

Memoria del proyecto

Lucky Picks



Lucky Picks	1
Descripción	3
Objetivos	4
Tecnologías utilizadas	5
Frontend:	5
Backend:	5
Entorno de desarrollo:	5
Entorno de despliegue:	5
Origen de datos:	6
Diagrama entidad relación de la base de datos	7
Código fuente	8
Frontend:	8
Backend:	11
Arquitectura del Sistema	14
Backend – Laravel y el patrón MVC	14
Frontend – React	14
Comunicación Frontend-Backend	15
Despliegue en Apache	15
Conclusión	15
Instrucciones de Despliegue	16
Instrucciones de Despliegue en Servidor	16
1. Preparación del servidor	16
1.1. Instalación de Apache	16
1.2. Instalación de un servidor FTP	16
2. Configuración de Apache	17
2.1. Activación de módulos necesarios	17
3. Backend (Laravel)	17
3.1. Instalación de Composer y PHP	17
3.2. Instalación de MySQL Server	17
3.3. Migraciones	17
3.4. Ejecución automática del backend	18
4. Despliegue del Frontend	18
5. Configuración de dominio y HTTPS	19
5.1. Configuración de DNS y dominio	19
5.2. Obtención de certificado SSL con Certbot	20
Resultado final	20
Despliegue en local:	20
Requisitos:	20
Pasos a seguir:	20
Manuales	21
Manual de usuario:	21
Acceder a la web:	21
Obtención de saldo:	23
Datos del perfil:	24
Paneles de administración	25
Panel con el rendimiento	25
Panel de usuarios	25

Descripción

LuckyPicks es una web de juegos de azar
La web cuenta con un inicio de sesión para que los usuarios puedan ver su historial de partidas y llevar un mejor seguimiento

Nos encontraremos con los típicos juegos de azar, como la slot machine y el blackjack, esta diseñado para la implementación y modificación de los juegos por parte del equipo de LuckyPicks, haciendo posible la futura incorporación de nuevos juegos con los que pasar el rato

En el cual podremos gastar nuestro balance para obtener más o perderlo, todo en tiempo real y registrado así podrás ver tu progreso.

También cuenta con un sistema de baneos, en caso de haber sido baneado encontrarás una cuenta atrás hasta el momento del desbaneo del usuario.

En caso de ser un administrador cuentas con más apartados, los cuales te dan acceso a administrar los diferentes usuarios, pasando desde ver su historial (partidas jugadas juegos favoritos etc) hasta administrarlos, cambiarles las contraseñas, añadirles saldo, cambiarles el rol e incluso banearlos.

También el rol de administrador cuenta con otras ventajas, obtener los datos de los juegos con su rentabilidad, haciendo posible llevar un buen seguimiento de la rentabilidad de la web viendo lo que genera cada juego y el momento en el que más se jugó, dando la posibilidad de filtrar la información para obtenerla del rango de fechas necesarios, ya sea un día concreto, un mes, trimestre o lo que se necesite.

Objetivos

Con este proyecto quise desarrollar diferentes aspectos que considero importante los cuales son:

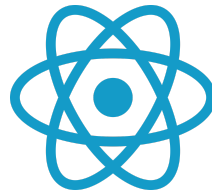
- Creación de un login y apartado de registro sólido, con el uso de localStorage, cookies, y tokens de inicio de sesión para que sea una web con cierta seguridad.
- Dominar la inserción y extracción de datos de una base de datos, en el caso de LuckyPicks lo vemos más notablemente en el panel de administrador, dando la posibilidad de crear informes de manera fácil y simple haciendo un par de clicks.
- Crear un buen diseño accesible e intuitivo para los diferentes usuarios, para un correcto funcionamiento en cualquier dispositivo, desde ordenador hasta móvil.
- Creación de un buen sistema de roles, en el que haya funcionalidades exclusivas para roles concretos, dando la posibilidad de crear aplicaciones web más completas.
- Dominar frameworks de backend de MVC (Modelo vista controlador), usando con facilidad las rutas, controladores, modelos, middleware etc
- Capacidad de usar frameworks de frontend como react, adaptandome al uso de componentes, las cargas de dichos componentes y el funcionamiento general de los frameworks SPA (Single Page Application).
- Dominar más apartados de JavaScript para la creación de juegos más complejos y visuales.

Tecnologías utilizadas

Frontend:

Para el frontend utilizaremos el framework de React con la versión 19.1.

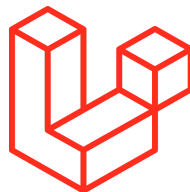
El motivo por el cual usare react es por su gran fluidez y manejo, considero que es fácil de usar y es muy rápido en tiempos de carga al ser un SPA (Single Page Application), de esta manera la experiencia del usuario es mejor y para el desarrollador también es cómodo de utilizar.



Backend:

En el backend de la aplicación nos encontraremos con laravel 12, el framework mas usado de PHP.

Seleccione laravel 12 para el desarrollo de la web porque es moderno y bastante popular y el hecho de que sea un modelo vista controlador hace que sea muy fácil de usar una vez que sabes la estructura y el funcionamiento del framework.



Entorno de desarrollo:

Aquí nos encontramos con Visual Studio Code, un editor de código ligero, con gran cantidad de extensiones y personalización que mejora la experiencia de usuario, con la versión 1.100.3.



