PROGRAMACIÓN WEB 2

Profesor(a):
Carlo Jose Luis Corrales Delgado

Estudiantes:
Yenaro Joel Noa Camino
Armando Steven Cuno Cahuari
Victor Narciso Mamani Anahua
Jorge Luis Mamani Huarsaya

Repositorio GitHub:
https://github.com/ynoacamino/pweb2-final-project

6 de julio, 2024

Backend - Proyecto PWEB

En esta sección, explicaremos detalladamente la implementación del backend para el proyecto.

Descripcion General

El backend del proyecto está construido utilizando Django, un framework de alto nivel para el desarrollo de aplicaciones web en Python.

Estructura del Proyecto

```
server/
-- academia/
   -- migrations/          # Archivos de migraciones de base de datos
      |-- __init__.py      # Inicialización del módulo de migraciones
      |-- 0001_initial.py  # Migración inicial
   -- __init__.py          # Inicialización del módulo academia
   -- admin.py             # Registro de modelos en el admin de Django
   -- apps.py              # Configuración de la aplicación academia
   -- models.py            # Definición de modelos de datos
   -- serializers.py       # Serializadores para la API
   -- tests.py             # Pruebas unitarias para la aplicación
   -- urls.py              # Enrutamiento de URLs específicas de la aplicación
   -- views.py             # Vistas y lógica de controladores
-- media_manager/
   -- migrations/          # Archivos de migraciones de base de datos
      |-- __init__.py      # Inicialización del módulo de migraciones
   -- __init__.py          # Inicialización del módulo media_manager
   -- admin.py             # Registro de modelos en el admin de Django
   -- apps.py              # Configuración de la aplicación media_manager
   -- urls.py              # Enrutamiento de URLs específicas de la aplicación
   -- views.py             # Vistas y lógica de controladores
-- server/
   -- __init__.py          # Inicialización del módulo principal del proyecto
   -- asgi.py              # Configuración para el servidor ASGI
   -- settings.py          # Configuraciones del proyecto Django
   -- urls.py              # Enrutamiento de URLs del proyecto principal
   -- wsgi.py              # Configuración para el servidor WSGI
-- .env.example             # Archivo de ejemplo para las variables de entorno
-- db.sqlite3               # Archivo de base de datos SQLite (para
 ↪  desarrollo)
-- manage.py                # Script principal para gestionar el proyecto
-- requirements.txt         # Lista de dependencias del proyecto
```

La estructura del proyecto está diseñada para facilitar la organización, el mantenimiento y la escalabilidad. A continuación, se presentará dicha estructura.

Flujo de Trabajo

1. El usuario enviara una solicitud HTTP al servidor.
2. La solicitud se enrutara atraves de los archivos 'urls.py' al controlador adecuado en el modulo de vistas.

3. El controlador maneja la logica de la solicitud, interactua con los modelos de datos y utiliza los serializadores si es necesario.

4. La respuesta es devuelta al usuario en formato JSON atraves de la API RESTful

Explicaremos la composición de cada módulo en nuestro proyecto para ilustrar el flujo de trabajo de Django utilizado para el backend.

Módulo Academia

La creación de nuestra aplicación dentro del proyecto backend es fundamental ya que cumple roles esenciales. Cada aplicación en Django actúa como un componente independiente que gestiona una parte específica de la funcionalidad del proyecto.

1. **Módulo models**:
   Definiremos varios modelos en Django para el proyecto relacionado con cursos, usuarios, reseñas, PDFs, entre otros. Cada modelo utilizará diferentes tipos de campos para su representación en la base de datos. Estableceremos relaciones entre estos modelos utilizando 'ManyToManyField' y 'ForeignKey' según sea necesario para capturar las interacciones y la estructura de datos requerida por la aplicación. Las relaciones que guardan los modelos son:
   Relacion de uno a muchos:

   - Curso a Teacher
   - Review a User
   - Review a Curso
   - Section a Curso
   - Pdf a Section

   Relacion de muchos a muchos:

   - Curso a User

```
from django.db import models

class Teacher(models.Model):
teacher_id = models.IntegerField(primary_key=True)
name = models.CharField(max_length=255)
phone_number = models.CharField(max_length=255)
email = models.EmailField()

def _str_(self):
return self.name

class User(models.Model):
user_id = models.IntegerField(primary_key=True)
name = models.CharField(max_length=255)
phone_number = models.CharField(max_length=255)
email = models.EmailField()

def _str_(self):
return self.name

class Curso(models.Model):
curso_id = models.IntegerField(primary_key=True)
name = models.CharField(max_length=255)
```

```
    description = models.TextField()
    teacher = models.ForeignKey(Teacher, on_delete=models.CASCADE)
    users = models.ManyToManyField(User, related_name='usuarios')

    def _str_(self):
    return self.name

    class Review(models.Model):
    comment = models.TextField()
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    curso = models.ForeignKey(Curso, on_delete=models.CASCADE)

    def _str_(self):
    return f'Reseña hecho/a por {self.user.name} hacia el curso
    ↪  {self.curso.name}'

    class Section(models.Model):
    section_id = models.IntegerField(primary_key=True)
    curso = models.ForeignKey(Curso, on_delete=models.CASCADE)
    name = models.CharField(max_length=255)
    description = models.TextField()
    video_url = models.URLField(max_length=200)

    def _str_(self):
    return self.name

    class Pdf(models.Model):
    name = models.CharField(max_length=255)
    url = models.URLField(max_length=200)
    section = models.ForeignKey(Section, on_delete=models.CASCADE)

    def _str_(self):
    return self.name
```

2. **Módulo admin**:
   Será necesario registrar los modelos definidos en el proyecto para que puedan ser gestionados a través del panel de administración de Django. Utilizaremos ‘admin.site.register‘ para registrar estos modelos, lo que facilitará la gestión y el mantenimiento del sistema al permitirnos administrar y modificar los datos de manera eficiente desde la interfaz de administración de Django.

```
    from django.contrib import admin
    from .models import *

    admin.site.register(Teacher)
    admin.site.register(User)
    admin.site.register(Curso)
    admin.site.register(Review)
    admin.site.register(Section)
    admin.site.register(Pdf)
```

3. **Módulo apps**:
   La función de este módulo será configurar la aplicación y proporcionar metadatos sobre ella para

que Django la reconozca y gestione correctamente. Para lograr esto, implementaremos una clase ‘AcademiaConfig‘ que contendrá las configuraciones específicas de la aplicación. Esta clase incluirá información necesaria para referirse y gestionar la aplicación dentro del entorno de Django, como el nombre de la aplicación y configuraciones adicionales que puedan ser requeridas, como configuraciones de permisos o configuraciones de visualización en el panel de administración.

```python
from django.apps import AppConfig

class AcademiaConfig(AppConfig):
default_auto_field = 'django.db.models.BigAutoField'
name = 'academia'
```

