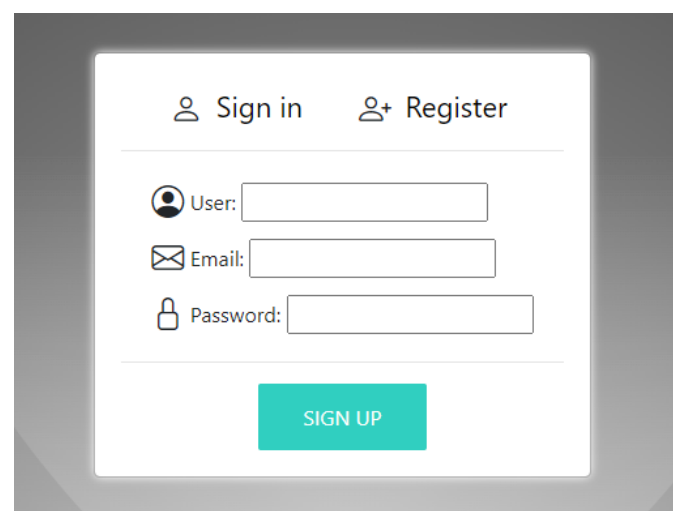
- El objetivo de este reto es mostrar la información de las cotizaciones de la bolsa (IBEX35) relativas a 10 de las empresas más importantes (INDITEX, SANTANDER, BBVA, NATURGY, CELLNEX, CAIXABANK, TELEFÓNICA, REPSOL, FERROVIAL, IBERDROLA).

## Desarrollo Web Servidor

- La primera vez que el usuario acceda a la página, le aparecerá un login en el que podrá iniciar sesión si ya está registrado o de lo contrario registrarse. Los datos introducidos por el usuario se guardarán en la base de datos. Además, el acceso es restringido sino se es usuario, por lo que una vez que el usuario haga login se generará un token único de acceso.



UI mockup of a login form. At the top, there are two links: 'Sign in' with a user icon and 'Register' with a user icon and a plus sign. Below these links are two input fields: 'Email:' with an envelope icon and 'Password:' with a lock icon. At the bottom, there is a teal button labeled 'LOGIN'.



UI mockup of a registration form. At the top, there are two links: 'Sign in' with a user icon and 'Register' with a user icon and a plus sign. Below these links are three input fields: 'User:' with a user icon, 'Email:' with an envelope icon, and 'Password:' with a lock icon. At the bottom, there is a teal button labeled 'SIGN UP'.

- Para poder hacer el login, se ha implementado el “*Laravel JWT Authentication*” que nos permite crear un método de autenticación en servicios API de forma sencilla, al enviar un usuario y una contraseña la API retorna un token para que esta compruebe que tiene acceso a las acciones que se quieran realizar.
- Lo primero que se ha hecho es crear un proyecto de laravel y crear una base de datos. Para poder conectarse a dicha base de datos se configura el archivo .env:

```
DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=stocks
DB_USERNAME=stocks
DB_PASSWORD=stocks
```

- La tabla de user viene preinstalada en laravel por lo que ejecutando el siguiente comando se creará la tabla de user:

```
php artisan migrate
```

- A partir de ahí se necesitaba un controlador que tuviera las funciones necesarias para que funcionase el login al completo:

```
public function login(Request $request)
{
    $request->validate([
        'email' => 'required|string|email',
        'password' => 'required|string',
    ]);
    $credentials = $request->only('email', 'password');

    $token = Auth::attempt($credentials);
    if (!$token) {
        return response()->json([
            'status' => 'error',
            'message' => 'Unauthorized',
        ], 401);
    }

    $user = Auth::user();
    return response()->json([
        'status' => 'success',
        'user' => $user,
        'authorisation' => [
            'token' => $token,
            'type' => 'bearer',
        ]
    ]);
}
```

```

public function register(Request $request){
    $request->validate([
        'name' => 'required|string|max:255',
        'email' => 'required|string|email|max:255|unique:users',
        'password' => 'required|string|min:6',
    ]);

    $user = User::create([
        'name' => $request->name,
        'email' => $request->email,
        'password' => Hash::make($request->password),
    ]);

    $token = Auth::login($user);
    return response()->json([
        'status' => 'success',
        'message' => 'User created successfully',
        'user' => $user,
        'authorisation' => [
            'token' => $token,
            'type' => 'bearer',
        ]
    ]);
}

