Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Guadalajara

Construcción se Software y Toma de Decisiones

DesarrolloWeb

Mini-Reto : Mi Web App

Alumnos

Ana Luisa G. del Rosal
A01566927

Iker Ochoa Villaseñor
A01640984

Profesor

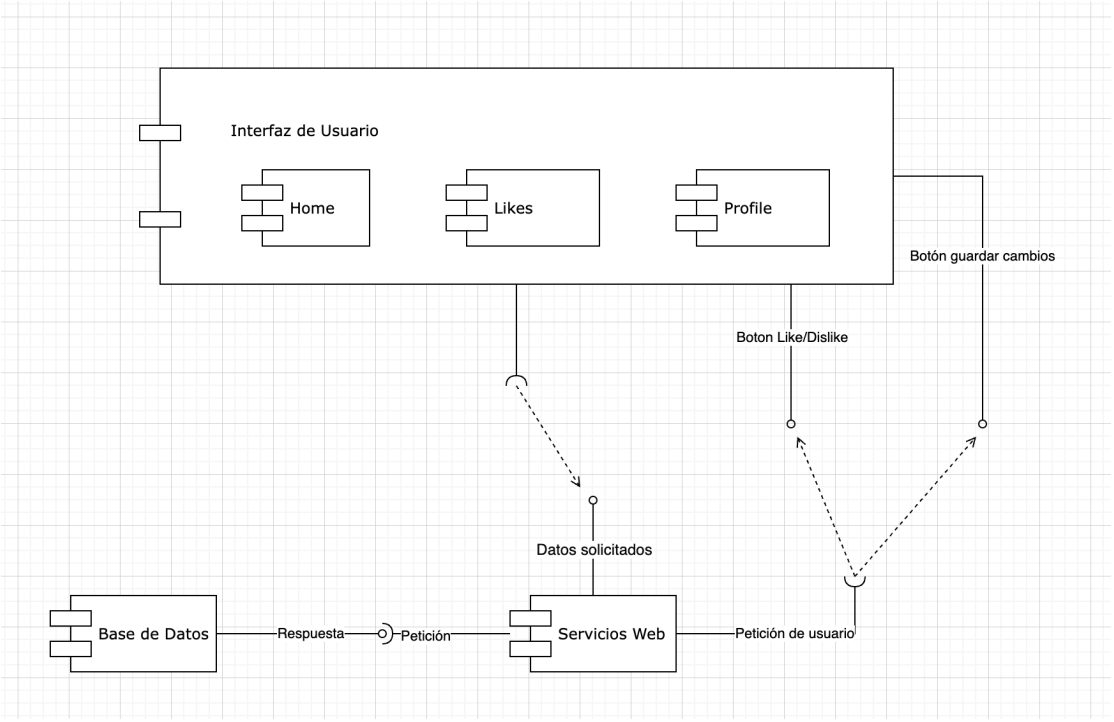
César Alfredo Espinosa Michel

30 de Abril del 2023

Mini Reto II : Smash Pass

Descripción

Smash Pass es una aplicación web que nos permite realizar un match con nuestros personajes favoritos a través de tarjetas interactivas, podrás ver los likes que has dado y decidir si mantenerlos en tus likes o descartarlos, así como crear tu perfil con información que motive que las personas a dar like y hacer match contigo.



Frontend

Dentro del código FrontEnd de nuestra App, hicimos uso de React para la construcción de nuestra Webapp, dividimos cada página en un componente distinto, esto para hacer uso de “Routes” o “Rutas” para la navegación de usuario, para las tarjetas interactivas realizamos un componentes por separado e independiente, esto para que pueda ser reutilizado y personalizado de acuerdo a las necesidades de la página.

En cuanto a la conexión entre FrontEnd o BackEnd, hicimos uso de Axios para conectar los endpoints y realizar las “Promesas” o “Promises” y así obtener la información deseada de la base de datos.