4. **Módulo serializers**:
Para emplear Django REST Framework, es necesario definir este módulo. Los serializadores en DRF se encargan de convertir objetos de Django en formatos de datos manipulables, como JSON. Esto es crucial para aplicaciones web donde la transferencia eficiente de datos entre el frontend y el backend es fundamental. Implementaremos un serializador para cada modelo necesario, cada uno recopilando la información del modelo correspondiente. Además, utilizaremos la clase ‘Meta‘ en cada serializador para especificar el modelo y los campos que se deben serializar. Esto asegura que los datos sean presentados de manera coherente y eficiente para su manipulación en la aplicación.

```python
from rest_framework import serializers
from .models import *

class TeacherSerializer(serializers.ModelSerializer):
class Meta:
model = Teacher
fields = ['teacher_id', 'name', 'phone_number', 'email']

class UserSerializer(serializers.ModelSerializer):
class Meta:
model = User
fields = ['user_id', 'name', 'phone_number', 'email']

class CursoSerializer(serializers.ModelSerializer):
class Meta:
model = Curso
fields = ['curso_id', 'name', 'description', 'teacher', 'users']

class ReviewSerializer(serializers.ModelSerializer):
class Meta:
model = Review
fields = ['comment', 'user', 'curso']

class SectionSerializer(serializers.ModelSerializer):
class Meta:
model = Section
fields = ['section_id', 'curso', 'name', 'description',
 ↪  'video_url']

class PdfSerializer(serializers.ModelSerializer):
class Meta:
model = Pdf
fields = ['name', 'url', 'section']
```

5. **Módulo urls**:
   Para emplear Django REST Framework y configurar las URLs de la API, utilizaremos el enrutador 'DefaultRouter'. Este enrutador nos permite gestionar las rutas de la API basadas en las vistas 'ViewSet' de manera automática. Aquí está cómo lo haremos:

   - Crearemos una instancia de 'DefaultRouter'.
   - Registraremos cada 'ViewSet' en el enrutador para que genere automáticamente las rutas correspondientes.

   Esto simplifica significativamente el proceso de definición y mantenimiento de las rutas de la API dentro de nuestra aplicación, asegurando que las URL se gestionen de manera coherente y eficiente.

```python
from rest_framework import routers
from .views import *

router = routers.DefaultRouter()

router.register('api/user', UserViewSet, 'user')
router.register('api/teacher', TeacherViewSet, 'teacher')
router.register('api/curso', CursoViewSet, 'curso')
router.register('api/review', ReviewViewSet, 'review')
router.register('api/section', SectionViewSet, 'section')
router.register('api/pdf', PdfViewSet, 'pdf')

urlpatterns = router.urls
```

6. **Módulo views**:
   Para emplear Django REST Framework y gestionar las operaciones CRUD en los modelos de la base de datos, utilizaremos un conjunto de vistas ('ViewSet'). Cada 'ViewSet' estará encargado de manejar las solicitudes HTTP, interactuar con la base de datos y presentar los datos correspondientes. Aquí está cómo lo haremos:

   - **Definición del ViewSet**
     Crearemos un ViewSet para cada modelo que necesitemos manejar. Este ViewSet contendrá métodos para realizar operaciones CRUD (Create, Retrieve, Update, Delete).
   - **Uso del serializador**
     Indicaremos y utilizaremos el serializador correspondiente para cada instancia del modelo dentro del ViewSet. El serializador se encargará de convertir los objetos de Django en formato de datos manipulable, como JSON, para las respuestas de la API.
   - **Autenticacion y permisos**
     Configuraremos el ViewSet para que no requiera autenticación ni permisos especiales para acceder a las operaciones CRUD. Esto se hace configurando adecuadamente las clases de autenticación y permisos en la vista.

   Este enfoque nos permite manejar de manera eficiente las operaciones sobre los modelos de la base de datos a través de una API RESTful, asegurando que los datos sean presentados y manipulados de manera segura y coherente.

```python
from django.shortcuts import render
from rest_framework import viewsets, permissions
from .models import *
from .serializers import *

class UserViewSet(viewsets.ModelViewSet):
queryset = User.objects.all()
serializer_class = UserSerializer
permission_classes = [permissions.AllowAny]

class TeacherViewSet(viewsets.ModelViewSet):
queryset = Teacher.objects.all()
serializer_class = TeacherSerializer
permission_classes = [permissions.AllowAny]

class CursoViewSet(viewsets.ModelViewSet):
queryset = Curso.objects.all()
serializer_class = CursoSerializer
permission_classes = [permissions.AllowAny]

class ReviewViewSet(viewsets.ModelViewSet):
queryset = Review.objects.all()
authentication_classes = [permissions.IsAuthenticated]
serializer_class = ReviewSerializer

class SectionViewSet(viewsets.ModelViewSet):
queryset = Section.objects.all()
serializer_class = SectionSerializer
permission_classes = [permissions.AllowAny]

class PdfViewSet(viewsets.ModelViewSet):
queryset = Pdf.objects.all()
serializer_class = PdfSerializer
permission_classes = [permissions.AllowAny]
```

Módulo Server

En nuestro proyecto de Django, será necesario ajustar las configuraciones para definir el comportamiento y la funcionalidad de la aplicación de manera óptima. Esto incluye la configuración de aspectos como la base de datos, la autenticación, las rutas de la API, y otros parámetros importantes que afectan cómo se ejecuta y se comporta la aplicación.

1. **Módulo settings**:
   Las configuraciones que establezcamos para el backend son esenciales para definir el comportamiento y la funcionalidad de nuestra aplicación.

   - Será necesario definir las aplicaciones instaladas para que sean reconocidas por el proyecto. Esto incluye tanto las aplicaciones propias del proyecto como las dependencias externas, como Django REST Framework y sus extensiones.

```python
INSTALLED_APPS = [
'media_manager',
'academia',
```

```
'django_extensions',
'rest_framework',
...
]
```

- La base de datos que utiliza el proyecto de manera predeterminada en Django es SQLite. Esta base de datos es adecuada para el desarrollo y pruebas debido a su configuración simple y su facilidad de uso.

```
DATABASES = {
        'default': {
                'ENGINE':
                ↪  'django.db.backends.sqlite3',
                'NAME': BASE_DIR / 'db.sqlite3',
        }
}
```

2. **Modulo urls**:
   En este archivo definiremos la configuración de las URLs del proyecto Django. Esto especificará las rutas principales del proyecto y cómo se deben enrutar las solicitudes a las aplicaciones correspondientes. Al organizar las URLs de esta manera, mantenemos la configuración modular y manejable, lo que facilita el mantenimiento y la escalabilidad del proyecto.

```
urlpatterns = [
path('admin/', admin.site.urls),
path('media_manager/', include('media_manager.urls')),
path('academia/', include('academia.urls')),
]
```

Módulo Media-Manager

Se desarrollará una aplicación dedicada al manejo de archivos multimedia, con la capacidad de gestionar la subida, almacenamiento y organización de estos archivos hacia la plataforma de Cloudinary. Esta integración asegurará que el proyecto maneje de manera eficiente los recursos multimedia, facilitando la gestión y optimización de los activos multimedia utilizados en la aplicación.