Entorno de despliegue:

Entorno seleccionado: Azure

En el despliegue se eligió Azure, el cual tiene un buen servicio de máquinas virtuales, las cuales nos da la posibilidad de administrar completamente y crear una aplicación escalable, para ello tenemos control total de la máquina, como también de su

cortafuego, dándonos la opción de abrir los puertos necesarios para en mi caso acceder remotamente desde ssh, crear un servidor ftp para subir los archivos de mi web, crear un servidor dns para usar un dominio entre otros.

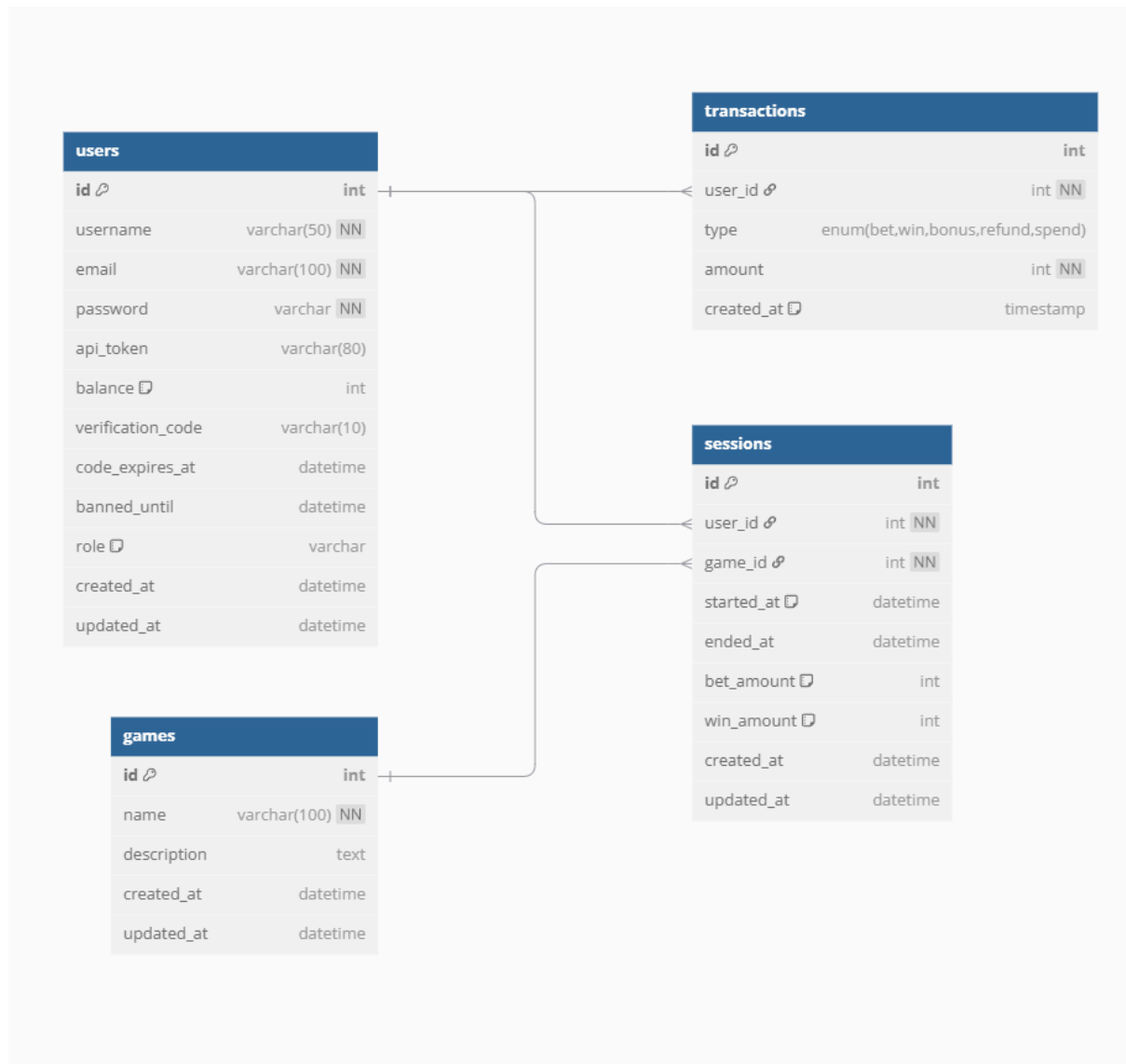


Origen de datos:

En este caso use MySQL con el motor de almacenamiento InnoDB, en su versión 8.0.36.