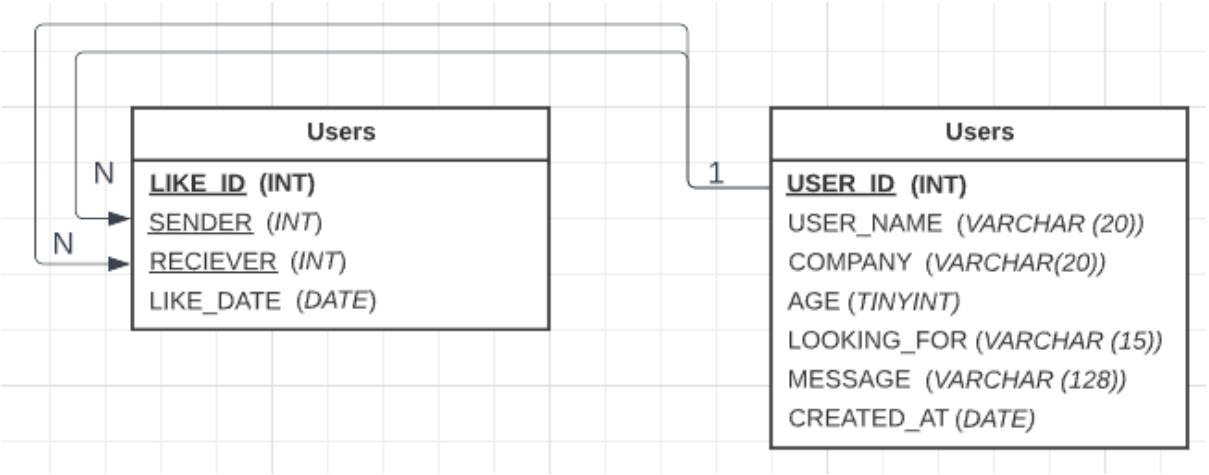
Por último, el diseño de la página se realizó con CSS puro, integrando animaciones y transiciones personalizadas por nuestro equipo y que sea amigable en cualquier pantalla en horizontal.

Backend

La página almacena su información dentro de una base de datos relacional SQL. La base incluye la información de los usuarios al igual que de los likes.

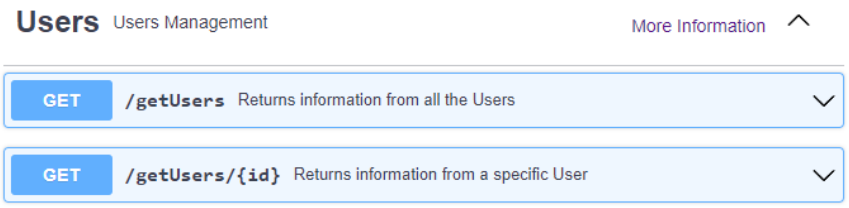
La aplicación SmashPass se encuentra en un servidor, por lo que es necesario que la base de datos no dependa de un local host (para poder tener acceso a la misma en todo momento). Por lo mismo se utilizó PlanetScale, una plataforma que aloja la base de datos sin necesidad de un servidor permanente.

A continuación se presenta un diagrama de la base de datos relacional:



El backend de la aplicación fue desarrollado con el entorno Node JS, el cual provee las herramientas necesarias para el desarrollo de un servidor y sus aplicaciones.

Por otro lado se cuenta con un archivo JavaScript, el cual se encarga de la interacción entre la aplicación y la base de datos de la misma. Se trata de la Web API diseñada para Smash Pass. Como se mencionó anteriormente, la aplicación es capaz de presentar los usuarios, generar y eliminar likes, y modificar el perfil principal. La API es llamada por la aplicación principal, respondiendo a las necesidades. La API no la utiliza directamente el cliente, sino que es una herramienta que apoya a la aplicación al procesar las llamadas de manera asincrónica, A continuación se muestran los distintos endpoints:



GET `/getUsers` Returns information from all the Users

Makes a SQL query to select all users data

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	Successful operation	*No links*
404	Users not found	*No links*
405	Validation exception	*No links*

```
//RETURNS DATA FROM ALL THE USERS
app.get('/getUsers', function (req, res, next) {
let sql = "SELECT * FROM users";
  db.query(sql, (err, result) => {
    if (err) throw err;
    res.send({ data: result });
  });
})
```

GET `/getUsers/{id}` Returns information from a specific User

Makes a SQL query to select all information from a user depending on their ID.