```

```

public function logout()
{
    Auth::logout();
    return response()->json([
        'status' => 'success',
        'message' => 'Successfully logged out',
    ]);
}

```

```

public function refresh()
{
    return response()->json([
        'status' => 'success',
        'user' => Auth::user(),
        'authorisation' => [
            'token' => Auth::refresh(),
            'type' => 'bearer',
        ]
    ]);
}

```

- Además, para que el sistema de login funcionara correctamente, se ha creado un middleware *"auth:api"* que se encarga de autenticar a un usuario antes de permitirle acceder a una ruta predeterminada. En este caso, se está especificando que la autenticación es requerida para todas las rutas excepto para las acciones *"login"* y *"register"*.

```
public function __construct()
{
    $this->middleware('auth:api', ['except' => ['login','register']]);
}
```

- Finalmente, se llama al controlador en las rutas y se le pasan las funciones que se han creado previamente en el AuthController.

```
Route::controller(AuthController::class)->group(function () {
    Route::post('login', 'login');
    Route::post('register', 'register');
    Route::post('logout', 'logout');
    Route::post('refresh', 'refresh');
});
```

## Desarrollo Web Cliente

- Una vez de haberse logueado, el usuario dispone de acceso para empezar a utilizar la aplicación. La página principal está dividida de la siguiente manera:
  - Un botón de logout en la parte superior derecha.
  - El título en la parte central de la página.
  - Una tabla con todas las empresas.
  - Un contenedor donde se van a arrastrar las empresas que se deseen mediante drag & drop.
  - Un botón de guardar que guarda el id de la empresa seleccionada en el localStorage y la muestra en una tarjeta de bootstrap junto a su cotización.
  - Finalmente hay un icono de una gráfica que despliega un modal donde se puede seleccionar la empresa de la cual se quiere ver la gráfica y su periodicidad.



# IBEX35

INDITEX	Santander	BBVA	Naturgy	cellnex
CaixaBank	Telefónica	REPSOL	ferrovial	IBERDROLA

Guardar



- Para gestionar el panel de las empresas a mostrar se ha utilizado la funcionalidad de “Drag & Drop”. Básicamente se arrastran las empresas que se quieran ver en el contenedor y se clicka en el botón de guardar. En este momento se guardan los ids de las imágenes que están en el contenedor en localStorage para a continuación mostrar dichas empresas junto a su última cotización.

## IBEX35

BBVA

101.513



## IBEX35

BBVA

98.1521



```

function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
    ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev) {
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
    $(document).ready(function () {
        $("img").css("pointer-events", "all");
    })
}

function safe(empresa) {
    var nombre = empresa;
    $(document).ready(function () {
        $("img").css("pointer-events", "none");
        $("# " + nombre).css("pointer-events", "all");
    })
}

function empresas() {
    var imagenes = document.getElementById("caja").getElementsByTagName("img");
    var idsImagenes = [];
    for (var i = 0; i < imagenes.length; i++) {
        idsImagenes.push(imagenes[i].id);
    }
    localStorage.setItem("idsImagenes", JSON.stringify(idsImagenes));
    cartas();
}