Diagrama entidad relación de la base de datos



Código fuente

Frontend:

- Rutas:

Esto se encarga de redireccionar en caso de no estar logueados o no tener el rol correcto, impidiendo que se carguen los componentes que no deberíamos de ver.

```
        <UserHistory />
      ) : (
        <Navigate to="/Home" replace />
      )
    }
  />
  <Route
    path="/games-performance"
    element={
      isUserBanned ? (
        <Navigate to="/banned" replace />
      ) : isAuthenticated && isAdmin ? (
        <GamesPerformance />
      ) : (
        <Navigate to="/Home" replace />
      )
    }
  />

  <Route
    path="*"
    element={
      isUserBanned ? (
        <Navigate to="/banned" replace />
      ) : isAuthenticated ? (
        <Navigate to="/Home" replace />
      ) : (
        <Navigate to="/" replace />
      )
    }
  />
</Routes>
</BrowserRouter>
</React.StrictMode>
);
```


- Juegos:

En esta imagen encontramos varias cosas importantes, primero las peticiones al backend, en las que vemos un bearer token el cual es para afirmar que el usuario está logueado (un token generado aleatoriamente al loguearse) y también vemos parte de la lógica del juego del BlackJack, en la que se crea la baraja de cartas, y se devuelve el valor dependiendo de la carta

```
const registrarSesion = async (betAmount, winAmount, endedAt) => {
  try {
    const res = await fetch(`${API_URL}/sessions`, {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${TOKEN}`, // Se envía el Bearer Token
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        user_id: localStorage.getItem('id'),
        game_id: 1,
        bet_amount: betAmount,
        win_amount: winAmount,
        ended_at: endedAt,
      }),
    });

    if (!res.ok) throw new Error("Error al registrar sesión");
  } catch (error) {
    console.error("Error en registrarSesion:", error);
  }
};

const formatDateForMySQL = (date) => {
  const d = new Date(date);
  const pad = (n) => n.toString().padStart(2, '0');
  return `${d.getFullYear()}-${pad(d.getMonth() + 1)}-${pad(d.getDate())} ${pad(d.getHours())}:${pad(d.getMinutes())}:${pad(d.getSeconds())}`;
};

const crearBaraja = () => {
  const palos = ['♥', '♦', '♣', '♠'];
  const valores = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A'];
  let nuevaBaraja = [];
  palos.forEach(palo => {
    valores.forEach(valor => {
      nuevaBaraja.push({ valor, palo });
    });
  });
  for (let i = nuevaBaraja.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [nuevaBaraja[i], nuevaBaraja[j]] = [nuevaBaraja[j], nuevaBaraja[i]];
  }
  return nuevaBaraja;
};

const valorCarta = (valor) => {
  if (['J', 'Q', 'K'].includes(valor)) return 10;
  if (valor === 'A') return 11;
  return parseInt(valor);
};
```

- **Añadir saldo:**

Aquí veremos un par de cosas presentes en más componentes, como sería el `triggerBalanceAnimation`, es el encargado de hacer la animación de perder o ganar dinero, se usa en los juegos para hacer más visible la pérdida y ganancia de saldo, también vemos como esta estructurado parte del html, siendo dinámico, da la posibilidad quitar o añadir más opciones para ingresar dinero simplemente modificando una pequeña línea de código

```
const triggerBalanceAnimation = (oldBalance, newBalance) => {
  const change = newBalance - oldBalance;
  const event = new CustomEvent('balanceUpdate', { detail: { newBalance, change } });
  window.dispatchEvent(event);
};

const confirmAddBalance = async () => {
  if (promoCode.trim().toUpperCase() === validPromoCode.toUpperCase()) {
    await updateBalance(selectedAmount);
    await registrarTransaccion('spend', selectedAmount);
    closeModal();
  } else {
    setIsPromoInvalid(true);
  }
};

return (
  <div id='webGames'>
    <Header />
    <InfoBar />
    <div className='Titulo'>
      <h1>Añadir saldo</h1>
    </div>

    <div className="contenedor10 linksSinEstilo">
      <div className="todasOpciones">
        {[5, 10, 20, 50, 100, 200].map((amount, idx) => (
          <div
            key={amount}
            className="opcion"
            id={`saldo${idx + 1}`}
            onClick={(e) => handleClick(e, amount)}
          >
            <Link className="enlaceAddBalance">
              <h2>{amount} €</h2>
            </Link>
          </div>
        )]}
      </div>
    </div>
  </div>
)
```

Backend:

- **Rutas Api:**

En esta imagen vemos todas las rutas posibles para hacer peticiones al backend, como vemos hay dos grupos, los que usan el middleware y los que no, la función del middleware la explicaré más adelante.

También podemos ver como funciona, importamos el controlador a usar, establecemos la url a la que se hará la petición y establecemos la 'acción' del controlador.

```
use App\Http\Controllers\SessionController;
use App\Http\Controllers\GamePerformanceController;
use App\Http\Controllers\BanController;
use App\Http\Controllers\AccountHistoryController;

// Rutas protegidas con el token de inicio de sesion
Route::middleware('auth.token')->group(function () {
    Route::get('/top-wins', [SessionController::class, 'topWins']);
    Route::post('/account/history', [AccountHistoryController::class, 'history']);
    Route::post('/check-ban', [BanController::class, 'checkBan']);
    Route::get('/games/performance', [GamePerformanceController::class, 'performance']);
    Route::post('/sessions', [SessionController::class, 'store']);
    Route::get('/users', [UserController::class, 'index']);
    Route::put('/users/{id}', [UserController::class, 'update']);
    Route::post('/transaction', [TransactionController::class, 'create']);
    Route::post('/update-balance', [UserBalance::class, 'updateBalance']);
});

// Rutas públicas (sin autenticación)
Route::middleware('guest')->post('/register', [AuthController::class, 'register']);
Route::post('/login', [AuthLogin::class, 'login']);
```