Parameters Cancel

Name	Description
id * required `integer($int64)` *(path)*	ID of user to return 24

Execute Clear

Curl

```
curl -X 'GET' \
  'https://minireto-api-a01566927-tecmx.vercel.app/getUsers/24' \
  -H 'accept: application/json'
```

Request URL

```
https://minireto-api-a01566927-tecmx.vercel.app/getUsers/24
```

Server response

Code	Details
200	Response body

```
[
  {
    "USER_ID": 24,
    "USER_NAME": "Doofenshmirtz",
    "COMPANY": "Disney",
    "AGE": 42,
    "LOOKING_FOR": "LOVE",
    "MESSAGE": " A platypus? (Perry puts on his hat) Perry the Platypus?!",
    "CREATED_AT": "2022-07-02T00:00:00.000Z"
  }
]
```

Download

Response headers

```
cache-control: public,max-age=0,must-revalidate
content-length: 203
content-type: application/json; charset=utf-8
```

Responses

Code	Description	Links
200	Successful operation	*No links*
404	User not found	*No links*
405	Validation exception	*No links*

```
// RETURNS DATA FROM SPECIFIC USER, THE REQUEST NEEDS AN ID
app.get('/getUsers/:id', function (req, res) {
  const user=req.params.id;
  let sql = "SELECT * FROM users WHERE USER_ID="+user+";";
    db.query(sql, (err, result) => {
      if (err) throw err;
      res.send (result);
    });
  });
```

Likes User interaction

More Information

POST `/like/{sender}/{reciever}` Generates a new like

DELETE `/delete/like/{sender}/{reciever}` Deletes a like

GET `/likes/{id}` Returns all likes from a specific User

POST `/like/{sender}/{reciever}` Generates a new like

Makes a SQL query to add a new row in the likes table.

Parameters Cancel

Name	Description
sender * required `integer($int64)` *(path)*	ID of user that generates the like 24
reciever * required `integer($int64)` *(path)*	ID of user that recieves the like 2

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'https://minireto-api-a01566927-tecmx.vercel.app/like/24/2' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
https://minireto-api-a01566927-tecmx.vercel.app/like/24/2
```

Server response

Code	Details
200	Response body

```
{
  "fieldCount": 0,
  "affectedRows": 1,
  "insertId": 59,
  "info": "",
  "serverStatus": 2,
  "warningStatus": 0
}
```

Download

Response headers

```
cache-control: public,max-age=0,must-revalidate
content-length: 92
content-type: application/json; charset=utf-8
```

Responses

Code	Description	Links
200	Successful operation	*No links*
404	Users not found	*No links*
405	Validation exception	*No links*

```
// GENERATES A NEW LIKE IN THE DATABASE
app.post('/like/:sender/:reciever', function(req,res){
  const sender=req.params.sender;
  const reciever=req.params.reciever;
  let sql="INSERT INTO likes (SENDER, RECIEVER, LIKE_DATE) VALUES ("+sender+","+reciever+",current_date());";
  db.query(sql, (err, result) => {
    if (err) throw err;
    res.send (result);
  });
})
```

DELETE `/delete/like/{sender}/{reciever}` Deletes a like

Makes a SQL query to remove a row in the likes table.

Parameters Cancel

Name	Description
sender * required integer($int64) *(path)*	ID of user that generates the like 24
reciever * required integer($int64) *(path)*	ID of user that recieves the like 2

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  'https://minireto-api-a01566927-tecmx.vercel.app/delete/like/24/2' \
  -H 'accept: application/json'
```