1. **Modulo apps**:
   La clase 'MediaManagerConfig' definirá la configuración específica de la aplicación, incluyendo cómo se manejan los campos y el nombre de la aplicación. Está diseñada para inicializar y gestionar adecuadamente estos aspectos dentro del entorno de Django.

```
class MediaManagerConfig(AppConfig):
default_auto_field = 'django.db.models.BigAutoField'
name = 'media_manager'
```

2. **Modulo urls**:
   En este caso, la configuración de URLs apuntará directamente a una vista que ejecutará la función upload-route para manejar la solicitud y devolver la respuesta correspondiente. Aquí te muestro cómo podrías estructurar esto en el archivo urls.py de tu aplicación multimedia:

```
urlpatterns = [
path('upload/', upload_route, name='upload')
]
```

3. **Modulo views**:
En esta vista de Django estara encargado de manejar la carga de archivos usando Cloudinary.

   - Para comenzar con la configuración necesaria en Django para manejar la subida de archivos multimedia y la integración con servicios como Cloudinary, necesitaremos realizar algunas importaciones y configuraciones clave.
     - **from django.http import HttpResponse:** Importa HttpResponse de django.http, que se utiliza para devolver respuestas HTTP desde la vista.
     - **from dotenv import load-dotenv:** Importa load-dotenv de dotenv, que se usa para cargar variables de entorno desde un archivo .env.
     - **load-dotenv():** Llama a la función load-dotenv() para cargar las variables de entorno definidas en un archivo .env. Esto es útil para manejar configuraciones sensibles como claves de API de Cloudinary de manera segura.

```
from django.http import HttpResponse

from dotenv import load_dotenv
load_dotenv()
```

   - Después configuraremos adecuadamente Cloudinary. Para esto, es necesario importar la biblioteca de Cloudinary y especificar que se debe utilizar una conexión segura para todas las operaciones relacionadas con el almacenamiento y gestión de archivos multimedia.

```
import cloudinary
import cloudinary.uploader

import json
config = cloudinary.config(secure=True)
```

   - Ahora, a través de una funcion llamada ‘uploadRoute‘, nos encargaremos de manejar las solicitudes POST que llegan a la URL definida para la subida de archivos.
     - Primero verificara si la solicitud es POST
     - Luego extraera el archivo enviado en la solicitud POST desde request.FILES esperando los nombres del campo.
     - Llamaremos a la funcion uploadFile con el archivo extraido y devuela la respuesta que esta funcion genera.
     - Si la solicitud no es POST, devuelve una respuesta HTTP con el mensaje .ºnly post method allowed”.

```
def upload_route(request):
if request.method == 'POST':
file = request.FILES['file']
return uploadFile(file)
return HttpResponse("Only post method allowed")
```

- Luego, mediante la función llamada ‘uploadFile‘, se procede a cargar los archivos multimedia utilizando la API de Cloudinary.
  Primero, esta función cargará el archivo ‘file‘ a Cloudinary como un recurso automático. Luego, devolverá un diccionario con los datos de la carga. Finalmente, la función responderá con una respuesta HTTP que contiene estos datos de carga serializados en formato JSON.

```
def uploadFile(file):
upload_data = cloudinary.uploader.upload(file,
↪  resource_type = "auto")
return HttpResponse(json.dumps(upload_data))
```

Los Componentes de la aplicación

El directorio `components` contiene todos los componentes reutilizables que forman la base de la interfaz de usuario de la aplicación. Estos componentes están organizados en subdirectorios según su funcionalidad y uso específico dentro de la aplicación , la cual esta estructurado de la siguiente forma .

Estrucutra de la carpeta components

- **components/icons**: Este subdirectorio almacena todos los íconos utilizados en la aplicación, organizados por categorías como certificaciones, pilares de la empresa y redes sociales. Incluye también el logo principal de la aplicación.
- **components/pages**: Este subdirectorio agrupa los componentes específicos de las diferentes páginas de la aplicación. Incluye componentes para páginas de cursos, la página principal y elementos globales como el encabezado y el pie de página.
- **components/providers**: Este subdirectorio contiene componentes proveedores que gestionan aspectos globales de la aplicación, como la configuración del tema.
- **components/ui**: Este subdirectorio contiene componentes reutilizables de la interfaz de usuario, tales como botones, tarjetas, carruseles y menús desplegables. Estos componentes son esenciales para construir la interfaz gráfica de la aplicación.

Components/icons

Este subdirectorio almacena todos los íconos utilizados en la aplicación, organizados por categorías. Incluye íconos para certificaciones, pilares de la empresa y redes sociales, así como el logo principal de la aplicación.

components/iconscertifications

Esta categoría contiene íconos relacionados con certificaciones específicas de la aplicación.

- **Icon1.tsx**: Ícono para certificación tipo 1.
- **Icon2.tsx**: Ícono para certificación tipo 2.
- **Icon3.tsx**: Ícono para certificación tipo 3.

components/iconspillars

Esta categoría contiene íconos que representan los pilares fundamentales de la empresa.

- **Icon1.tsx**: Ícono para pilar 1.
- **Icon2.tsx**: Ícono para pilar 2.
- **Icon3.tsx**: Ícono para pilar 3.

components/iconssocial

Esta categoría contiene íconos de las redes sociales utilizadas en la aplicación.

- **FacebookLogo.tsx**: Ícono para Facebook.
- **InstagramLogo.tsx**: Ícono para Instagram.
- **TiktokLogo.tsx**: Ícono para TikTok.
- **TwitterLogo.tsx**: Ícono para Twitter.

Logo.tsx

Este componente es el logo principal de la aplicación.

Components/providers

Este subdirectorio contiene componentes que proporcionan contexto y servicios globales a la aplicación. Estos componentes están diseñados para gestionar aspectos globales y compartir datos entre diferentes partes de la aplicación.

- **ThemeProvider.tsx**: Este componente utiliza el `NextThemesProvider` de la biblioteca `next-themes` para manejar el tema de la aplicación. Permite la aplicación de un tema predeterminado, la adaptación al sistema del usuario y la desactivación de transiciones en el cambio de tema. El componente envuelve a los hijos en el proveedor de temas, facilitando el control del tema en toda la aplicación.

```tsx
import { ThemeProvider as NextThemesProvider } from 'next-themes';

export function ThemeProvider({ children }: { children: React.ReactNode }) {
  return (
    <NextThemesProvider
      attribute="class"
      defaultTheme="system"
      enableSystem
      disableTransitionOnChange
    >
      {children}
    </NextThemesProvider>
  );
}
```

Components/ui

Este subdirectorio contiene componentes reutilizables de la interfaz de usuario que son fundamentales para construir la apariencia y funcionalidad de la aplicación.