- **Middleware:**

Aquí establecemos un middleware, simplemente comprueba de que el token de inicio de sesión exista, con esto evitamos que alguien pueda acceder y usar la app sin haberse creado una cuenta

```
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use App\Models\User;
//Este middleware obtiene un token aleatorio que se genera al iniciar sesion
//comprueba si existe para validar las peticiones
class AuthenticateWithToken
{
    public function handle(Request $request, Closure $next)
    {
        $token = $request->bearerToken();

        if (!$token || !User::where('api_token', $token)->exists()) {
            return response()->json(['error' => 'Unauthorized'], 401);
        }

        return $next($request);
    }
}
```

- **Modelo:**

En el establecemos los campos con los que podemos interactuar, así cuando el controlador quiera implementar datos a una tabla usa su modelo y puede así insertar, borrar o modificar datos

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Session extends Model
{
    protected $fillable = [
        'user_id', 'game_id', 'bet_amount', 'win_amount', 'ended_at',
    ];
}
```

- **Controlador:**

El controlador más usado, encargado de añadir balance o quitarlo al usuario correspondiente

```
class UserBalance extends Controller
{
    public function updateBalance(Request $request)
    {
        $request->validate([
            'id' => 'required|exists:users,id',
            'amount' => 'required|integer',
        ]);

        $user = User::findOrFail($request->id);

        if ($user->balance + $request->amount < 0) {
            $user->balance = 0;
            $user->save();
            return response()->json([
                'message' => 'Balance actualizado correctamente.',
                'balance' => $user->balance,
            ]);
        }

        $user->balance += $request->amount;
        $user->save();

