Práctica Gatsby:

El objetivo de la práctica desarrollar un prototipo de página de inicio de un sitio web estático basado en micro frontend, siguiendo la arquitectura JAMSTACK. En concreto se desarrollará un component Gatsby que exponga al usuario información sobre bienes patrimoniales de la ciudad de La Laguna. El componente mostrará información relativa a los antecedentes, el tipo de edificio, una imagen y la ubicación del edificio.

Preparación del entorno de pruebas:

En la fase de desarrollo la información respecto al bien la puedes obtener a través de un servicio de simulación de API REST que facilite la creación de un entorno de pruebas en el desarrollo del prototipo. Existen varias opciones para esto: MockAPI, Beeceptor, Mockable, etc. Un ejemplo con Mockable: https://www.mockable.io/a/#/space/demo7708722

Se ha elaborado con un número reducido de bienes (3), del tipo Arquitectura Civil y Doméstica anterior al s. XX. cuya información se ha obtenido en el Gestor del Patrimonio Cultural del CICOP.

El conjunto de datos sobre los bienes usados en el ejemplo está disponible en el Campus Virtual de la asignatura. Un ejemplo de bien:


```
{
 "nombre": "CASA ALVARADO BRACAMONTE",
 "antecedentes": "El Capitán General de Tenerife don Diego de Alvarado Bracamonte
edificó esta casa entre 1624 y 1631",
 "tipo": {
     "arquitectura": "CIVIL Y DOMÉSICA",
     "época":"Anterior al s. XX"
 },
 "img":
"http://gestorpatrimoniocultural.cicop.com/imgFicha/17_1_2021_0_49_40.png",
 "localizacion":{
     "lat": "28°29'21.2 N",
     "long": "16°18'51.5 W"
     }
}
```


Esta información la debes recuperar mediante llamadas a la API para inyectar la información al componente, en esta práctica esta tarea la desempeña **GraphQL**. En la

página de inicio se distribuirán varias cartas, por ejemplo, podrían ser los 3 componentes con mejores valoraciones, o con cualquier otro criterio que quieras considerar.

Configuración del proyecto:

Será necesario tener instalado node.js, versión posterior a v.18. Con npm instalar Gatsby y su interfaz de comando:

```
npm install -g gatsby-cli
```

Algunos comandos de Gatsby:

```
gatsby --version
gatsby new
gatsby develop
gatsby build
gatsby serve
```

Esta información se puede ampliar en la página de comandos de Gatsby
El comando gatsby new nos permite generar un proyecto básico de Gatsby. Podemos utilizar starters en el sitio de Gatsby. Si no especifcamos ninguno, se crea un sitio básico, Gatsby nos irá preguntando de forma interactiva aspectos relacionados con la configuración de nuestro proyecto.

Tarea 1: Preparación del proyecto

Instala Gatsby
Crea un directorio para el proyecto.
Inicializa Git en el directorio anterior.
Inicializa un proyecto Gatsby.

Estructura de los Componentes: Gatsby -React

Un componente en Gatsby será una función React, que implementaremos con la sintaxis JSX.

- Importar React, ya que Gatsby es un framework basado en React.
- Definir el componente, usaremos la sintaxis JSX. El componente será una función en un fichero `.js`
- Exportar el componente, para usarlo en cualquier página del sitio

```
// Importar React
import * as React from 'react'

/* Definir el componente.*/
const MyComponent = () => {
  return (
```

```
      <h1>Intro Gatsby</h1>
  )
}

/* Exportar el componente */
export default MyComponent
```

Enlazar otro componente

Para utilizar componentes Gatsby en la página debemos usar **`Link.`** Es necesario importar el componente, y especificar a quién enlaza. Si enlazamos páginas externas se debe usar la etiqueta HTML **`<a>.`** Se debe asignar al atributo to la ruta. Gatsby genera rutas basadas en la propia ruta del script.