- **button.tsx**: Define un componente de botón reutilizable con variantes estilísticas (por ejemplo, default, destructive, outline, secondary, ghost, link) y tamaños (default, sm, lg, icon). Utiliza la librería class-variance-authority para manejar variantes de estilos y la biblioteca @radix-ui/react-slot para permitir el uso de componentes personalizados como hijos.
- **card.tsx**: Contiene un conjunto de componentes relacionados con la presentación de tarjetas. Card es el contenedor principal, mientras que CardHeader, CardTitle, CardDescription, CardContent y CardFooter estructuran diferentes partes de la tarjeta para organizar el contenido de forma clara y estilizada.
- **carousel.tsx**: Implementa un carrusel utilizando embla-carousel-react. El componente Carousel maneja la lógica del carrusel, incluyendo la navegación y la gestión del estado de desplazamiento. CarouselContent, CarouselItem, CarouselPrevious, y CarouselNext son subcomponentes para el contenido del carrusel, los ítems individuales y los botones de navegación, respectivamente. Utiliza el contexto de React para manejar el estado y las acciones del carrusel.
- **dropdown-menu.tsx**: Proporciona un conjunto de componentes para menús desplegables utilizando @radix-ui/react-dropdown-menu. Incluye DropdownMenu, DropdownMenuTrigger, DropdownMenuContent, DropdownMenuItem, y otros para construir menús complejos con soporte para submenús, casillas de verificación, y opciones de radio. Estiliza estos componentes para una apariencia consistente.

- **input.tsx**: Define un componente de entrada de datos estilizado. Permite la personalización del tipo de entrada y aplica estilos consistentes para el tamaño, el borde y los estados de enfoque o desactivado.
- **label.tsx**: Implementa un componente de etiqueta utilizando @radix-ui/react-label, estilizado con la librería class-variance-authority. Este componente se utiliza para asociar texto descriptivo con campos de formulario o elementos interactivos, con soporte para personalización de estilos.
- **sheet.tsx**: Define un componente de hoja modal utilizando @radix-ui/react-dialog. Incluye Sheet, SheetTrigger, SheetClose, SheetPortal, y SheetContent para manejar la presentación de un panel modal deslizante con varios efectos de entrada/salida. SheetHeader, SheetFooter, SheetTitle, y SheetDescription estructuran el contenido dentro del modal.
- **ThemeToggle.tsx**: Implementa un componente para alternar entre temas claros y oscuros en la aplicación. Utiliza la biblioteca next-themes para gestionar el tema actual y permite a los usuarios cambiar entre los temas utilizando botones con iconos de sol y luna.

Components/pages

Los componentes que definen y estructuran el contenido específico de las diferentes páginas y secciones de la aplicación. Cada subdirectorio dentro de components/pages se dedica a una sección particular de la aplicación, como cursos, páginas de inicio o elementos globales. Los componentes en esta carpeta están diseñados para representar y organizar el contenido y la funcionalidad de cada página de manera modular y reutilizable, facilitando así el mantenimiento y la expansión de la interfaz de usuario de la aplicación.

- **components/pages/home/**
- **src/components/pages/global**
- **src/components/pages/courses**
- **src/components/pages/course**

Componentes de components/pages/home/

Se encuentran los componentes específicos para la página de inicio de la aplicación. Esta carpeta contiene componentes diseñados para presentar secciones clave en la página principal, como los pilares de la plataforma y el título principal.

- `Pillars.tsx`:
  - Define una sección que presenta los pilares o fundamentos de la aplicación.
  - Utiliza un conjunto de iconos y etiquetas para ilustrar los conceptos clave.
  - Los iconos se importan de la carpeta de iconos y se asocian con etiquetas descriptivas como Aprende, Crea, y Comparte.
  - Cada pilar se representa dentro de un contenedor estilizado con una imagen del icono y un botón para la acción.

```
import Icon1 from '@/components/icons/pillars/Icon1';
import Icon2 from '@/components/icons/pillars/Icon2';
import Icon3 from '@/components/icons/pillars/Icon3';
import { Button } from '@/components/ui/button';
const UTILITIES = [
  {
    Icon: Icon1,
    label: 'Aorende',
  },
```

```
  {
    Icon: Icon2,
    label: 'Crea',
  },
  {
    Icon: Icon3,
    label: 'Comparte',
  },
];
export default function Pillars() {
  return (
    <section className="w-full max-w-6xl flex flex-col items-center justify-start gap-10 px-6 my-40">
      <h1 className="text-4xl font-bold text-center">Nuestros pilares</h1>
      <div className="flex gap-10 justify-around items-center flex-wrap w-full max-w-4xl">
        {UTILITIES.map(({ Icon, label }) => (
          <div
            key={crypto.randomUUID()}
            className="bg-card rounded-md py-6 px-14 flex flex-col items-center justify-center gap-4 shadow-md"
          >
            <Icon className="w-28 h-28 m-4" />
            <span className="text-2xl font-semibold">{label}</span>
            <Button variant="secondary" size="icon" className="mt-4">
              -&gt;
            </Button>
          </div>
        ))}
      </div>
    </section>
  );
}
```

- `Title.tsx`:
  - Presenta el título principal de la página de inicio.
  - Incluye un encabezado que destaca el objetivo principal de la plataforma y un subtítulo que refuerza el mensaje central.
  - Ofrece botones para acciones adicionales como "Ver mász Çomienza ahora".
  - Está diseñado para captar la atención del usuario y proporcionar información introductoria sobre la plataforma.

```
import { Button } from '@/components/ui/button';
export default function Title() {
  return (
    <section className="w-full max-w-6xl flex flex-col items-center justify-center gap-8 my-40">
      <h1 className="w-full flex flex-col items-center justify-center font-bold">
        <span className="text-5xl text-center">Aprende a programar y desarrollar</span>
        <span className="text-6xl text-primary text-center">con Learning</span>
```

```
      </h1>
      <p className="w-full max-w-sm text-center">
        la plataforma que te ayuda a mejorar tus habilidades en
↪  programación.
      </p>
      <div className="flex items-center gap-4 text-lg">
        <Button variant="secondary" size="lg">
          Ver mas →
        </Button>
        <Button size="lg">
          Comienza ahora
        </Button>
      </div>
    </section>
  );
}
```

Componentes de components/pages/global/

Contiene componentes que proporcionan funcionalidad y estructura global para la aplicación. Estos componentes incluyen elementos comunes y fundamentales, como encabezados, pies de página, y funciones de autenticación y perfil de usuario.