        return response()->json([
            'message' => 'Balance actualizado correctamente.',
            'balance' => $user->balance,
        ]);
    }
}
```

Arquitectura del Sistema

La arquitectura del proyecto se basa en un enfoque de desarrollo web moderno que combina una aplicación **frontend en React** con un **backend desarrollado en Laravel**, todo desplegado en un entorno **Apache**. Esta separación de responsabilidades permite una mejor mantenibilidad, escalabilidad y experiencia de usuario.

Backend – Laravel y el patrón MVC

El backend está construido con **Laravel**, un framework PHP que implementa el patrón arquitectónico **MVC (Modelo-Vista-Controlador)**. Este patrón separa claramente la lógica de negocio, la gestión de datos y la presentación, facilitando la organización del código y la colaboración entre desarrolladores.

- **Modelo (Model):** Encargado de la lógica de acceso a datos y la representación de las entidades del sistema. Laravel proporciona Eloquent ORM, lo que simplifica el trabajo con bases de datos relacionales.
- **Vista (View):** Aunque el frontend se gestiona externamente con React, Laravel puede generar vistas para algunos procesos específicos del servidor, como emails o respuestas simples.
- **Controlador (Controller):** Actúa como intermediario entre los modelos y las vistas, gestionando la lógica de la aplicación y las peticiones del cliente. En este proyecto, los controladores se centran en exponer endpoints API RESTful que luego consume el frontend.

Además, Laravel incluye funcionalidades integradas como middleware, validaciones, rutas, servicios, y autenticación, que han sido aprovechadas para crear una API robusta y segura.

Frontend – React

El frontend ha sido desarrollado utilizando **React**, una biblioteca de JavaScript orientada a la construcción de interfaces de usuario reactivas y modulares. React permite el uso de componentes reutilizables, lo que mejora la mantenibilidad y escalabilidad del código.

- La interfaz se comunica con el backend a través de **peticiones HTTP (fetch/axios)** hacia la API de Laravel.
- Se utilizan **hooks** y el sistema de estado de React para controlar la lógica del lado del cliente.
- La separación del frontend del backend también permite futuras adaptaciones, como el uso de aplicaciones móviles que consuman la misma API.

Comunicación Frontend-Backend

El intercambio de datos entre React y Laravel se realiza a través de **peticiones RESTful**, generalmente en formato JSON. Laravel actúa como proveedor de datos y lógica del negocio, mientras que React se encarga de mostrar y manipular la información en el navegador.

Despliegue en Apache

El sistema está desplegado en un servidor **Apache**, que actúa como servidor web para ambas partes:

- Apache sirve el **build de React**, que se genera a partir del código fuente con herramientas como Vite.
- A través de reglas de configuración (por ejemplo, **.htaccess**), Apache también redirige las peticiones API al backend Laravel, que normalmente se encuentra en un subdirectorio o ruta específica (como **/api**).
- Se ha configurado correctamente el **enrutamiento tanto para React (SPA) como para Laravel**, evitando conflictos y asegurando una navegación fluida en el navegador.

Conclusión

Esta arquitectura desacoplada, basada en Laravel como backend API y React como frontend SPA (Single Page Application), permite un desarrollo limpio, modular y preparado para ampliaciones futuras. La integración mediante Apache garantiza un despliegue sencillo y estable, aprovechando tecnologías ampliamente utilizadas y documentadas.

Instrucciones de Despliegue

Instrucciones de Despliegue en Servidor

Para desplegar la aplicación web en un servidor, se siguió un conjunto de pasos que permitieron configurar correctamente tanto el servidor web como el entorno necesario para ejecutar Laravel y React. A continuación se detallan los procedimientos realizados:

1. Preparación del servidor

1.1. Instalación de Apache

Se instaló **Apache2**, que será el servidor web encargado de servir tanto la aplicación frontend (React) como de redirigir las peticiones API al backend (Laravel).

Comando para instalar Apache:

sudo apt update

sudo apt install apache2

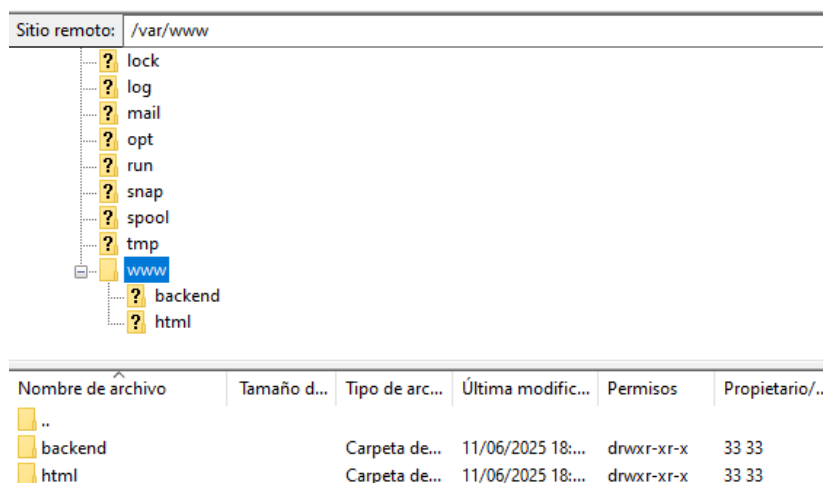
1.2. Instalación de un servidor FTP

Para poder subir archivos al servidor desde un entorno local (como Visual Studio Code o FileZilla), se instaló y configuró un servidor **FTP** (por ejemplo, vsftpd). Es importante asegurarse de otorgar los permisos adecuados a las carpetas de destino, especialmente **/var/www**.

Comando para dar permisos:

sudo chown -R www-data:www-data /var/www

sudo chmod -R 755 /var/www



Nombre de archivo	Tamaño d...	Tipo de arc...	Última modific...	Permisos	Propietario/...
..					
backend		Carpeta de...	11/06/2025 18:...	drwxr-xr-x	33 33
html		Carpeta de...	11/06/2025 18:...	drwxr-xr-x	33 33

2. Configuración de Apache

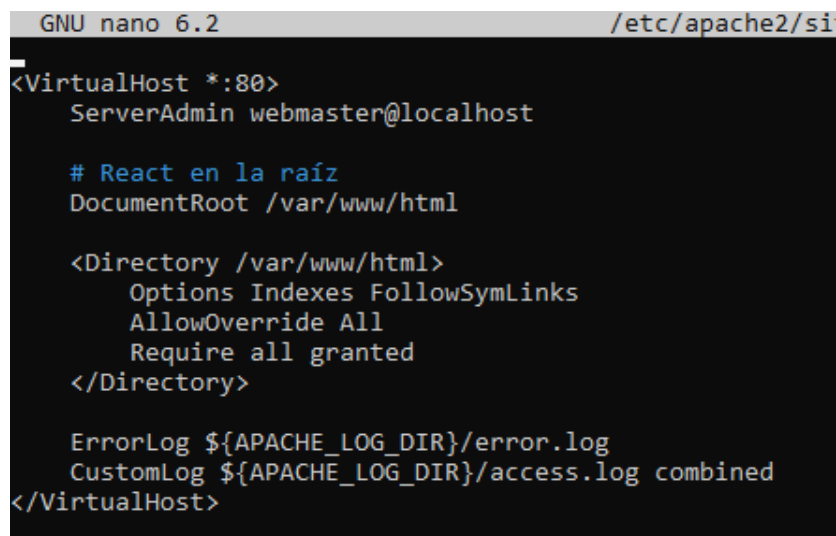
2.1. Activación de módulos necesarios

Se activaron los módulos de Apache necesarios para que Laravel funcione correctamente, en especial el módulo **rewrite**:

sudo a2enmod rewrite

sudo systemctl restart apache2

También se ajustaron las configuraciones de Apache (/etc/apache2/sites-available/000-default.conf) para que permita **.htaccess**:



```
GNU nano 6.2 /etc/apache2/si
<VirtualHost *:80>
    ServerAdmin webmaster@localhost

    # React en la raíz
    DocumentRoot /var/www/html

    <Directory /var/www/html>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

3. Backend (Laravel)

3.1. Instalación de Composer y PHP

Se instaló **Composer**, el gestor de dependencias de PHP. Al instalarlo, también se instalaron las dependencias necesarias de PHP:

sudo apt install php-cli unzip curl

curl -sS https://getcomposer.org/installer | php

sudo mv composer.phar /usr/local/bin/composer

3.2. Instalación de MySQL Server

Para gestionar la base de datos, se instaló **MySQL Server**:

sudo apt install mysql-server

Se creó una base de datos y un usuario con permisos para ser utilizado por Laravel.

3.3. Migraciones

Una vez configurada la base de datos, se ejecutaron las migraciones de Laravel:

php artisan migrate

3.4. Ejecución automática del backend

Para mantener el backend corriendo constantemente con **php artisan serve**, se creó una tarea cron que lo reinicia cada 15 minutos:

crontab -e

Y se añadió la siguiente línea:

***/15 * * * * cd /var/www/backend && php artisan serve --host=0.0.0.0 --port=8000**

Esto permite que el servidor esté disponible en la IP pública del sistema.

