

E-COMMERCE CON MODELO 3D INTERACTIVO

Realizado por

JOSE ÁNGEL DOMÍNGUEZ ESPINACO

Dirigido por

PABLO TRINIDAD MARTÍN-ARROYO

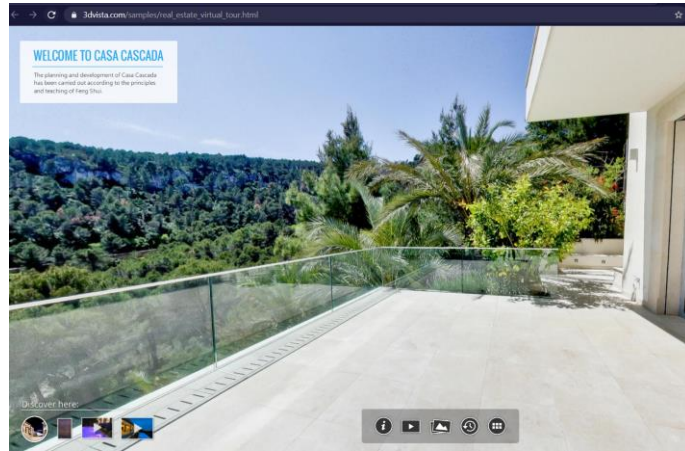
ÍNDICE DE CONTENIDOS

1	CONTEXTO Y OBJETIVOS PRINCIPALES
2	METODOLOGÍA, PLANIFICACIÓN Y COSTES
3	ANÁLISIS DE LAS TECNOLOGÍAS EMPLEADAS
4	DESARROLLO DEL PROYECTO
5	CONCLUSIONES

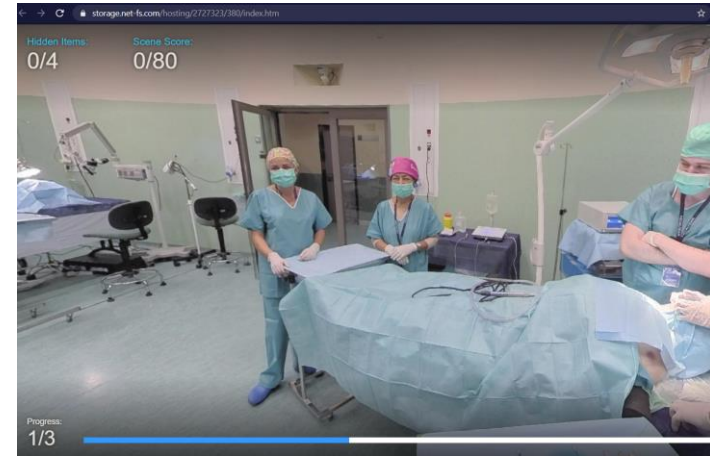
ÍNDICE DE CONTENIDOS

1	CONTEXTO Y OBJETIVOS PRINCIPALES
2	METODOLOGÍA, PLANIFICACIÓN Y COSTES
3	ANÁLISIS DE LAS TECNOLOGÍAS EMPLEADAS
4	DESARROLLO DEL PROYECTO
5	CONCLUSIONES