- `Access.tsx`:
  - Controla el acceso a la aplicación mostrando el componente de inicio de sesión (`Login`) si el usuario no está autenticado. Si el usuario está autenticado, muestra el componente de perfil (`Profile`) con la información del usuario.
  - Usa la función `getServerSession` para verificar la sesión del usuario y renderiza el componente correspondiente basado en el estado de autenticación

```
import { authOptions } from '@/app/api/auth/[...nextauth]/authOptions';
import { getServerSession } from 'next-auth';
import Login from './Login';
import Profile from './Profile';
export default async function Access() {
  const session = await getServerSession(authOptions);
  if (!session) {
    return (
      <Login />
    );
  }
  return (
    <Profile
      name={session.user?.name || ''}
      email={session.user?.email || ''}
      image={session.user?.image || ''}
    />
  );
}
```

- `BrandTitle.tsx`:

- Muestra el nombre de la marca acompañado de un logotipo. Es un componente simple que define el estilo del título de la aplicación.
- Utiliza el componente `Logo` para incluir un ícono y estiliza el texto de la marca.

```
import Logo from '@/components/icons/Log
export default function BrandTitle() {
  return (
    <div className="text-primary text-4xl font-bold flex gap-1">
      <Logo className="w-10 h-10" />
      <span>
        earning
      </span>
    </div>
  );
}
```

- `Footer.tsx`:
  - Se utilizan componentes íconos específicos para cada red social, importados desde rutas designadas.
  - Cada ícono está vinculado a la página de la red social correspondiente mediante un enlace (`href`).
  - Se incluye un componente `BrandTitle` que muestra el nombre de la marca.

```
import FacebookLogo from '@/components/icons/social/FacebookLogo';
import InstagramLogo from '@/components/icons/social/InstagramLogo';
import TwitterLogo from '@/components/icons/social/TwitterLogo';
import TiktokLogo from '@/components/icons/social/TiktokLogo';
import BrandTitle from './BrandTitle
const SOCIALS = [
  {
    Icon: FacebookLogo,
    href: 'https://www.facebook.com/learning',
    label: 'Facebook',
  },
  {
    Icon: InstagramLogo,
    href: 'https://www.instagram.com/learning',
    label: 'Instagram',
  },
  {
    Icon: TwitterLogo,
    href: 'https://www.twitter.com/learning',
    label: 'Twitter',
  },
  {
    Icon: TiktokLogo,
    href: 'https://www.tiktok.com/@learning',
    label: 'TikTok',
  },
];
export default function Footer() {
  return (
```

```
    <footer className="w-full border-t border-foreground/10 px-6 flex justify-between items-start py-6">
      <div className="flex flex-col gap-4">
        <BrandTitle />
        <span>
          © 2024 Learning. Todos los derechos reservados.
        </span>
      </div>
      <ul className="flex gap-6 items-center justify-center">
        {
          SOCIALS.map(({ Icon, href, label }) => (
            <li key={label}>
              <a href={href}>
                <Icon className="w-10 h-auto" />
                <span className="sr-only">{label}</span>
              </a>
            </li>
          ))
        }
      </ul>
    </footer>
  );
}
```

- `Header.tsx`:
  - Define la cabecera de la aplicación, incluyendo el logotipo, enlaces de navegación y componentes para el acceso de usuario y el cambio de tema.
  - Incluye un logotipo y enlaces de navegación que se renderizan en una barra superior, y utiliza los componentes `Access` y `ThemeToggle` para funcionalidades adicionales.

```
import Logo from '@/components/icons/Logo';
import { ThemeToggle } from '@/components/ui/ThemeToggle';
import Link from 'next/link';
import Access from './Acces
export function Header() {
  const LINKS = [
    {
      label: 'Home',
      path: '/',
    },
    {
      label: 'Cursos',
      path: '/cursos',
    },
    {
      label: 'Planes',
      path: '/planes',
    },
  ];
  return (
    <header className="w-full px-6 flex justify-center items-center sticky top-0 backdrop-blur-md bg-neutral-100/40 dark:bg-slate-800/40 z-50">
      <div className="w-full max-w-6xl py-4 flex justify-between">
```

```
        <Link href="/" className="flex items-center gap-1 text-3xl font-bold text-primary">
          <Logo className="w-10 h-10" />
          <span>
            earning
          </span>
        </Li
        <nav className="flex gap-6 items-center">
          {
            LINKS.map(({ label, path }) => (
              <Link key={label} href={path} className="hover:underline">
                {label}
              </Link>
            ))
          }
          <Access />
          <ThemeToggle />
        </nav>
      </div>
    </header>
  );
}
```

- `Login.tsx`:
  - Proporciona un modal para el inicio de sesión de usuario. Permite el acceso mediante Google, Github o con correo electrónico y contraseña.
  - Utiliza el componente `Sheet` para crear el modal, y dentro del modal incluye botones para iniciar sesión con diferentes métodos y un formulario para credenciales.

```
'use client';

import { Button } from '@/components/ui/button';
import { Input } from '@/components/ui/input';
import { Label } from '@/components/ui/label';
import {
  Sheet,
  SheetContent,
  SheetDescription,
  SheetHeader,
  SheetTitle,
  SheetTrigger,
} from '@/components/ui/sheet';
import { signIn } from 'next-auth/react';
import BrandTitle from './BrandTitle';

export default function Login() {
  return (
    <Sheet key={1}>
      <SheetTrigger asChild>
        <Button>
          Login
        </Button>
```

```
      </SheetTrigger>
      <SheetContent side="right">
        <SheetHeader>
          <SheetTitle>
            <BrandTitle />
          </SheetTitle>
          <SheetTitle>
            <h2 className="text-3xl font-bold">
              Registrate
            </h2>
          </SheetTitle>
          <SheetDescription>
            Ingresar o crear una cuenta con:
          </SheetDescription>
        </SheetHeader>
        <div className="grid gap-4 py-4">
          <div aria-label="button" onClick={() => signIn('google')}
↪   className="w-full rounded-md py-4 px-6 text-center text-xl font-semibold
↪   border border-input hover:bg-accent transition-colors duration-75
↪   hover:cursor-pointer">
            Google
          </div>
          <div aria-label="button" onClick={() => signIn('github')}
↪   className="w-full rounded-md py-4 px-6 text-center text-xl font-semibold
↪   border border-input hover:bg-accent transition-colors duration-75
↪   hover:cursor-pointer">
            Github
          </div>
          <div className="w-full rounded-md py-4 px-6 text-center text-xl
↪   font-semibold border border-input hover:bg-accent transition-colors
↪   duration-75 hover:cursor-pointer">
            Google
          </div>
          <div className="flex flex-col gap-4">
            <span className="w-full text-muted-foreground text-sm">O usa tu
↪   correo electronico:</span>
            <form className="grid gap-4">
              <div>
                <Label htmlFor="email">Correo electronico</Label>
                <Input type="email" id="email" />
              </div>
              <div>
                <Label htmlFor="password">Contraseña</Label>
                <Input type="password" id="password" />
              </div>
              <Button type="submit">Registrarse</Button>
            </form>
          </div>
        </div>
      </SheetContent>
    </Sheet>
  );
}
```