4. Despliegue del Frontend

El frontend en React se compila con:

npm run build

```
<?php
$laravelHost = 'http://127.0.0.1:8000';

// Construir URL destino con path y query
$path = $_SERVER['REQUEST_URI'];
$url = $laravelHost . $path;

// Inicializar cURL
$ch = curl_init($url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HEADER, true);

// Pasar método HTTP
$method = $_SERVER['REQUEST_METHOD'];
curl_setopt($ch, CURLOPT_CUSTOMREQUEST, $method);

// Pasar cuerpo si es POST, PUT, etc.
$input = file_get_contents('php://input');
if (!empty($input)) {
    curl_setopt($ch, CURLOPT_POSTFIELDS, $input);
}

$headers = [];

// Obtener cabeceras del cliente
$clientHeaders = getallheaders();
foreach ($clientHeaders as $key => $value) {
    if (strtolower($key) === 'authorization') {
        $headers[] = "Authorization: $value";
    } else {
        $headers[] = "$key: $value";
    }
}

// Fallback por si Authorization no llega vía getallheaders()
if (!isset($clientHeaders['Authorization']) && isset($_SERVER['HTTP_AUTHORIZATION'])) {
    $headers[] = "Authorization: {$_SERVER['HTTP_AUTHORIZATION']}";
}

curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
file_put_contents('proxy-debug.log', print_r($headers, true));

// Ejecutar la petición
$response = curl_exec($ch);
$headerSize = curl_getinfo($ch, CURLINFO_HEADER_SIZE);
$header = substr($response, 0, $headerSize);
$body = substr($response, $headerSize);

// Enviar headers al cliente
foreach (explode("\r\n", $header) as $line) {
    if (strpos($line, 'Transfer-Encoding') === false && $line !== '') {
        header($line, false);
    }
}

echo $body;
curl_close($ch);
```

El contenido generado en la carpeta **dist** se mueve a una carpeta servida por Apache (en nuestro caso, **/var/www/html**).

Tras esto tendremos que crear una carpeta denominada **api**, la cual tendrá dentro un **index.php** el cual servirá como **proxy** y un **.htaccess** para que se ejecute correctamente el proxy:

5. Configuración de dominio y HTTPS

5.1. Configuración de DNS y dominio

Para acceder a la aplicación mediante un nombre de dominio personalizado, se instaló y configuró un **servidor DNS** en la misma máquina (por ejemplo, usando Bind9).





Posteriormente se **obtuvo un dominio** (en este caso, **luckypicks.es**) y se configuró para que **apunten al servidor DNS** instalado.

```
GNU nano 6.2 /etc/bind/zones/db.luckypicks.es
$TTL 604800
@ IN SOA ns1.luckypicks.es. admin.luckypicks.es. (
2025061202 ; Serial (incrementa si ya editaste antes)
604800 ; Refresh
86400 ; Retry
2419200 ; Expire
604800 ) ; Negative Cache TTL
;
@ IN NS ns1.luckypicks.es.
@ IN NS ns2.luckypicks.es.
ns1 IN A 52.164.222.56
ns2 IN A 52.164.222.56
@ IN A 52.164.222.56
www IN A 52.164.222.56
```

Recuerda abrir el puerto 53 UDP/TCP, es el que usará el servidor DNS

Tras la configuración del servidor DNS en nuestro panel del dominio hay que poner como servidor DNS primario el que acabamos de crear, para eso pondremos de nombre ns1.luckypicks.es y la ip publica del servidor, una vez configurado esto toca esperar a que se propaguen los cambios por internet a los distintos servidores DNS

Con el dominio también es necesario poner que la dirección ip sea la misma que la de tu maquina, y tambien es recomendable crear un nuevo registro con el nombre de www, para poder acceder a la web mediante www.luckypicks.es

<input type="checkbox"/>	TIPO	NOMBRE DE HOST	VALOR	SERVICIO ▼	ACCIONES
<input type="checkbox"/>	A	www	52.164.222.56	-	 
<input type="checkbox"/>	A	@	52.164.222.56	-	 

5.2. Obtención de certificado SSL con Certbot

Una vez configurado el dominio y verificado su correcto funcionamiento, se instaló un **certificado SSL gratuito** utilizando Certbot para habilitar HTTPS:

```
sudo snap install --classic certbot  
sudo certbot --apache
```

Decir cual es tu dominio y se pondra automaticamente el HTTPS

Ten en cuenta que las peticiones a la api ya no podrán ser HTTP, hay que modificarlas para que sean HTTPS

Esto asegura que la conexión entre los usuarios y el servidor sea cifrada y segura.