```

- Para poder mostrar la última cotización lo que se ha hecho es crear una función que obtenga los ids de las empresas que se vayan a mostrar. Se hace un fetch para hacer una llamada a la API del backend y así obtener la última cotización generada previamente y compararla para saber si es mayor o menor y así gestionar el color de dicha cotización. Los valores de las últimas cotizaciones se van insertando cada minuto en la base de datos con un script.

```
//Funcion de crear las cartas con el fetch de la base de datos del docker
function cartas() {
  var lastValue;
  $(".card").remove();
  var misEmpresas = JSON.parse(localStorage.getItem("idsImagenes"));

  // Fetch para conseguir el ultimo dato de la base de datos
  var getOptions = {
    method: "POST",
    redirect: "follow",
  };

  //Parte estatica de la carta:
  fetch("//HZ114487:8000/api/stocks", getOptions)
    .then((response) => response.json())
    .then((result) => {
      // Comparar cada nombre de las empresas con los nombres del localStorage
      result.data.forEach(function (element) {
        misEmpresas.forEach(function (nombre) {
          //En caso de que coincida el nombre se imprimira su imagen, nombre y valor
          if (element.nombre === nombre) {
            lastValue = element.valor;
            var cartahtml = `<div class="card" style="width: 300px; border: 2px solid black;">
              <img src=Imagenes/${element.nombre}.png class="card-img-top" style="padding: 25px;"><br>
              <div id="price${element.nombre}" class="card-body" style="display: flex; justify-content: center;">
                <h2>${element.valor}</h2><br>
              </div>
            </div>`;
            var card = $(cartahtml);
            $(".name~='info']").css("margin-left", "10px");
            $(".card-columns").append(card);
          }
        });
      });
    })
    .catch((error) => {
      console.log("error", error);
    });
};
```

```
setInterval(() => {
  var getOptions = {
    method: "POST",
    redirect: "follow",
  };

  fetch("//HZ114487:8000/api/stocks", getOptions)
    .then((response) => response.json())
    .then((result) => {
      result.data.forEach(function (element) {
        misEmpresas.forEach(function (nombre) {
          if (element.nombre === nombre) {
            if (lastValue > element.valor) {
              document.getElementById("price" + element.nombre).innerHTML =
                "<h2 style='color: red;'>" + element.valor + "</h2>";
              setTimeout(function () {
                document.getElementById("price" + element.nombre).innerHTML =
                  "<h2>" + element.valor + "</h2>";
              }, 3000);
            } else if (lastValue < element.valor) {
              document.getElementById("price" + element.nombre).innerHTML =
                "<h2 style='color: green;'>" + element.valor + "</h2>";
              setTimeout(function () {
                document.getElementById("price" + element.nombre).innerHTML =
                  "<h2>" + element.valor + "</h2>";
              }, 3000);
            }
            lastValue = element.valor;
          }
        });
      });
    })
    .catch((error) => {
      console.log("error", error);
    });
}, 60 * 100);
```

- Además, mediante fetch también se hacen las llamadas a la API del backend para el register, login y logout.

```
function registercheck() {
  const nombre = document.getElementById("user").value;
  const email = document.getElementById("email-register").value;
  const contraseña = document.getElementById("contraseña-register").value;

  var formdata = new FormData();
  formdata.append("name", nombre);
  formdata.append("email", email);
  formdata.append("password", contraseña);

  var requestOptions = {
    method: "POST",
    body: formdata,
    redirect: "follow",
  };

  fetch("//HZ114487:8000/api/register", requestOptions)
    .then((response) => response.text())
    .then((result) => {
      console.log(result)
      const mensaje = document.getElementById("message-register");
      mensaje.innerHTML = "<p style='color: green;'>Registro Correcto</p>";
      setTimeout(function () {
        sign();
        document.getElementById("user").value = "";
        document.getElementById("email-register").value = "";
        document.getElementById("contraseña-register").value = "";
        mensaje.style.display = "none";
      }, 2000);
    })
    .catch((error) => {
      console.log("error", error);
      const mensaje = document.getElementById("message-register");
      mensaje.innerHTML = "<p style='color: red;'>Registro incorrecto</p>";
    });
}
```



```

function logoutcheck() {
  const token = localStorage.getItem("token");
  const email = localStorage.getItem("email");
  const password = localStorage.getItem("password");
  var myHeaders = new Headers();
  myHeaders.append("Authorization", 'Bearer ' + token);

  var formdata = new FormData();
  formdata.append("email", email);
  formdata.append("password", password);

  var requestOptions = {
    method: 'POST',
    headers: myHeaders,
    body: formdata,
    redirect: 'follow'
  };

  fetch("/HZ114487:8000/api/logout", requestOptions)
    .then(response => response.text())
    .then(result => {
      console.log(result);
      localStorage.removeItem("token");
      localStorage.removeItem("email");
      localStorage.removeItem("password");
      localStorage.removeItem("idsImagenes");
      $(".modal").css('display', 'block');
    })
    .catch(error => console.log('error', error));
}