- `Profile.tsx`:
  - Muestra un modal de perfil con la información del usuario, incluyendo opciones para acceder a diferentes secciones como cursos y perfil, y un botón para cerrar sesión.
  - Similar a `Login.tsx`, utiliza el componente `Sheet` para el modal. Muestra información del usuario y proporciona botones para diferentes acciones relacionadas con el perfil del usuario.

```tsx
'use client';
import { Button } from '@/components/ui/button';
import {
  Sheet,
  SheetContent,
  SheetDescription,
  SheetHeader,
  SheetTitle,
  SheetTrigger,
} from '@/components/ui/sheet';
import Image from 'next/image';
import { signOut } from 'next-auth/react';
import BrandTitle from './BrandTitle';
export default function Profile(
  { name, image, email }:
  { name: string, image: string, email: string },
) {
  const wordInName = name?.split(' ');
  return (
    <Sheet key={crypto.randomUUID()}>
      <SheetTrigger asChild>
        <Button>
          Perfil
        </Button>
      </SheetTrigger>
      <SheetContent side="right">
        <SheetHeader>
          <SheetTitle>
            <BrandTitle />
          </SheetTitle>
          <SheetTitle>
            <h2 className="text-3xl font-bold flex items-center">
              <span className="text-primary">
                Hi,
                {' '}
              </span>
              <span>
                {`${wordInName[0]} ${wordInName[1]}` || email}
              </span>
              <Image
                src={image}
                alt="profile image"
                width={50}
                height={50}
                className="rounded-full ml-6"
              />
            </h2>
          </SheetTitle>
```

```
          <SheetDescription>
            Un gusto tenerte de vuelta, recuerda que puedes seguir aprendiendo con nosotros.
          </SheetDescription>
        </SheetHeader>
        <div className="grid gap-4 py-4">
          <div className="w-full rounded-md py-4 px-6 text-center text-xl font-semibold border border-input hover:bg-accent transition-colors duration-75 hover:cursor-pointer">
            Mis cursos
          </div>
          <div className="w-full rounded-md py-4 px-6 text-center text-xl font-semibold border border-input hover:bg-accent transition-colors duration-75 hover:cursor-pointer">
            Explorar
          </div>
          <div className="w-full rounded-md py-4 px-6 text-center text-xl font-semibold border border-input hover:bg-accent transition-colors duration-75 hover:cursor-pointer">
            Facturacion
          </div>
          <div className="w-full rounded-md py-4 px-6 text-center text-xl font-semibold border border-input hover:bg-accent transition-colors duration-75 hover:cursor-pointer">
            Mi perfil
          </div>
          <Button onClick={() => signOut()}>Cerrar sesion</Button>
        </div>
      </SheetContent>
    </Sheet>
  );
}
```

Componentes de components/pages/courses/

El directorio contiene componentes de React utilizados para presentar información sobre cursos y certificaciones. Cada archivo en este directorio contribuye a una parte específica de la interfaz de usuario relacionada con cursos y certificaciones.

- `Certifications.tsx`:
  - Este componente muestra una lista de certificaciones con íconos y etiquetas. Utiliza una rejilla para organizar las certificaciones de manera atractiva, con cada certificación representada por un ícono y una etiqueta descriptiva. Utiliza el método `map()` de JavaScript para iterar sobre un array de objetos y renderizar cada certificación en un componente `div` estilizado. La clase de Tailwind CSS `grid` se usa para crear una disposición en rejilla que se adapta a diferentes tamaños de pantalla.

```
import Icon1 from '@/components/icons/certifications/Icon1';
import Icon2 from '@/components/icons/certifications/Icon2';
import Icon3 from '@/components/icons/certifications/Icon3';

const CERTIFICATIONS = [
  {
```

```
    Icon: Icon1,
    label: 'Cybersecurity for Beginners from CISCO',
  },
  {
    Icon: Icon2,
    label: 'AWS Certified Solutions Architect',
  },
  {
    Icon: Icon3,
    label: 'Google Cloud Platform Fund',
  },
  {
    Icon: Icon1,
    label: 'Cybersecurity for Beginners from CISCO',
  },
];

export default function Certifications() {
  return (
    <section className="w-full max-w-6xl px-6 flex flex-col gap-10 my-40">
      <h1 className="text-4xl font-bold flex flex-col gap-1 items-center justify-start">
        <span>
          Obten
          {' '}
        </span>
        <span className="text-primary text-5xl">
          certificaciones oficiales
          {' '}
        </span>
        <span>
          de:
        </span>
      </h1>
      <div className="grid grid-cols-1 md:grid-cols-2 gap-10 w-full">
        {
          CERTIFICATIONS.map(({ Icon, label }) => (
            <div key={crypto.randomUUID()} className="w-full bg-card shadow-md rounded-md p-10 flex gap-10 items-center min-h-56">
              <Icon className="w-24 h-auto" />
              <span className="text-xl font-semibold text-center flex-1">{label}</span>
            </div>
          ))
        }
      </div>
    </section>
  );
}
```

- `CourseCard.tsx`:
  - El componente `CourseCard` presenta una tarjeta para un curso, mostrando la imagen del curso, su nombre y una breve descripción. Utiliza el componente `Image` de Next.js para optimizar

y mostrar imágenes, asegurando un rendimiento eficiente. La tarjeta está estructurada con una disposición flexible y un diseño responsivo usando Tailwind CSS, lo que garantiza que el contenido se presente de manera atractiva en diferentes dispositivos.

```tsx
import Image from 'next/image';

export default function CourseCard(
  { name, description, image }:
  { name: string, description: string, image: string },
) {
  return (
    <article className="w-full max-w-sm px-4">
      <div className="w-full flex flex-col items-center justify-start gap-4 rounded-md bg-card h-96 shadow-lg">
        <Image
          src={image}
          alt={name}
          width={300}
          height={200}
          className="rounded-t-md aspect-video w-full"
        />
        <div className="p-4 flex flex-col gap-4">
          <h2 className="text-2xl font-bold text-center">{name}</h2>
          <div className="w-full border border-b border-foreground" />
          <p className="text-center">
            {description}
          </p>
        </div>
      </div>
    </article>
  );
}
```