Request URL

```
https://minireto-api-a01566927-tecmx.vercel.app/delete/like/24/2
```

Code	Details
200	Response body

```
{
  "fieldCount": 0,
  "affectedRows": 3,
  "insertId": 0,
  "info": "",
  "serverStatus": 2,
  "warningStatus": 0
}
```

Download

Response headers

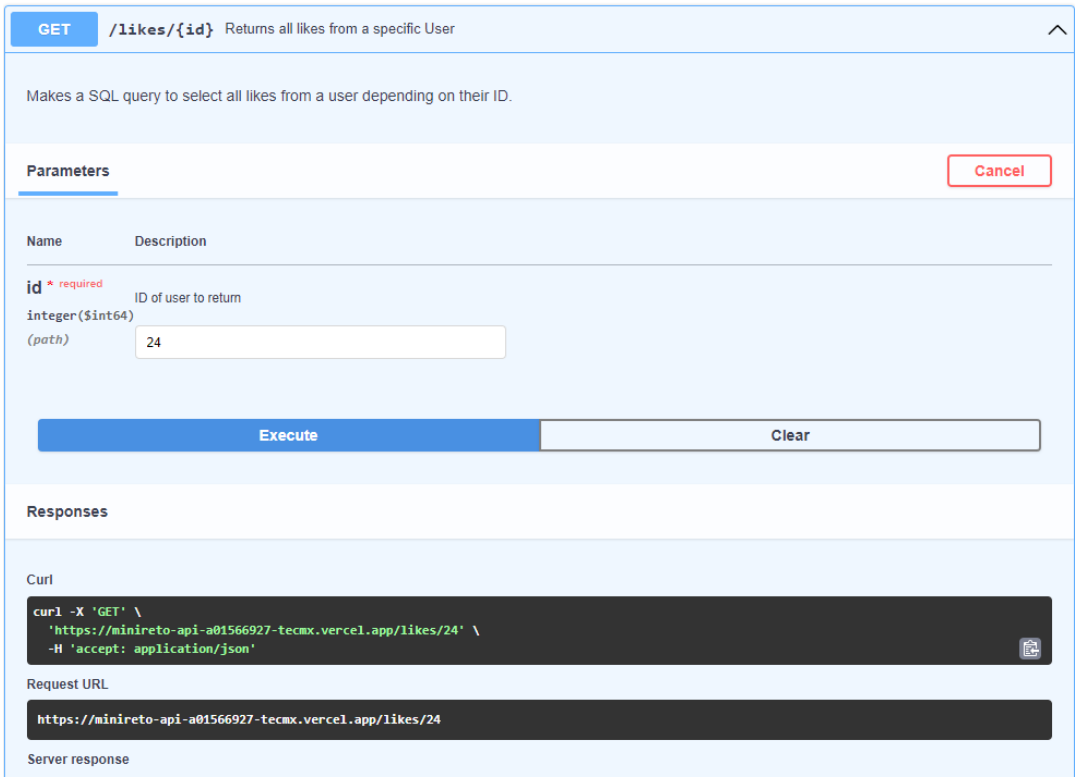
```
cache-control: public,max-age=0,must-revalidate
content-length: 91
content-type: application/json; charset=utf-8
```