```

```

function logincheck() {
  localStorage.removeItem("token");
  const email = document.getElementById("email-sign").value;
  const contraseña = document.getElementById("contraseña-sign").value;

  var formdata = new FormData();
  formdata.append("email", email);
  formdata.append("password", contraseña);

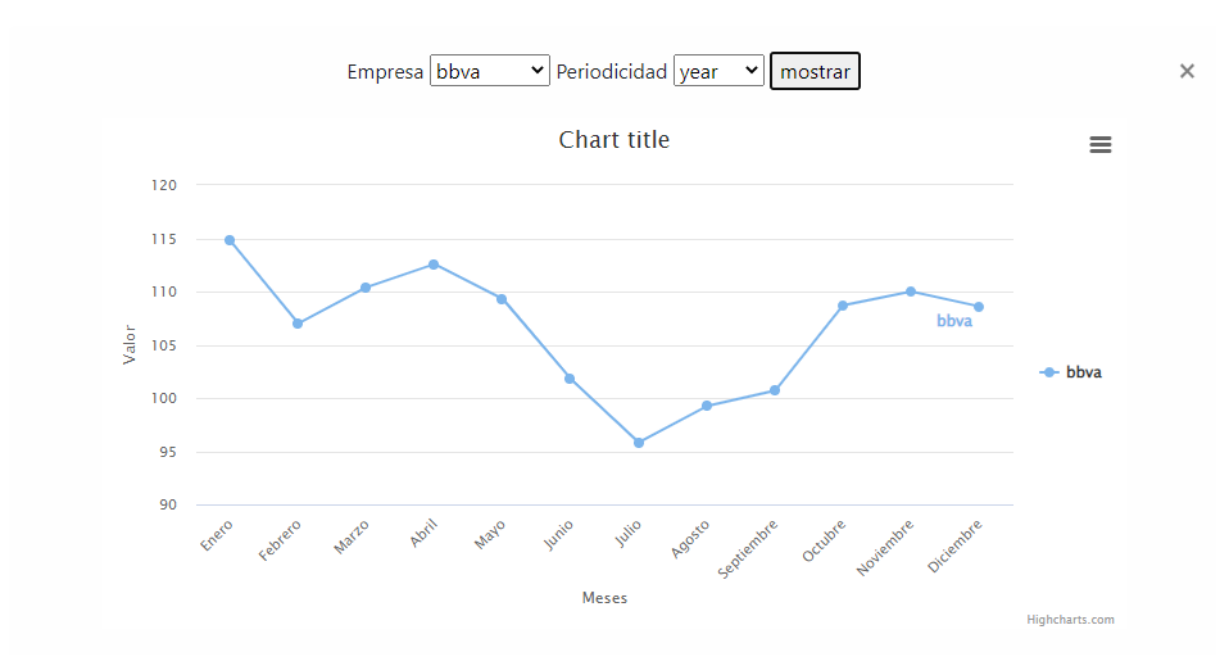
  var requestOptions = {
    method: 'POST',
    body: formdata,
    redirect: 'follow'
  };

  fetch("/HZ114487:8000/api/login", requestOptions)
    .then(response => response.json())
    .then(result => {
      console.log(result)
      localStorage.setItem("token", result['authorisation']['token']);
      const mensaje = document.getElementById("message-sign");
      mensaje.innerHTML = "<p style='color: green;'>Inicio de sesion correcto, entrando...</p>";
      setTimeout(function () {
        document.getElementById("email-sign").value = "";
        document.getElementById("contraseña-sign").value = "";
        mensaje.style.display = "none";
        modal.setAttribute('style', 'display:none !important')
      }, 2000);
    })
    .catch(error => {
      console.log('error', error)
      const mensaje = document.getElementById("message-sign");
      mensaje.innerHTML = "<p style='color: red;'>Email o password incorrectos</p>";
      setTimeout(function () {
        document.getElementById("email-sign").value = "";
        document.getElementById("contraseña-sign").value = "";
        mensaje.style.display = "none";
      }, 3000);
      mensaje.style.display = "flex";
    });
}