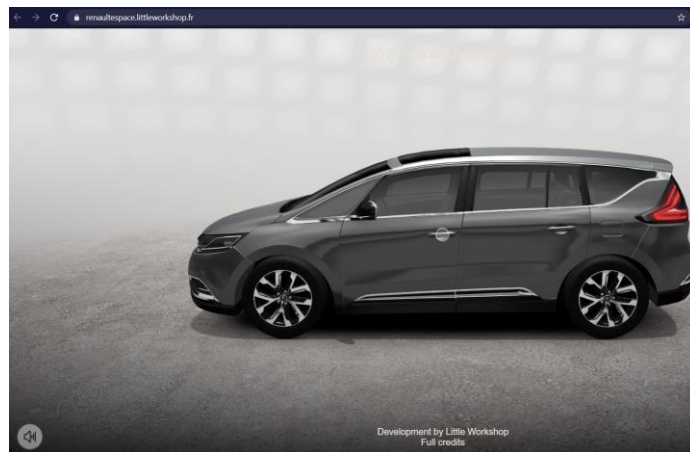
CONTEXTO



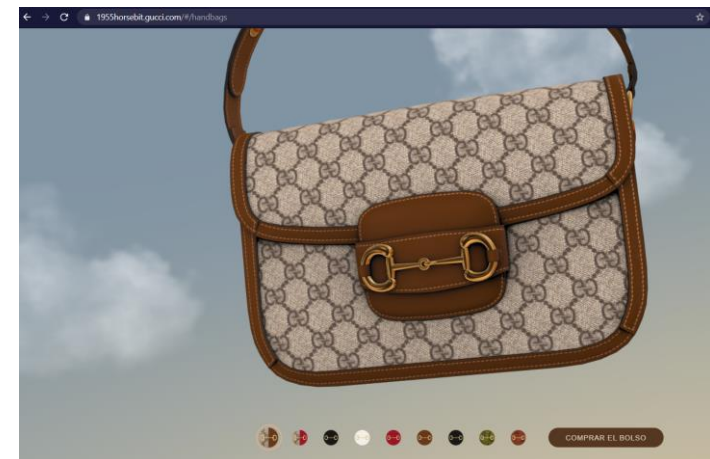
Tour Inmobiliario



Simulaciones virtuales



Concesionarios virtuales



Tiendas virtuales

OBJETIVOS PRINCIPALES

- 01 Analizar el estado actual del e-commerce.
- 02 Analizar las tecnologías actuales existentes para este tipo de aplicaciones.
- 03 Desarrollo de una aplicación web e-commerce que muestre un modelo 3D del producto a vender.
- 04 Explicación de todo el proceso de desarrollo para acercar al lector a la tecnología WebGL.

ÍNDICE DE CONTENIDOS

1	CONTEXTO Y OBJETIVOS PRINCIPALES
2	METODOLOGÍA, PLANIFICACIÓN Y COSTES
3	ANÁLISIS DE LAS TECNOLOGÍAS EMPLEADAS
4	DESARROLLO DEL PROYECTO
5	CONCLUSIONES

METODOLOGÍA



Metodología SCRUM:

- 5 Iteraciones de distinta duración.



git

PLANIFICACIÓN

ITERACIONES

01

Análisis de las tecnologías a usar.

02

Desarrollo de la parte Frontend del proyecto.

03

Desarrollo de la parte Backend del proyecto.

04

Despliegue del proyecto.

05

Elaboración de la documentación del proyecto.

PLANIFICACIÓN

Resumen del proyecto

Fecha de inicio	10/07/2020
Fecha de fin	30/08/2020
Periodicidad de las revisiones	1 semanas
Carga de trabajo semanal	40 horas
Horas totales previstas	296 horas
Horas finales	308 horas

Resumen de iteraciones

Iteración 1	10/07/20 a 15/07/20
Iteración 2	15/07/20 a 01/08/20
Iteración 3	01/08/20 a 11/08/20
Iteración 4	11/08/20 a 23/08/20
Iteración 5	23/08/20 a 30/08/20

Planificado

Resumen de iteraciones

Iteración 1	10/07/20 a 14/07/20
Iteración 2	14/07/20 a 05/08/20
Iteración 3	05/08/20 a 11/08/20
Iteración 4	11/08/20 a 23/08/20
Iteración 5	23/08/20 a 30/08/20

Real

COSTES

Resumen del proyecto

Costes de personal	1.904,14 €
Sueldo bruto	1.451,88 €
Costes sociales	452,26 €
Costes materiales	1.075,85 €
Costes indirectos	297,99 €
TOTAL	3.277,98 €



BOLETÍN OFICIAL DEL ESTADO



Núm. 134

Miércoles 13 de mayo de 2020

Sec. III. Pág. 32833

Categoría profesional	Puestos de trabajo	Salario de referencia grupo (salario 2019+1,8%)
	Oficial Administrativo de primera.	39,24
	Programador/Analista Aplic. Informáticas.	39,24
	Oficial Cualificado de Mantenimiento.	39,24
	Técnico de Ventas.	39,24
	Técnico de Organización.	39,24
	Oficial de primera.	39,24

ÍNDICE DE CONTENIDOS

1	CONTEXTO Y OBJETIVOS PRINCIPALES
2	METODOLOGÍA, PLANIFICACIÓN Y COSTES
3	ANÁLISIS DE LAS TECNOLOGÍAS EMPLEADAS
4	DESARROLLO DEL PROYECTO
5	CONCLUSIONES