Ruta de la página index **`/`**

Ruta de la página ejemplo **`/ejemplo`**

```
import * as React from 'react'
import {Link} from 'gatsby'

/* Definir el componente.*/
const IndexPage = () => {
  return (
    <div>
        <h1>Intro Gatsby</h1>
        <p>Ejemplo de la práctica sobre Gatsby</p>
        <Link to="/ejemplo">About</Link>
    </div>
  )
}

/* Exportar el componente */
export default IndexPage
```

Componente enlazado

```
import * as React from 'react'
const Ejemplo = () => {
  return (
    <main>
```

```
        <h1>Página Ejemplo</h1>
      <p>Componenete simple, genera una página enlazada en index.</p>
    </main>
  )
}

export default Ejemplo
```

Tarea 2: Crear un sitio con una página de inicio, y otras 2 páginas adicionales, que se enlazarán en la página de inicio
La reutilización de componentes como micro frontends se realiza en React a través del prop propio de React children. React todo lo que esté dentro de la etiqueta se lo pasa a children. El componente importa al contenedor, y todo su contenido se lo pasa al contenedor. Esta característica es útil para generar un layout que se repita en el sitio. A continuación un componente con un `div` y `h1`

```
import React from 'react'
const Contenedor = ({children}) => {
  return (
    <div>
      <h1>Página Ejemplo</h1>
      {children}
    </div>
  )
}

export default Contenedor
```

Este componente lo utilizamos para incrustar el contenido en otro componente. Por tanto este último debe importar el contenedor.

```
import * as React from 'react'
import Contenedor from './contenedor'
const Comp = () => {
  return (
    <Contenedor>
      <p> Información para incluir en el contenedor a través de children</p>
    </Contenedor>
  )
}
```

```
export default Comp
```

Tarea 3: Crear un layout que se pueda utilizar en las 3 páginas anteriores y modifica el proyecto para utilizarlo.

Uso de plugins:

Los plugins nos permiten instalar y utilizar características adicionales en Gatsby. Se deben instalar y configurar en el fichero: `gatsby-config.js:`

- Instalación del plugin con `npm: npm install plugin-name.` Esto puede requerir la instalación de dependencias.
- Configuración del fichero **`gatsby-config.js`** Este fichero contiene el objeto module.exports que incluye un array plugins, que recogerá los plugins y sus parámetros de configuración:

```
plugins: [
  {
    resolve: "plugin-name",
    options: {
        }
  }, ...]
```

Tarea 4: Usar el componente **`<StaticImage/>`** disponible en el plugin gatsby-plugin-image para incluir una imagen en la página de inicio. Los props básicos serán **`alt`** y **`src`**:
`<StaticImage src =”...” alt =”...”/>`