```

## Diseño Interfaces Web

- **Bootstrap**
- Los gráficos de las empresas están generados mediante la librería de JavaScript “*Highchart*”. Para poder ver los gráficos de las empresas primero hay que clicar en el icono del gráfico que está en la parte inferior de la página y se desplegará un “*modal*” en el que se podrá seleccionar la empresa de la cual se quiera ver el gráfico y su periodicidad.



- Para generar los datos de los gráficos se ha hecho una función que mediante un fetch hace las llamadas a la API del backend. El controlador de la API “*ChartController*” gestiona los gráficos mediante diferentes funciones, cada función tiene su query según la periodicidad que queramos consultar (anual, mensual y semanal).

```
let valores = [];  
var empresa;  
var periodicidad;  
  
function tabla() {  
  empresa = document.getElementById("emp").value;  
  periodicidad = document.getElementById("per").value;  
  
  if (periodicidad === "year") {  
    anual();  
  } else if (periodicidad === "month") {  
    mensual();  
  } else if (periodicidad === "week") {  
    semanal();  
  }  
}
```

```

function anual() {
  console.log("Anual");
  valores = [];
  var requestOptions = {
    method: 'POST',
    redirect: 'follow'
  };

  fetch("/HZ114487:8000/api/year", requestOptions)
    .then(response => response.json())
    .then(result => {
      console.log(result)
      result.data.forEach(function (element) {
        if (element.nombre === empresa) {
          console.log(element);
          valores.push(element.media_valor);
        }
      })
      Highcharts.chart('container-chart', {

        yAxis: {
          title: {
            text: 'Valor'
          }
        },
        xAxis: {
          categories: ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'],
          title: {
            text: 'Meses'
          }
        },
        legend: {
          layout: 'vertical',
          align: 'right',
          verticalAlign: 'middle'
        },
        plotOptions: {
          series: {
            label: {
              connectorAllowed: false
            },
          }
        }
      })
    })
}

```

```

      series: [{
        name: empresa,
        data: valores
      }],

      responsive: {
        rules: [{
          condition: {
            maxWidth: 500
          },
          chartOptions: {
            legend: {
              layout: 'horizontal',
              align: 'center',
              verticalAlign: 'bottom'
            }
          }
        }]
      }
    });
  });
})
.catch(error => console.log('error', error));
console.log(valores)
}

```

```

function mensual() {
  console.log("Mensual");
  valores = [];
  let fechas = [];
  var requestOptions = {
    method: 'POST',
    redirect: 'follow'
  };

  fetch("//HZ114487:8000/api/month", requestOptions)
    .then(response => response.json())
    .then(result => {
      console.log(result)
      result.data.forEach(function (element) {
        if (element.nombre === empresa) {
          console.log(element);
          valores.push(element.valor);
          fechas.push(element.fecha);
        }
      })
      Highcharts.chart('container-chart', {

        yAxis: {
          title: {
            text: 'Valor'
          }
        },
        xAxis: {
          categories: fechas,
          title: {
            text: 'dias del mes pasado'
          }
        },
        legend: {
          layout: 'vertical',
          align: 'right',
          verticalAlign: 'middle'
        },
        plotOptions: {
          series: {
            label: {
              connectorAllowed: false
            },
          }
        }
      },
    )
  }
}