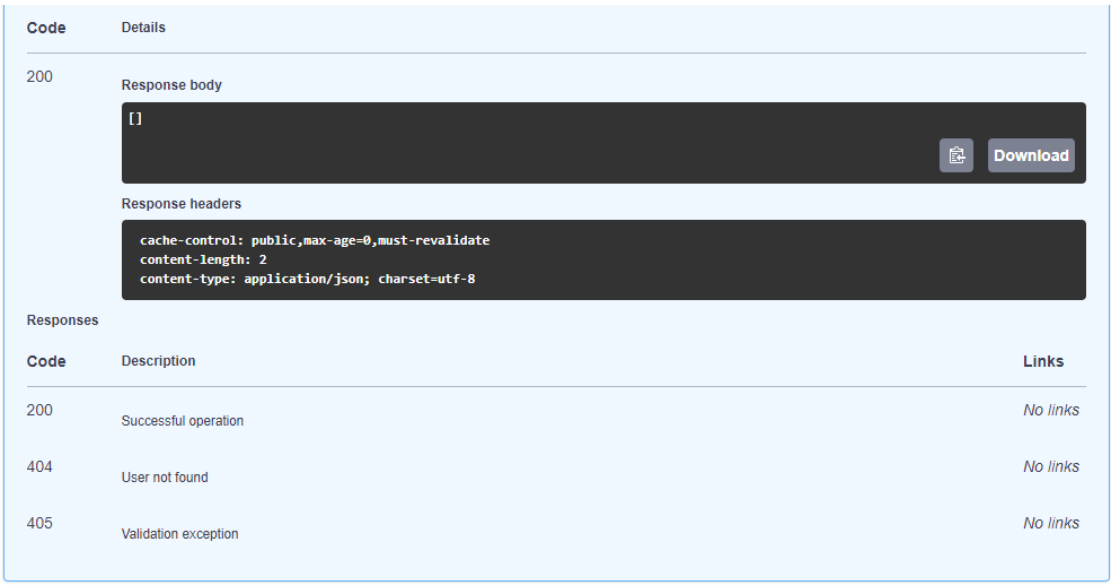
Responses

Code	Description	Links
200	Successful operation	*No links*
404	Like not found	*No links*
405	Validation exception	*No links*

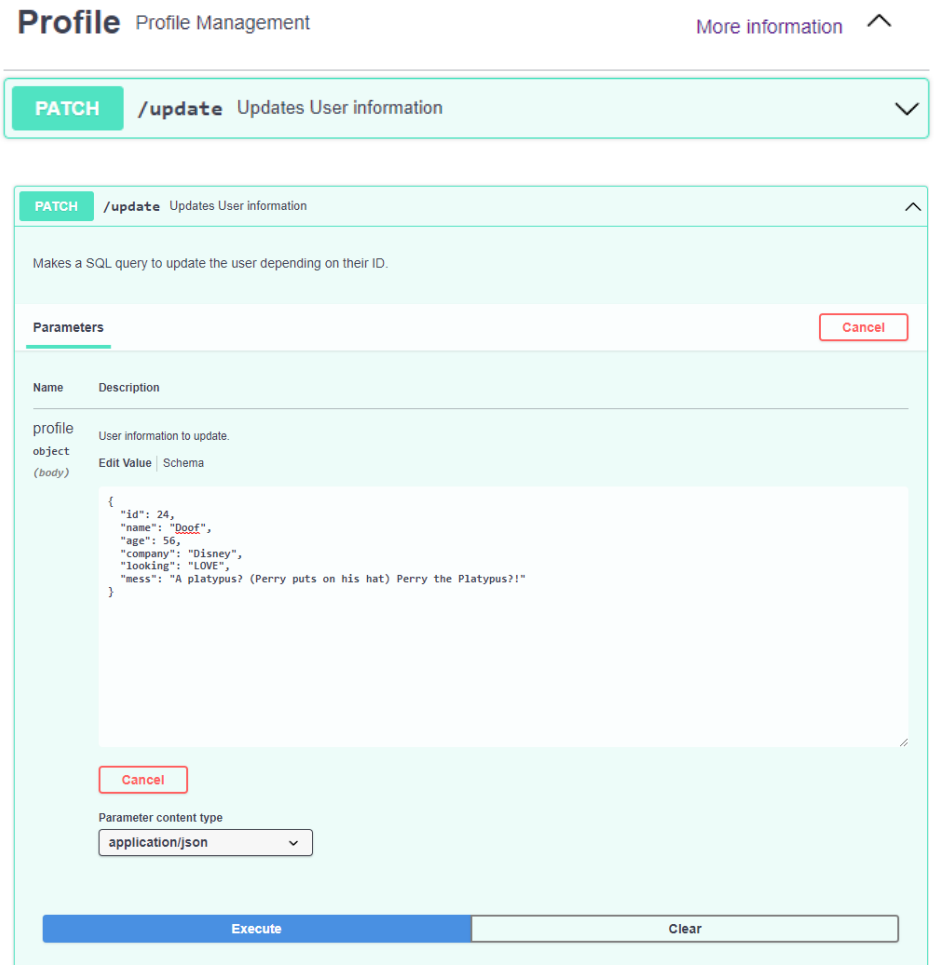
```
// DELETES A LIKE
app.delete('/delete/like/:sender/:reciever', function(req,res){
  const sender=req.params.sender;
  const reciever=req.params.reciever;

  let sql="DELETE FROM likes WHERE SENDER='"+sender+"' AND RECIEVER='"+reciever+"';";
  db.query(sql, (err, result) => {
    if (err) throw err;
    res.send (result);
  });
})
```





```
// RETURNS ALL LIKED PROFILES DEPENDING FROM THE CURRENT USER
app.get('/likes/:sender', function(req,res){
  const sender=req.params.sender;
  let sql="SELECT  USER_ID, USER_NAME, COMPANY, AGE, LOOKING_FOR, MESSAGE FROM users INNER JOIN likes ON users.USER_ID = likes.RECIEVER AND likes.SENDER="+sender+";";
  db.query(sql, (err, result) => {
    if (err) throw err;
    res.send (result);
  });
});
```