ANÁLISIS DE LAS TECNOLOGÍAS



ANÁLISIS DE LAS TECNOLOGÍAS



VENTAJAS

Gratuito

Open Source

Estable

Popular

Gran comunidad colaborativa

Extensible mediante plugins

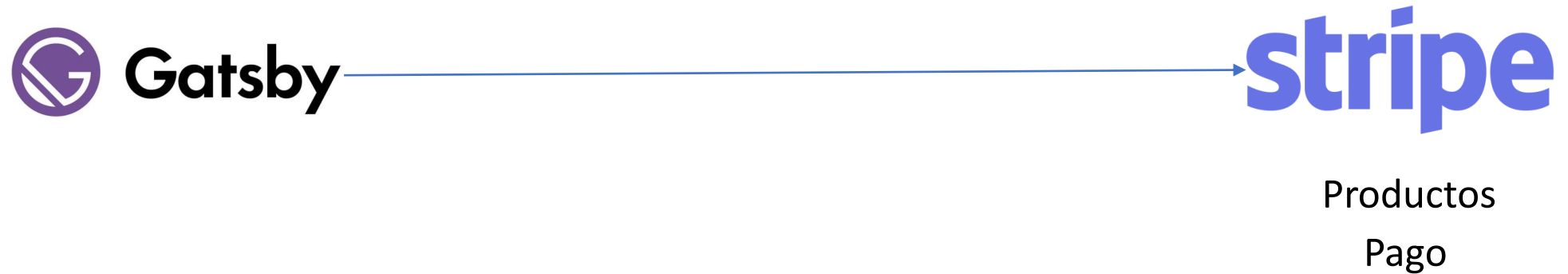
INCONVENIENTES

Hacking

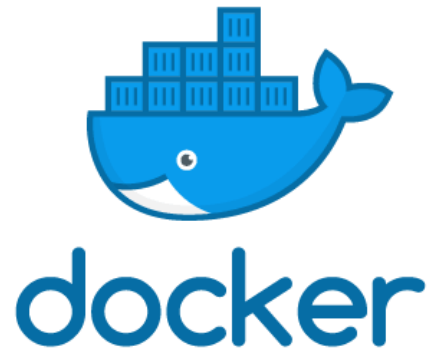
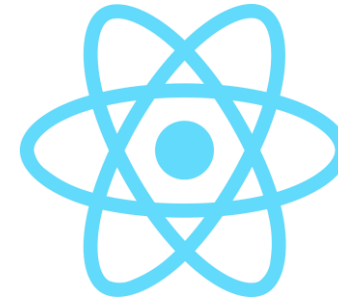
Consume muchos recursos