```

```

      series: [{
        name: empresa,
        data: valores
      }],
      responsive: {
        rules: [{
          condition: {
            maxWidth: 500
          },
          chartOptions: {
            legend: {
              layout: 'horizontal',
              align: 'center',
              verticalAlign: 'bottom'
            }
          }
        }]
      }
    });
  });
  .catch(error => console.log('error', error));
  console.log(valores)
}

```

```

function semanal() {
  console.log("Semanal");
  valores = [];
  let fechas = [];
  var requestOptions = {
    method: 'POST',
    redirect: 'follow'
  };

  fetch("//HZ114487:8000/api/week", requestOptions)
    .then(response => response.json())
    .then(result => {
      console.log(result)
      result.data.forEach(function (element) {
        if (element.nombre === empresa) {
          console.log(element);
          valores.push(element.valor);
          fechas.push(element.fecha);
        }
      });
      Highcharts.chart('container-chart', {

        yAxis: {
          title: {
            text: 'Valor'
          }
        },
        xAxis: {
          categories: fechas,
          title: {
            text: 'dias del mes pasado'
          }
        },
        legend: {
          layout: 'vertical',
          align: 'right',
          verticalAlign: 'middle'
        },
        plotOptions: {
          series: {
            label: {
              connectorAllowed: false
            },
          }
        },

```

```

        series: [{
          name: empresa,
          data: valores
        }],

        responsive: {
          rules: [{
            condition: {
              maxWidth: 500
            },
            chartOptions: {
              legend: {
                layout: 'horizontal',
                align: 'center',
                verticalAlign: 'bottom'
              }
            }
          }]
        }
      });
    });
  })
  .catch(error => console.log('error', error));
  console.log(valores)
}

```

- Funciones del “ChartController” y rutas.

```
public function stockYear()
{
    $query = "SELECT t2.id, t2.nombre, AVG(t1.valor) as media_valor, DATE_FORMAT(t1.fecha, '%m') as mes
FROM stock t1
JOIN empresas t2 ON t2.id = t1.empresas_id
GROUP BY t1.empresas_id, mes
ORDER BY t1.empresas_id, mes;";
    $data = DB::select($query);

    // Return the data as a JSON response
    return response()->json([
        'data' => $data
    ]);
}
```

```
public function stockMonth()
{
    $query = "SELECT t2.id, t2.nombre, MAX(t1.valor) as valor, DATE(t1.fecha) as fecha
FROM stock t1
JOIN empresas t2 ON t2.id = t1.empresas_id
WHERE t1.fecha >= NOW() - INTERVAL 1 MONTH
GROUP BY t1.empresas_id, DATE(t1.fecha)
ORDER BY t1.empresas_id, fecha;";
    $data = DB::select($query);

    // Return the data as a JSON response
    return response()->json([
        'data' => $data
    ]);
}
```