- `CoursesList.tsx`:
  - Este componente utiliza un carrusel para mostrar múltiples tarjetas de curso (`CourseCard`). El carrusel permite a los usuarios deslizarse entre las tarjetas de curso, mejorando la experiencia de navegación. Los componentes `Carousel`, `CarouselContent`, `CarouselItem`, `CarouselPrevious`, y `CarouselNext` proporcionan funcionalidades para la navegación y el diseño del carrusel. Utiliza un array para mapear los datos de los cursos a tarjetas individuales, que luego se muestran en un carrusel deslizable.

```tsx
import {
  Carousel,
  CarouselContent,
  CarouselItem,
  CarouselNext,
  CarouselPrevious,
} from '@/components/ui/carousel';
import CourseCard from './CourseCard';

const COURSES = [
  {
    name: 'Curso de JavaScript',
```

```
    description: 'Aprende a programar con JavaScript, el lenguaje de programación más popular.',
    image: 'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
  {
    name: 'Curso de React',
    description: 'Aprende a crear aplicaciones web con React, el framework de JavaScript más popular.',
    image: 'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
  {
    name: 'Curso de Vue',
    description: 'Aprende a crear aplicaciones web con Vue, el framework de JavaScript más popular.',
    image: 'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
  {
    name: 'Curso de JavaScript',
    description: 'Aprende a programar con JavaScript, el lenguaje de programación más popular.',
    image: 'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
  {
    name: 'Curso de React',
    description: 'Aprende a crear aplicaciones web con React, el framework de JavaScript más popular.',
    image: 'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
  {
    name: 'Curso de Vue',
    description: 'Aprende a crear aplicaciones web con Vue, el framework de JavaScript más popular.',
    image: 'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
  {
    name: 'Curso de JavaScript',
    description: 'Aprende a programar con JavaScript, el lenguaje de programación más popular.',
    image: 'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
  {
    name: 'Curso de React',
    description: 'Aprende a crear aplicaciones web con React, el framework de JavaScript más popular.',
    image: 'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
```

```
  {
    name: 'Curso de Vue',
    description: 'Aprende a crear aplicaciones web con Vue, el framework de
↪   JavaScript más popular.',
    image:
↪   'https://res.cloudinary.com/daxt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgydsjx.png
  },
];

export function CoursesList() {
  return (
    <Carousel className="w-full max-w-6xl my-20">
      <CarouselContent>
        {COURSES.map(({ description, image, name }) => (
          <CarouselItem className="md:basis-1/2 lg:basis-1/3"
↪   key={crypto.randomUUID()}>
            <CourseCard
              description={description}
              image={image}
              name={name}
            />
          </CarouselItem>
        ))}
      </CarouselContent>
      <CarouselPrevious />
      <CarouselNext />
    </Carousel>
  );
}
```

- `Info.tsx`:
  - El componente `Info` ofrece una visión general de los cursos disponibles, incluyendo una imagen destacada, un título principal y un párrafo descriptivo. Utiliza el componente `Button` para proporcionar una llamada a la acción que invita a los usuarios a comenzar con los cursos. La estructura utiliza Flexbox para alinear el contenido en una disposición flexible y responsiva, asegurando que la sección se vea bien en diferentes tamaños de pantalla. Tailwind CSS se utiliza para aplicar estilos y garantizar una presentación visual atractiva.

```
import { Button } from '@/components/ui/button';
import Image from 'next/image';

export default function Info() {
  return (
    <section className="w-full max-w-6xl flex flex-col gap-10 my-20
↪   shadow-lg">
      <article className="w-full rounded-md bg-card p-8 flex gap-10 flex-col
↪   lg:flex-row items-center">
        <Image

↪   src="https://res.cloudinary.com/daxt6g3o1/image/upload/v1719160177/jbarsqhcz83x8mp4l27a
          alt="Certificaciones"
          width={400}
```

```
          height={400}
          className="rounded-md aspect-square"
        />
        <div className="flex flex-col flex-1 items-center justify-center gap-8">
          <h1 className="text-3xl md:text-5xl font-bold text-center">
            Conviertete en un profesional
          </h1>
          <p className="text-lg md:text-xl text-center w-full max-w-lg">
            Ofrecemos una amplia gama de cursos impartidos a millones de estudiantes en
            Learning, diseñados para cubrir diversas áreas de conocimiento y adaptarse a
            las demandas del mercado actual.
          </p>
          <Button>
            Comienza ahora
          </Button>
        </div>
      </article>
    </section>
  );
}
```

- `Title.tsx`:
  - Este componente presenta un título prominente y una breve descripción sobre la autoridad de la plataforma en educación profesional. Utiliza un diseño centrado y clases de Tailwind CSS para estilizar el texto y el diseño, destacando el mensaje principal. El componente se centra en proporcionar un encabezado claro y motivador, acompañado de una breve descripción para captar la atención de los usuarios y animarlos a explorar los cursos ofrecidos.

```
export default function Title() {
  return (
    <section className="w-full max-w-6xl flex flex-col items-center justify-center gap-8 my-40">
      <h1 className="w-full flex flex-col items-center justify-center font-bold">
        <span className="text-6xl text-primary text-center">Somos la autoridad</span>
        <span className="text-5xl text-center">en educación profesional en América Latina.</span>
      </h1>
      <p className="w-full max-w-sm text-center">
        Explora nuestros cursos profesionales capaces de cambiar tu vida profesional.
      </p>
    </section>
  );
}
```

Componentes de components/pages/course/

El directorio contiene componentes de React relacionados con la visualización y organización de detalles de cursos en una aplicación web. Estos componentes están diseñados para mostrar información relevante sobre un curso, incluyendo su descripción, cursos relacionados, secciones, y un título destacado. Los componentes utilizan técnicas modernas de desarrollo web, como el diseño responsivo con Tailwind CSS y la optimización de imágenes con Next.js, para proporcionar una experiencia de usuario fluida y atractiva.

- `Description.tsx`:
  - Este componente muestra una sección con la descripción de un curso. Utiliza una estructura flexbox para organizar el contenido en una columna con un título y una descripción proporcionada como propiedad. Las clases de Tailwind CSS `bg-card`, `rounded-md`, `px-8`, y `py-6` se utilizan para aplicar estilos y garantizar una presentación visual atractiva. El componente se centra en mostrar el texto descriptivo de manera clara y ordenada.

```tsx
export default function Description({ description }: { description: string }) {
  return (
    <section className="bg-card rounded-md w-full flex flex-col gap-3 px-8 py-6">
      <h1 className="text-2xl font-semibold">Descripcion: </h1>
      <span className="text-lg">
        {description}
      </span>
    </section>
  );
}
```

- `RelatedCourses.tsx`:
  - El componente `RelatedCourses` presenta una lista de cursos relacionados en una disposición flexible. Utiliza el componente `CourseCard` para mostrar cada curso, proporcionando una imagen, un nombre y una descripción. La función `Array.from()` se utiliza para crear un array de longitud 3, y se mapean los datos a instancias del componente `CourseCard`. Las clases de Tailwind CSS `flex`, `flex-wrap`, y `justify-around` aseguran que los cursos se presenten de manera organizada y responsiva.

```tsx
import CourseCard from '../courses/CourseCard';

export default function RelatedCourses() {
  return (
    <section className="w-full flex flex-col gap-5">
      <h1 className="text-3xl font-bold">
        Cursos relacionados:
      </h1>
      <div className="flex flex-wrap justify-around items-center gap-y-10">
        {Array.from({ length: 3 }).map(() => (
          <CourseCard
            key={crypto.randomUUID()}
            name="Curso de React"
            description="Aprende React desde cero"

            image="https://res.cloudinary.com/dazt6g3o1/image/upload/v1719157975/ecyf3ye0dnn5hbgyds
```

```
          />
        ))}
      </div>
    </section>
  );
}
```