Resultado final

Tras completar todos estos pasos, la aplicación quedó correctamente desplegada y accesible desde:

<https://www.luckypicks.es>

Despliegue en local:

Requisitos:

Requisitos Xampp Tener instalado PHP y Composer

Pasos a seguir:

1. Inicie MySQL y apache en xampp
2. Revise el .env dentro de la carpeta Backend para poner una cuenta de mysql válida
3. Dentro de la carpeta Backend ejecute el comando php artisan migrate (Crearé la base de datos con sus tablas y relaciones)
4. Posteriormente, en la misma carpeta de Backend ejecutar el comando php artisan serve
5. Dentro de la carpeta Frontend ejecute el comando npm install (Instala las dependencias necesarias del proyecto)
6. Posteriormente ejecutar el comando npm run dev
7. Acceda a la url proporcionada para ver el proyecto lanzado

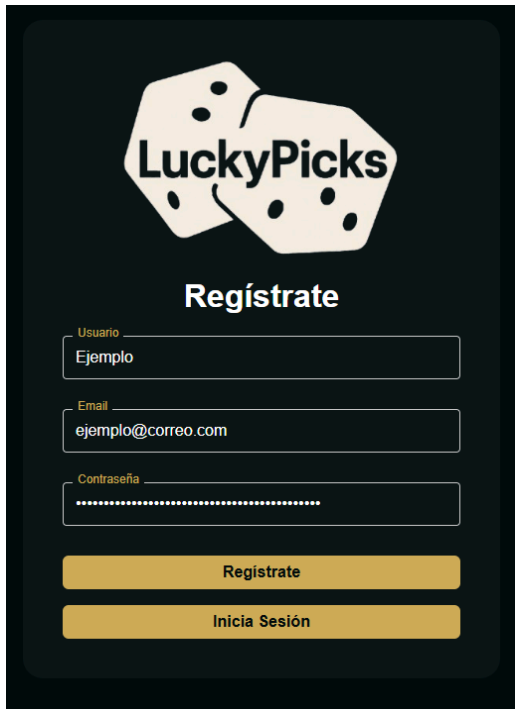
Manuales

Manual de usuario:

Acceder a la web:

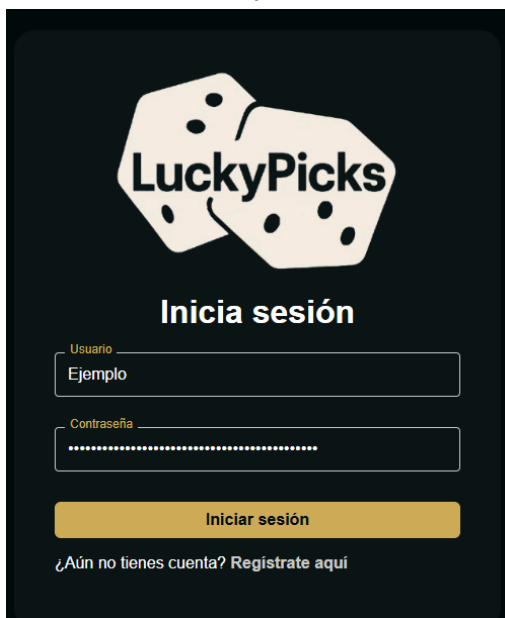
Accede a la url de la web <https://www.luckypicks.es/>

Una vez ingresas por primera vez debes de crearte una cuenta, para ello hay un botón el cual pone '¿Aún no tienes cuenta? Regístrate aquí' y le llevara a este apartado

The image shows a registration form for 'Lucky Picks'. At the top is the logo, which consists of two dice and the text 'Lucky Picks'. Below the logo is the heading 'Regístrate'. There are three input fields: 'Usuario' with the placeholder 'Ejemplo', 'Email' with the placeholder 'ejemplo@correo.com', and 'Contraseña' with a masked password. Below these fields are two yellow buttons: 'Regístrate' and 'Inicia Sesión'.

Deberás rellenar los campos, con tu nombre de usuario, correo y una contraseña de al menos 8 caracteres, Tras haberlo rellenado, al darle click a registrarse será redireccionado al login

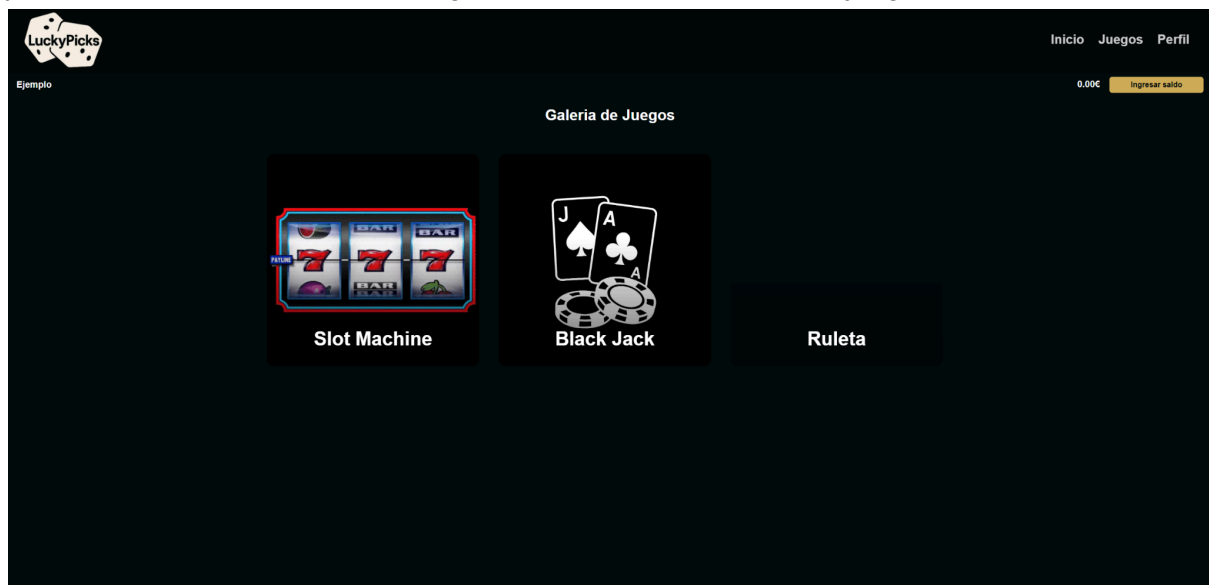
Tras este paso solo tendrá que poner su usuario y contraseña en caso de que no este ya puesta por defecto y darle a iniciar sesión