Aprender Wordpress y PHP

ANÁLISIS DE LAS TECNOLOGÍAS



ANÁLISIS DE LAS TECNOLOGÍAS

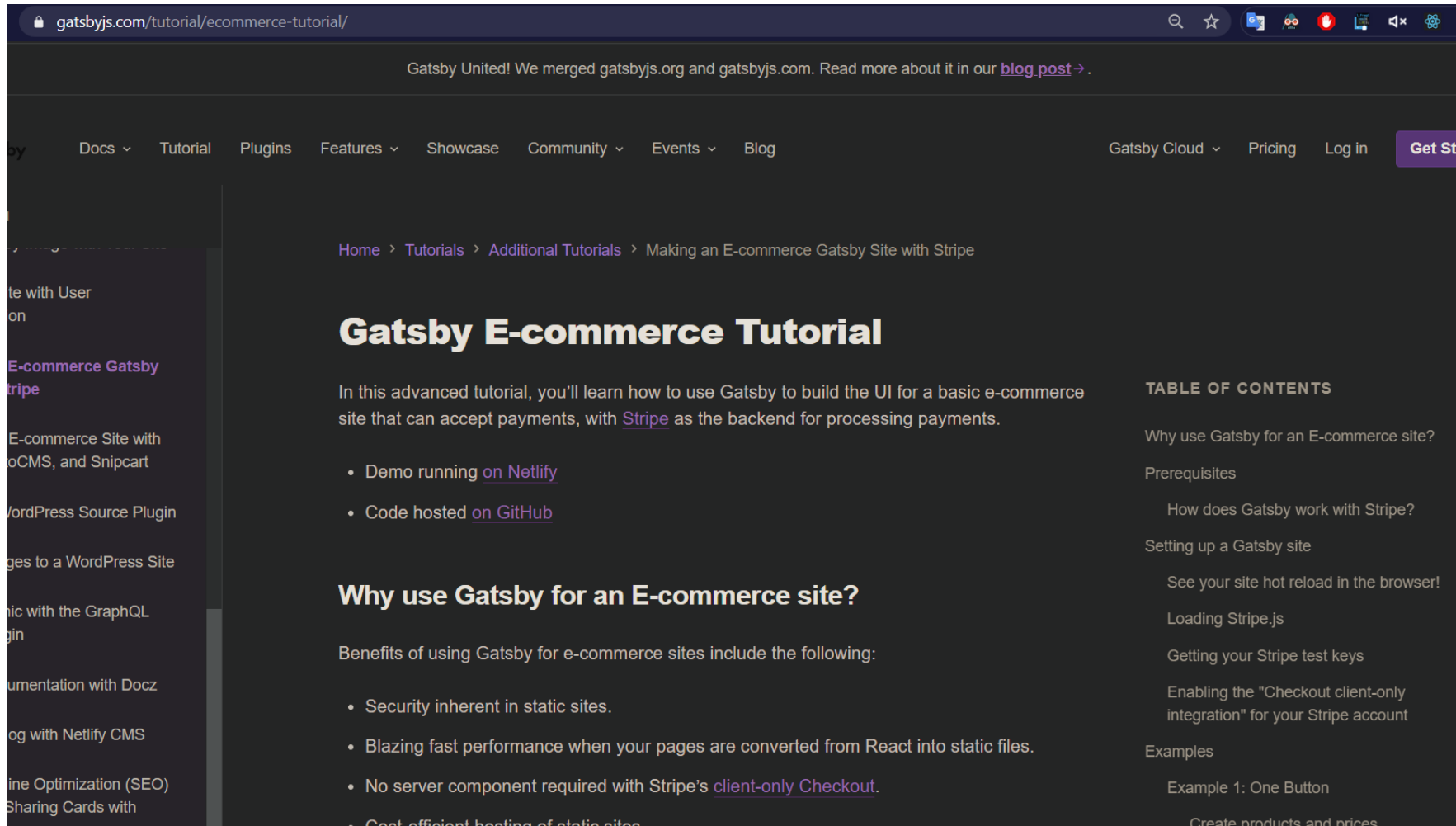


ÍNDICE DE CONTENIDOS

1	CONTEXTO Y OBJETIVOS PRINCIPALES
2	METODOLOGÍA, PLANIFICACIÓN Y COSTES
3	ANÁLISIS DE LAS TECNOLOGÍAS EMPLEADAS
4	DESARROLLO DEL PROYECTO
5	CONCLUSIONES

DESARROLLO DEL PROYECTO

01 Inicio de un proyecto con Gatsby



The screenshot shows the Gatsby E-commerce Tutorial page. The browser address bar displays `gatsbyjs.com/tutorial/ecommerce-tutorial/`. The page header includes navigation links: Docs, Tutorial, Plugins, Features, Showcase, Community, Events, Blog, Gatsby Cloud, Pricing, Log in, and a Get Started button. The main content area is titled "Gatsby E-commerce Tutorial" and includes a sub-header "Making an E-commerce Gatsby Site with Stripe". The text describes an advanced tutorial for building an e-commerce site with Stripe. A table of contents is provided on the right side of the page.

Gatsby E-commerce Tutorial

In this advanced tutorial, you'll learn how to use Gatsby to build the UI for a basic e-commerce site that can accept payments, with [Stripe](#) as the backend for processing payments.

- Demo running [on Netlify](#)
- Code hosted [on GitHub](#)

Why use Gatsby for an E-commerce site?

Benefits of using Gatsby for e-commerce sites include the following:

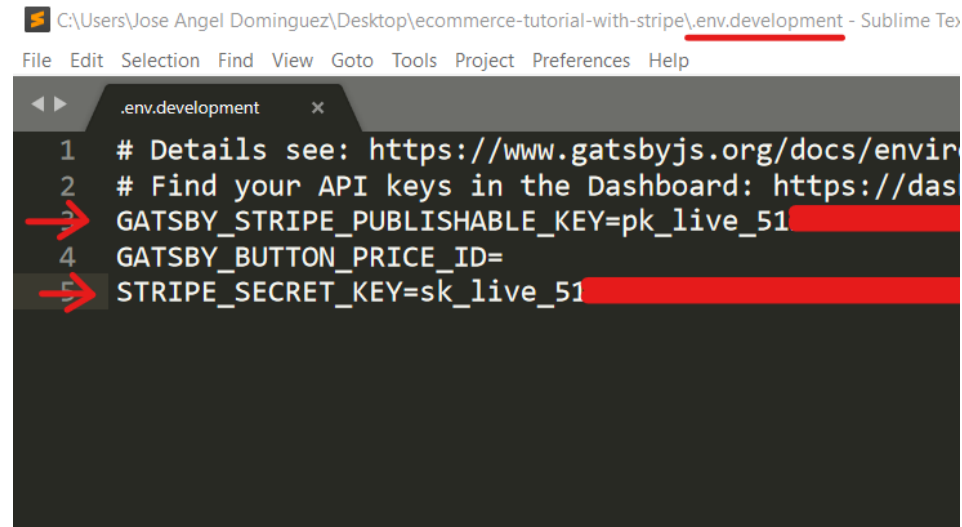
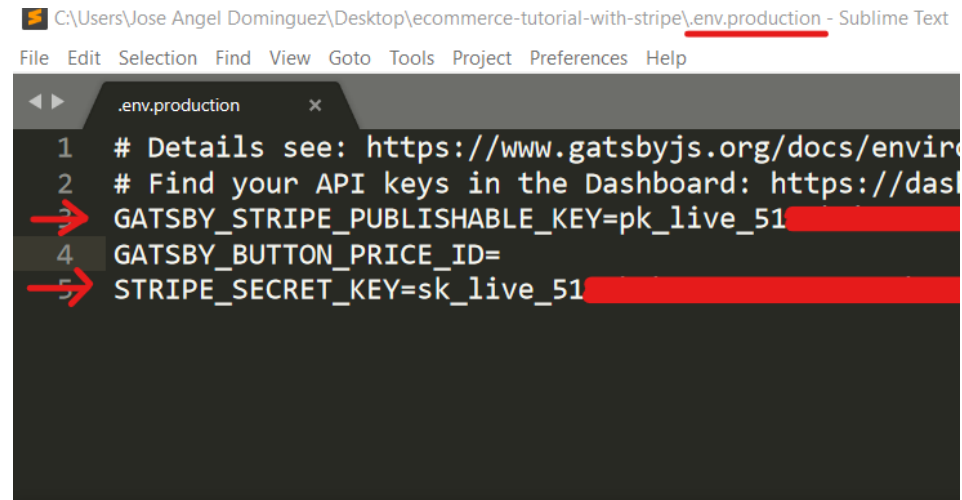
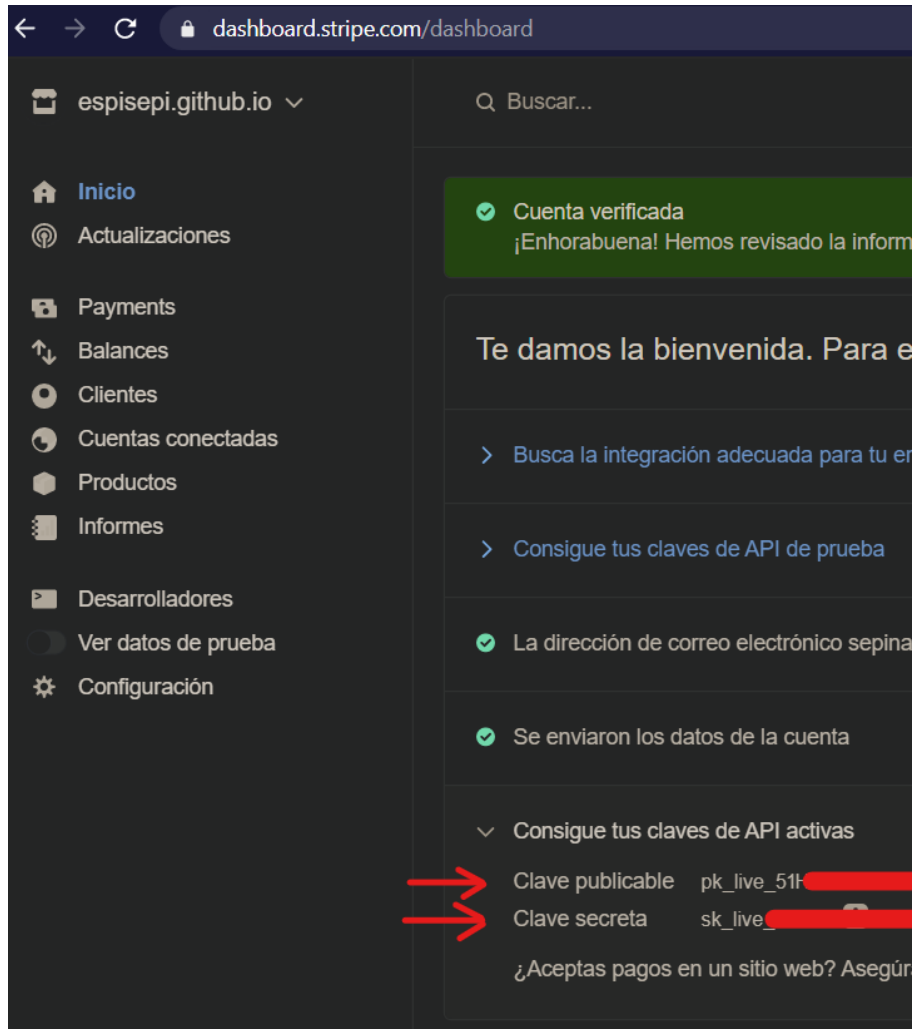
- Security inherent in static sites.
- Blazing fast performance when your pages are converted from React into static files.
- No server component required with Stripe's [client-only Checkout](#).
- Cost-efficient hosting of static sites.