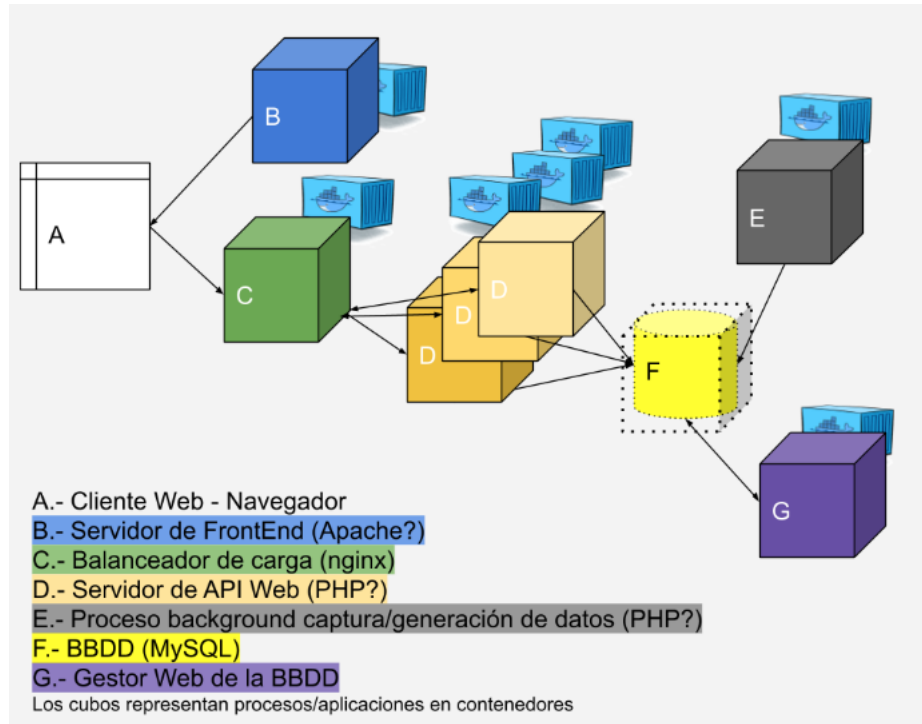
```
public function stockWeek()
{
    $query = "SELECT t2.id, t2.nombre, MAX(t1.valor) as valor, DATE(t1.fecha) as fecha
FROM stock t1
JOIN empresas t2 ON t2.id = t1.empresas_id
WHERE t1.fecha >= NOW() - INTERVAL 1 WEEK
GROUP BY t1.empresas_id, DATE(t1.fecha)
ORDER BY t1.empresas_id, fecha;";
    $data = DB::select($query);

    // Return the data as a JSON response
    return response()->json([
        'data' => $data
    ]);
}
```

```
Route::controller(ChartController::class)->group(function () {
    Route::post('week', 'stockWeek');
    Route::post('month', 'stockMonth');
    Route::post('year', 'stockYear');
});
```

## Despliegue Aplicaciones Web

- En cuanto a la contenerización del proyecto se ha seguido el siguiente esquema:



- Primero para poder levantar el proyecto contenerizado, hay que configurar el archivo "docker-compose.yml" indicando los servicios que se van a utilizar y dependiendo el servicio, la configuración que se requiera:

```
🐳 docker-compose.yml
1  version: "3.7"
2
3  services:
4      db:
5          image: mysql:5.7
6          container_name: db
7          environment:
8              MYSQL_ROOT_PASSWORD: stocks
9              MYSQL_DATABASE: stocks
10             MYSQL_USER: stocks
11             MYSQL_PASSWORD: stocks
12         ports:
13             - "3306:3306"
14         volumes:
15             - dbdata:/var/lib/mysql
16         networks:
17             - david
18     phpmyadmin:
19         image: phpmyadmin/phpmyadmin
20         container_name: pma
21         links:
22             - db
23         environment:
24             PMA_HOST: db
25             PMA_PORT: 3306
26             PMA_ARBITRARY: 1
27         restart: always
28         ports:
29             - 8081:80
30         networks:
31             - david
32     api1:
33         build: ./api
34
35         volumes:
36             - ./api:/src/app
37         depends_on:
38             - db
39         networks:
40             - david
41         ports:
42             - "8000:80"
```



```

43     api2:
44         build: ./Api
45
46         depends_on:
47             - db
48         networks:
49             - david
50         ports:
51             - "8001:80"
52     frontend:
53         build: ./frontend
54         volumes:
55             - ./frontend:/var/www/html
56         networks:
57             - david
58         ports:
59             - "81:80"
60     fake:
61         build: ./fake
62         volumes:
63             - ./fake:/app
64         depends_on:
65             - db
66         networks:
67             - david
68     proxy:
69         image: nginx
70         ports:
71             - "80:80"
72         volumes:
73             - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
74         networks:
75             - david
76         depends_on:
77             - api1
78             - api2
79             - frontend
80 volumes:
81     dbdata:
82
83 networks:
84     david:
85         driver: bridge

```

- Frontend – Dockerfile

```


frontend > Dockerfile
1  # Use an image de nginx como base
2  FROM nginx
3
4  # Copiar los archivos a la carpeta de trabajo de nginx
5  COPY . /usr/share/nginx/html
6
7  # Establecer la carpeta de trabajo
8  WORKDIR /usr/share/nginx/html
9
10 # Exponer el puerto 80
11 EXPOSE 80
12
13 # Ejecutar nginx
14 CMD ["nginx", "-g", "daemon off;"]