The image shows a login form for 'Lucky Picks'. At the top is the logo, which consists of two dice and the text 'Lucky Picks'. Below the logo is the heading 'Inicia sesión'. There are two input fields: 'Usuario' with the placeholder 'Ejemplo' and 'Contraseña' with a masked password. Below these fields is a yellow button labeled 'Iniciar sesión'. At the bottom of the form, there is a link that says '¿Aún no tienes cuenta? Regístrate aquí'.

Ahora sera redirigido a la web, en primera instancia vera el home, en el cual se ve el mayor premio que solto la web y a quien fue, con los puestos que los suceden



Como se puede ver arriba en la barra de navegacion hay dos apartados mas, el de Juegos y el Perfil, si accedemos al de Juegos veremos un menu con los juegos disponibles.



Si hacemos click nos llevara al respectivo juego, en el cual podremos apostar nuestro saldo y jugar, para ello deberemos de obtener saldo

Obtención de saldo:

En cualquier apartado dentro del área de Juegos tendremos una barra superior en la que nos aparecerá nuestro saldo actual y un botón para ingresar dinero. Dicho botón nos lleva a un menú con diferentes cantidades a ingresar, si hacemos click en la deseada aparecerá un popup en el cual pondremos el código promocional 'LUCKYPICKS2025' y nos dará la cantidad deseada.



Una vez tenemos saldo podremos comenzar a jugar.

Datos del perfil:

Si accedemos al apartado del perfil encontraremos tres tablas plegadas para ocupar poco espacio, Nuestro nombre junto a un boton para cerrar sesion y unos parametros para la obtencion de datos de las tablas

[Inicio](#) [Juegos](#) [Perfil](#)

Ejemplo [Cerrar sesión](#)

Desde:

Hasta:

Orden:

El funcionamiento es simple, estableces el rango de fechas de los cuales quieres obtener los datos, y el orden

En las primeras dos tablas veremos las veces jugadas a cada juego y las veces que ingresamos saldo, al ordenarlo esta ultima tabla se ordena segun las fechas

Juegos jugados ▲		Transacciones ▲	
Juego	Veces jugado	Cantidad	Fecha
Slot Machine	2	200	12/6/2025, 15:33:52
Black Jack	2		

Y por último encontramos la tercera tabla

Aquí podremos ver cuando jugamos, a que juego, cuanto apostamos y cuanto obtuvimos

Historial ▲			
Juego	Apostado	Ganado	Fecha
Black Jack	10	20	12/6/2025, 15:38:32
Black Jack	10	10	12/6/2025, 15:38:17
Slot Machine	1	0	12/6/2025, 15:38:10
Slot Machine	1	0	12/6/2025, 15:38:04

Paneles de administración

En caso de estar en posesión de una cuenta con un rol de administrador veras dos apartados mas

Panel con el rendimiento

Aqui podremos obtener informes según las fechas seleccionadas con diferentes datos, la rentabilidad de los juegos, y las horas en las que más se juega a dicho juego en las fechas seleccionadas.

Fecha inicio:

12/05/2025

Fecha fin:

12/06/2025

Rentabilidad por Juego

Juego	Apuestas al juego	Devuelto del juego	Veces jugado	Rentabilidad
Black Jack	2604.00	1863.00	47	741
Slot Machine	1163.00	120.00	27	1063

Horas mas jugadas

Juego	Intervalo más activo	Jugadas
Black Jack	08:15 - 08:29	20
Slot Machine	08:15 - 08:29	10

Panel de usuarios

En el panel de usuarios podremos acceder al historial de cada persona y modificar diferentes parametros

Administrar usuarios					
ID	Username	Email	Balance	Role	Banned until
1	Bertin	arodru351@g.educaand.es	290	Admin	—
2	Alberto	rodriguezaberto14@gmail.com	180	user	—
3	leandro	leandro@correo.com	0	user	—
4	Luna	lunaraaheso@msn.com	152	user	—
5	asd	asdsa@a.es	195	user	2025-01-01 02:02:00
6	PCM	pgommasd@g.educaand.es	0	user	—
7	ZonaGomelos	zonagomelos@gmail.com	0	user	—
8	Ejemplo	ejemplo@correo.com	208	user	—

En caso de querer editar un usuario veremos este popup

Usuario: Bertin

Cambiar Balance:

290

Cambiar Rol:

Admin

Cambiar contraseña:

Banear hasta:

dd/mm/aaaa --:--

Cancel

Save

Aquí podemos modificar el balance, cambiar el rol, su contraseña y banear al usuario para impedir su acceso.