- `Sections.tsx`:
  - El componente `Sections` muestra una lista de secciones de un curso con enlaces. Cada sección se representa como un elemento de lista (`li`) con un enlace (`Link`) que lleva a la URL de la sección. El componente utiliza el ícono `PlayIcon` para indicar que el elemento es interactivo. La función `map()` se usa para iterar sobre un array de secciones y renderizar cada una de ellas. Las clases de Tailwind CSS `hover:bg-foreground/10` y `rounded-md` se utilizan para mejorar la interacción del usuario con los elementos de la lista.

```
import Link from 'next/link';
import { PlayIcon } from 'lucide-react';

export default function Sections({ sections }: { sections: { name:string, url:string }[] }) {
  return (
    <section className="bg-card rounded-md w-full flex flex-col gap-3 px-8 py-6">
      <h1 className="text-2xl font-semibold">Secciones: </h1>
      <ul className="flex flex-col gap-1">
        {sections.map(({ name, url }) => (
          <li key={crypto.randomUUID()}>
            <Link href={url} className="flex gap-2 items-center hover:bg-foreground/10 rounded-md px-4 py-3">
              <PlayIcon />
              <span>
                {name}
              </span>
            </Link>
          </li>
        ))}
      </ul>
    </section>
  );
}
```

- `Title.tsx`:
  - El componente `Title` muestra un título con una imagen destacada y un nombre. Utiliza el componente `Image` de Next.js para optimizar y mostrar la imagen del curso, garantizando un rendimiento eficiente. La imagen se presenta con un diseño de aspecto de video usando la clase `aspect-video`. El título se muestra en un formato prominente usando las clases de Tailwind CSS `text-3xl` y `font-bold`. Este componente está diseñado para destacar la imagen y el nombre del curso en una presentación visualmente atractiva.

```
import Image from 'next/image';

```

```tsx
export default function Title({ image, name }: { image: string, name: string }) {
  return (
    <section className="w-full flex flex-col gap-4">
      <Image
        src={image}
        alt={name}
        width={800}
        height={800}
        className="w-full object-cover aspect-video rounded-md"
      />
      <h1 className="text-3xl font-bold">
        {name}
      </h1>
    </section>
  );
}
```

Visualización de la App

Home - Theme Dark - System

Figura 1: Home - Theme Dark and System

Figura 2: Home - Theme Dark and System

Home - Theme Ligth

Figura 3: Home - Theme Ligth

Figura 4: Home - Theme Ligth

Cursos - Theme Dark - System

Figura 5: Cursos - Theme Dark and System

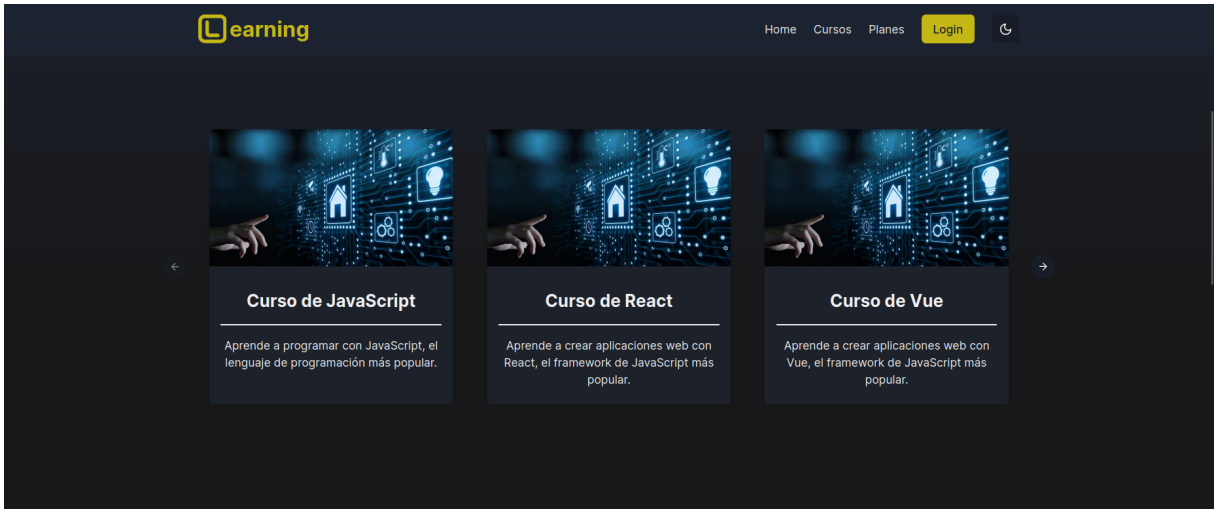

Figura 6: Cursos - Theme Dark and System

Figura 7: Cursos - Theme Dark and System

Figura 8: Cursos - Theme Dark and System

Cursos - Theme Ligth

Figura 9: Cursos - Theme Ligth

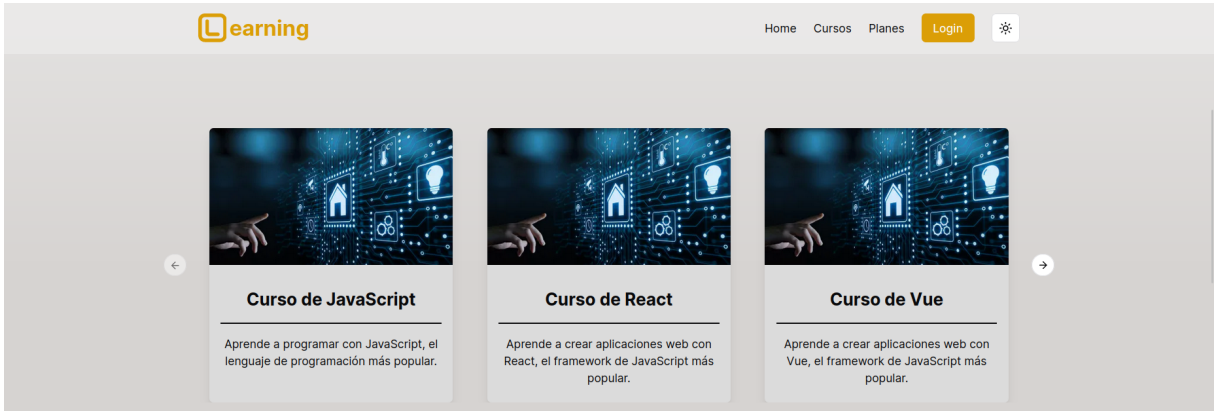

Figura 10: Cursos - Theme Ligth

Figura 11: Cursos - Theme Ligth

Figura 12: Cursos - Theme Ligth