TABLE OF CONTENTS

- Why use Gatsby for an E-commerce site?
- Prerequisites
 - How does Gatsby work with Stripe?
- Setting up a Gatsby site
 - See your site hot reload in the browser!
 - Loading Stripe.js
 - Getting your Stripe test keys
 - Enabling the "Checkout client-only integration" for your Stripe account
- Examples
 - Example 1: One Button
 - Create products and prices

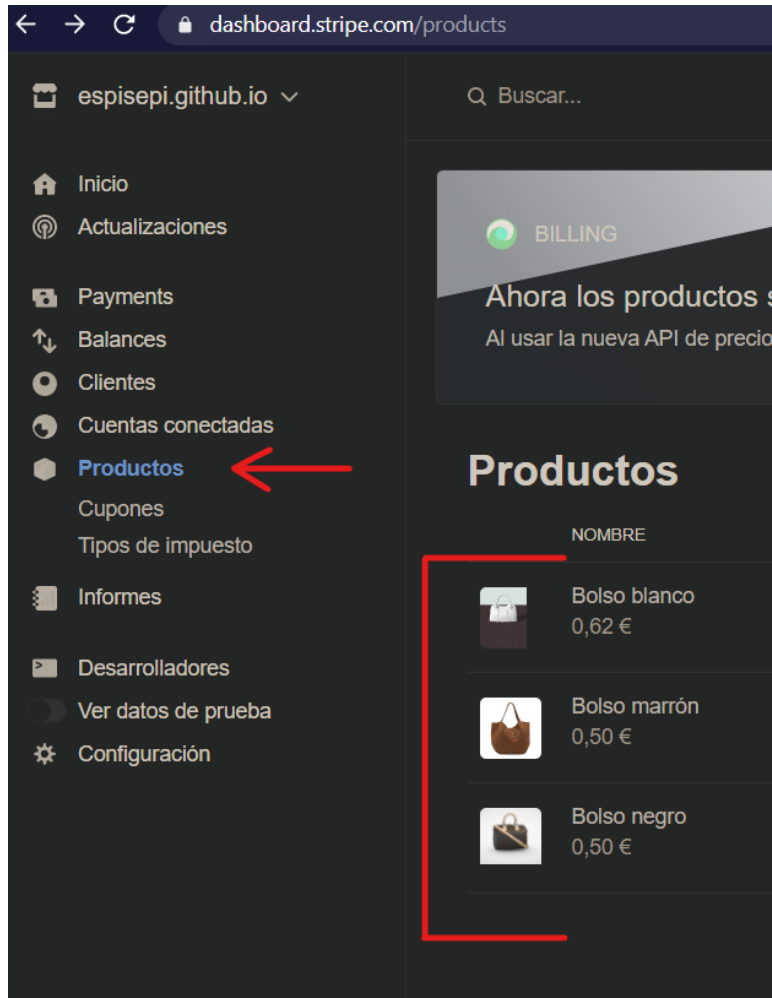
DESARROLLO DEL PROYECTO

02 Conexión a la API de Stripe



DESARROLLO DEL PROYECTO

03 Obtención de los productos por medio de GraphQL

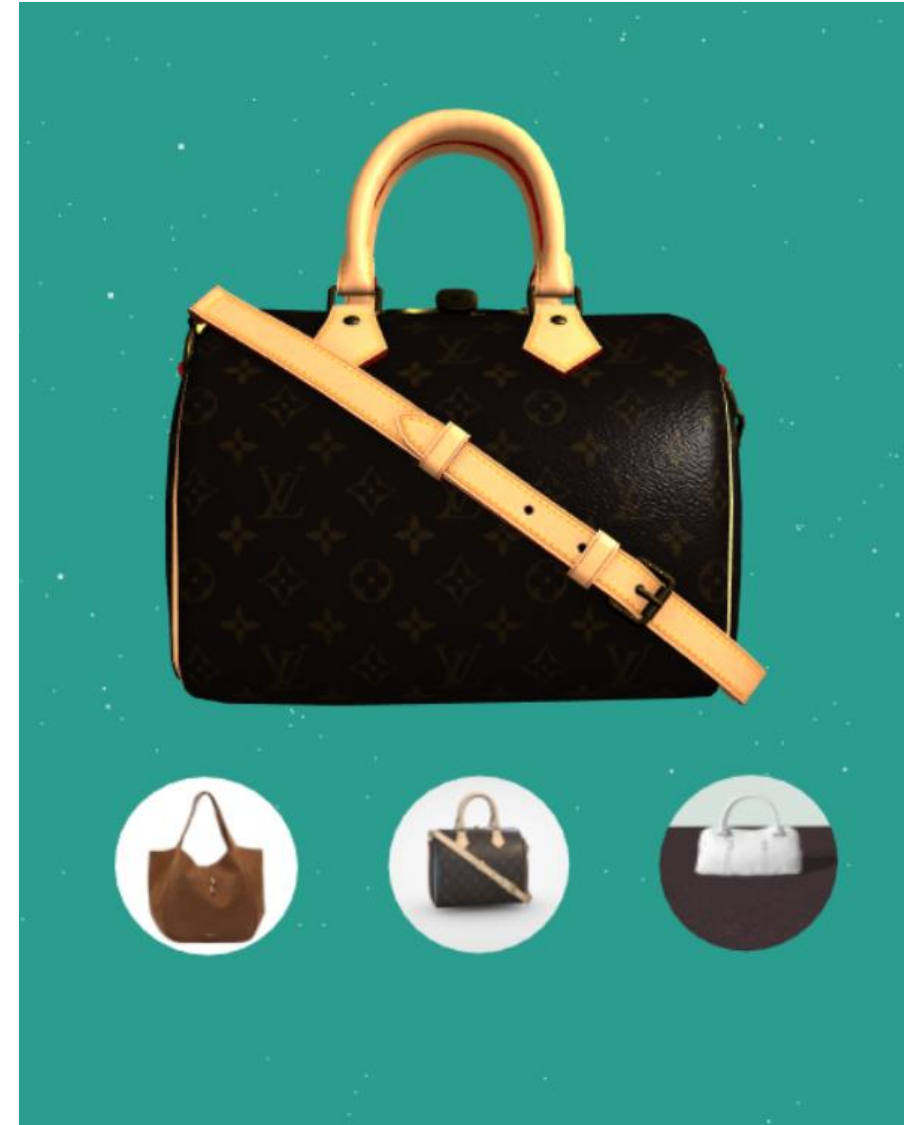
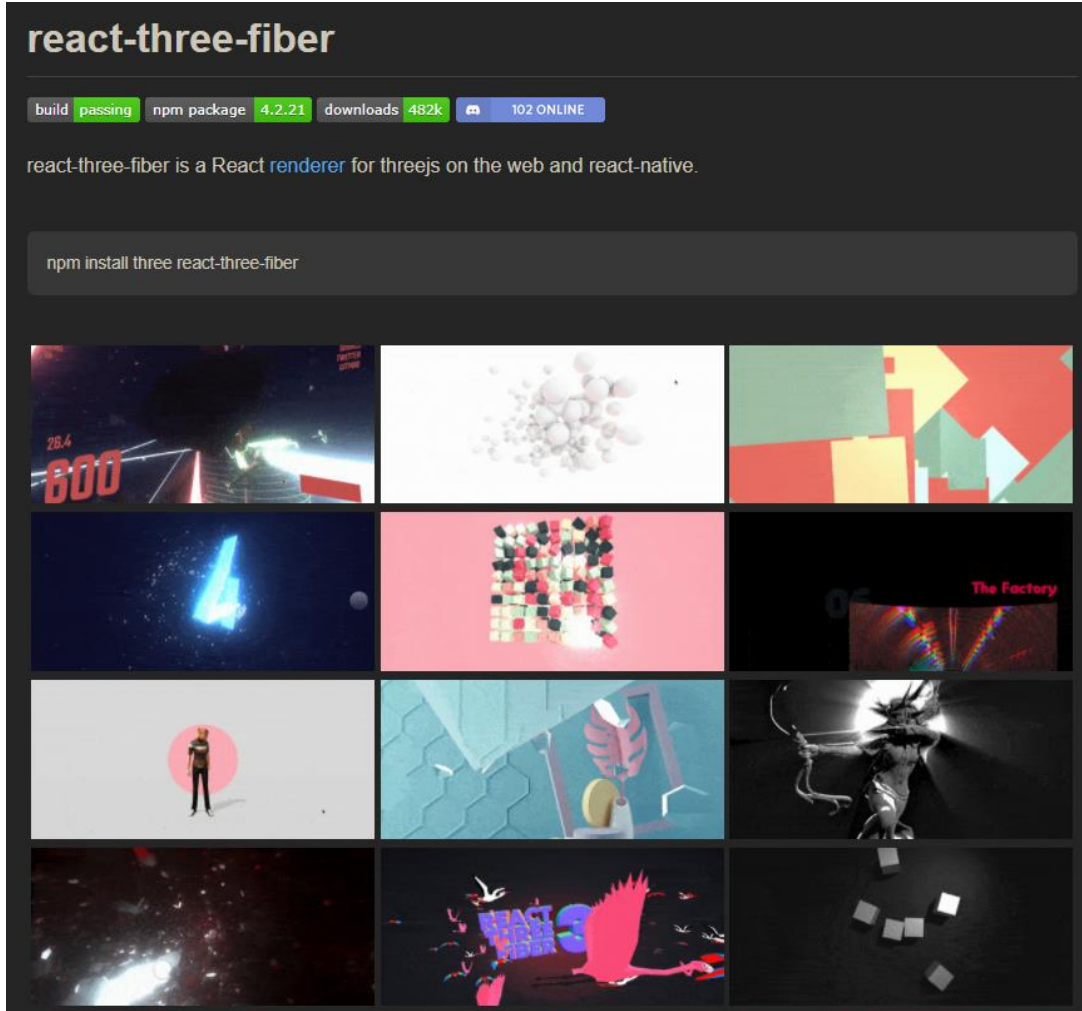


```
query MyQuery {
  allStripePrice {
    nodes {
      product {
        name
        description
        images
      }
    }
  }
}
```

```
{
  "data": {
    "allStripePrice": {
      "nodes": [
        {
          "product": {
            "name": "Bolso blanco",
            "description": "Este producto es ficticio, si introduce sus datos la compra será realizada.",