Responses

Code	Description	Links
200	Successful operation	*No links*
404	User not found	*No links*
405	Validation exception	*No links*

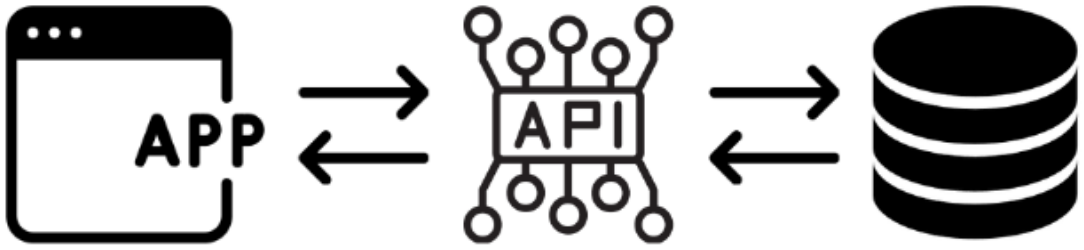
```
// UPDATES THE USER INFORMATION
app.patch('/update', function(req,res){
  const id=req.body.id;
  const name=req.body.name;
  const age=req.body.age;
  const company=req.body.company;
  const looking=req.body.looking;
  const mess=req.body.mess;

  let sql="UPDATE users SET USER_NAME='"+name+"', COMPANY='"+company+"', AGE="+age+", LOOKING_FOR='"+looking+"', MESSAGE='"+mess+"' WHERE USER_ID="+id+";";
  db.query(sql, (err, result) => {
    if (err) throw err;
    res.send (result);
  });

})
```

Funcionamiento

SmashPass vive en un servidor de Vercel, una plataforma de nube que provee los recursos necesarios para poder hacer un “deployment” de la aplicación. En este caso se cuenta con dos servidores distintos: uno que maneja la RestAPI y otro que presenta la aplicación al público. De tal manera es posible generar una conexión entre la aplicación y la base de datos sin necesidad de un servidor local.



Links

1. *Repositorio Github:* https://github.com/Iker8av/ProyectoWeb-Tinder
2. *Página Web:* https://smash-pass-a01566927-tecmx.vercel.app/