```

- Balanceador de carga (nginx.conf)

```
nginx >  nginx.conf
1  events {}
2  http {
3      upstream web {
4          server frontend:80;
5      }
6      upstream api {
7          server api1:80;
8          server api2:80;
9      }
10
11     server {
12         listen 80;
13
14         location / {
15             proxy_pass http://web;
16         }
17         location /api {
18             proxy_pass http://api;
19         }
20     }
21 }
```

- API – Dockerfile

```
api >  Dockerfile
1  FROM ubuntu:latest
2  RUN export DEBIAN_FRONTEND=noninteractive
3  RUN export TZ=Europe/Madrid
4  RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
5  RUN apt update
6  RUN apt install -y gnupg gosu curl ca-certificates zip unzip git supervisor sqlite3 libcap2-bin libpng-dev python2 dnsmutils
7  RUN curl -sS 'https://keyserver.ubuntu.com/pks/lookup?op=get&search=0x14aa48ec0831756756d7f66c4f4ea0aa5267a6c' | gpg --dearmor | tee /usr/share/keyrings/ppa_ondrej_php.gpg > /dev/null
8  RUN echo "deb [signed-by=/usr/share/keyrings/ppa_ondrej_php.gpg] https://ppa.launchpadcontent.net/ondrej/php/ubuntu jammy main" > /etc/apt/sources.list.d/ppa_ondrej_php.list
9  RUN apt update
10 RUN apt install -y php8.2-cli php8.2-dev
11 RUN apt install -y php8.2-pgsql php8.2-sqlite3 php8.2-gd
12 RUN apt install -y php8.2-curl
13 RUN apt install -y php8.2-imap php8.2-mysql php8.2-mbstring
14 RUN apt install -y php8.2-xml php8.2-zip php8.2-bcmath php8.2-soap
15 RUN apt install -y php8.2-intl php8.2-readline
16 RUN apt install -y php8.2-ldap
17 RUN apt install -y php8.2-msgpack php8.2-igbinary php8.2-redis php8.2-swoole
18 RUN apt install -y php8.2-memcached php8.2-pcov php8.2-xdebug
19 RUN php -r "readfile('https://getcomposer.org/installer');" | php -- --install-dir=/usr/bin/ --filename=composer
20 RUN curl -sLS https://deb.nodesource.com/setup_18.x | bash -
21 RUN apt install -y nodejs
22 RUN npm install -g npm
23 RUN curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | gpg --dearmor | tee /usr/share/keyrings/yarn.gpg > /dev/null
24 RUN echo "deb [signed-by=/usr/share/keyrings/yarn.gpg] https://dl.yarnpkg.com/debian/ stable main" > /etc/apt/sources.list.d/yarn.list
25 RUN apt update
26 RUN apt install -y yarn
27 RUN apt install -y mysql-client
28 RUN apt install -y apache2 libapache2-mod-php
29 RUN apt -y autoremove
30 RUN apt clean
31 RUN rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
32
33
34 WORKDIR /src/app
35
36 EXPOSE 8080
37
38 COPY [".", ".", "/src/app"]
39 RUN cd /src/app \
40     && composer install
41 COPY 000-default.conf /etc/apache2/sites-available/000-default.conf
42 COPY php.ini /etc/php/8.2/cli/conf.d/10-laravel.ini
43 RUN chmod -R 775 /src/app \
44     && chown -R www-data:www-data /src/app
45 RUN a2enmod rewrite
46 CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

- Fake – Dockerfile

```
fake > 🐳 Dockerfile
1  FROM node:latest
2
3  WORKDIR /app
4
5  COPY . .
6
7  RUN npm install
8
9  # ENV NODE_ENV=production
10
11 RUN apt-get update
12
13 EXPOSE 3306
14
15 COPY [ "./.", "./" ]
16
17 CMD [ "node", "script.js" ]
```