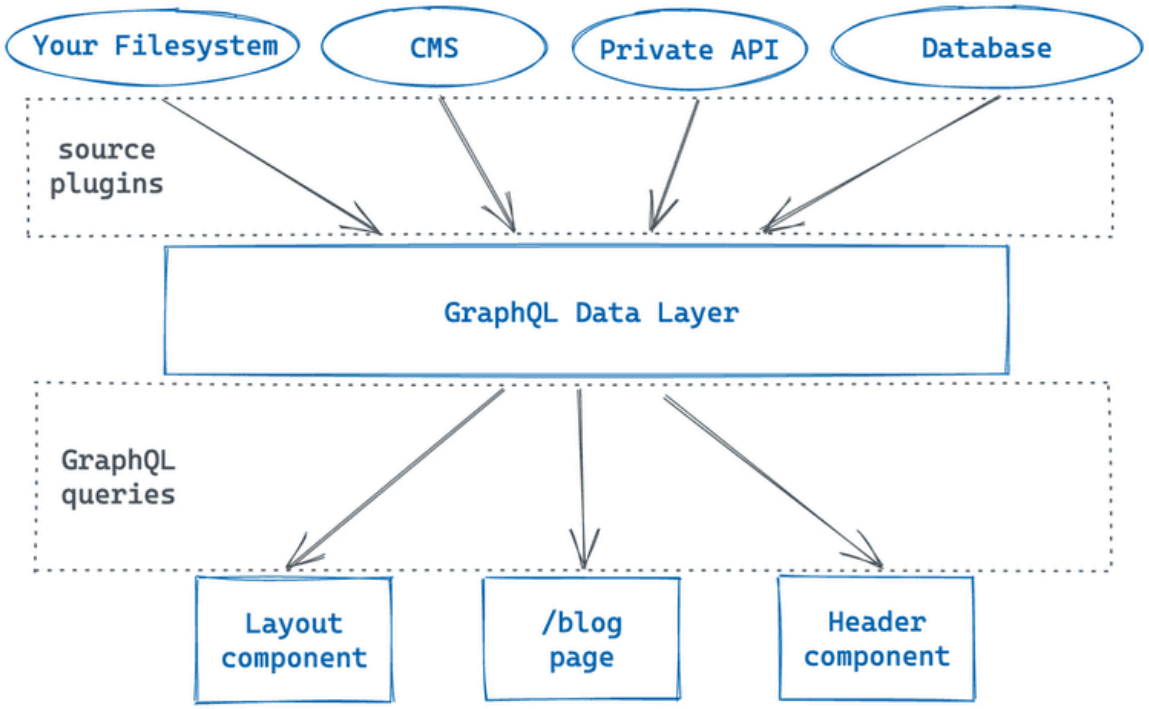
Utilizar estilos:

Al igual que con las imágenes debemos importar los .css. Las clases las utilizaremos con el atributo: className, propio de React.

GraphQL

Gatsby incluye en su capa de datos todos los datos provenientes de cualquier fuente: locales, de APIs, de las páginas, etc. La integración de las múltiples fuentes las realiza mediante GraphQL. Para utilizar los datos en la capa de datos en los componentes haremos consultas a la capa mediante GraphQL. Gatsby nos proporciona la herramienta: GrapiQL

para facilitar la construcción de las consultas, permitiéndonos explorar la capa de datos y seleccionando los elementos que necesitemos.



Se puede usar el hook **`useStaticQuery`** para acceder a los resultados de las consultas. Es necesario importarla. Con este hook podemos invocar la ejecución de consultas a la capa de datos. El resultado será un JSON con una estructura similar a lo que obtenemos con GraphiQL. Es conveniente asignar el resultado a una variable que nos dará acceso a los datos.

```
// Importar React
import * as React from 'react'
import {Link, useStaticQuery, graphql} from 'gatsby'
import Layout from './layout'

/* Definir el componente.*/
const IndexPage = () => {
  const data = useStaticQuery(graphql`query {
    site{
      siteMetadata {
        title}
      }
  }`)
  return (
    <Layout>

        <h1>{data.site.siteMetadata.title}</h1>
        <p>Ejemplo de la práctica sobre Gatsby</p>
    </Layout>
  )
}
export const Head = () => <title>Home</title>
/* Exportar el componente */
export default IndexPage
```

Para incluir los resultados de una consulta en componentes de página usa el **`props data`** que devuelve la consulta. Es similar, pero existen diferencias. En este caso, crearemos enlaces a las restantes páginas usando una consulta en GraphQL que recupere todos los ficheros en una carpeta. La consulta la construimos con la ayuda de GrapiQL:

```
import * as React from 'react'
import {graphql} from 'gatsby'
const Ficheros = ({data}) => {
  return (
    <main>
      <h1>Página Ejemplo</h1>
      <div>
        { data.allFile.nodes.map(node =>
            <p key = {node.name}> {node.name} </p>
        )
        }
      </div>
    </main>
  )
}
export const query = graphql`
query {
    allFile {
      nodes {
        name
      }
    }
  }`
export default Ficheros
```

https://www.gatsbyjs.com/docs