            "images": [
              "https://files.stripe.com/links/fl_live_GFCN9YC8x5eE9glhzZ5oXuYI"
            ]
          }
        },
        {
          "product": {
            "name": "Bolso marrón",
            "description": "Este producto es ficticio, si introduce sus datos la compra será realizada.",
            "images": [
              "https://files.stripe.com/links/fl_live_RfFUKaRGVB4utsxORB2E7RPF"
            ]
          }
        },
        {
          "product": {
            "name": "Bolso marrón",
            "description": "Este producto es ficticio, si introduce sus datos la compra será realizada.",
            "images": [
              "https://files.stripe.com/links/fl_live_RfFUKaRGVB4utsxORB2E7RPF"
            ]
          }
        },
        {
          "product": {
            "name": "Bolso negro",
            "description": "Este producto es ficticio, si introduce sus datos la compra será realizada.",
            "images": [
              "https://files.stripe.com/links/fl_live_1bG1S19Alubd1E5iIiHS46TL"
            ]
          }
        }
      ]
    }
  }
}
```

DESARROLLO DEL PROYECTO

04 Creación de la escena 3D con React-three-fiber



DESARROLLO DEL PROYECTO

05

Creación del fichero docker-compose.yaml

ayyazzafar Update README.md 5e8fa3d on 18 Aug 2019 4 commits

docker	files	13 months ago
.env	files	13 months ago
README.md	Update README.md	13 months ago
docker-compose.yaml	files	13 months ago

README.md

Install Lets Encrypt SSL Certificate using Docker, Wordpress and DigitalOcean

This is the source code created during a Video tutorial.

Youtube Video Link:

<https://youtu.be/lbXc6mKh7U0>

In this video tutorial, you would learn that how to install lets encrypt ssl certificate using Docker, Docker-Compose, Wordpress and Digital ocean.

Create DigitalOcean account using this link and get \$50 FREE: <https://m.do.co/c/40a4f3245660>

Create Payoneer account from this link and earn \$25: <https://bit.ly/2HbTXF2>

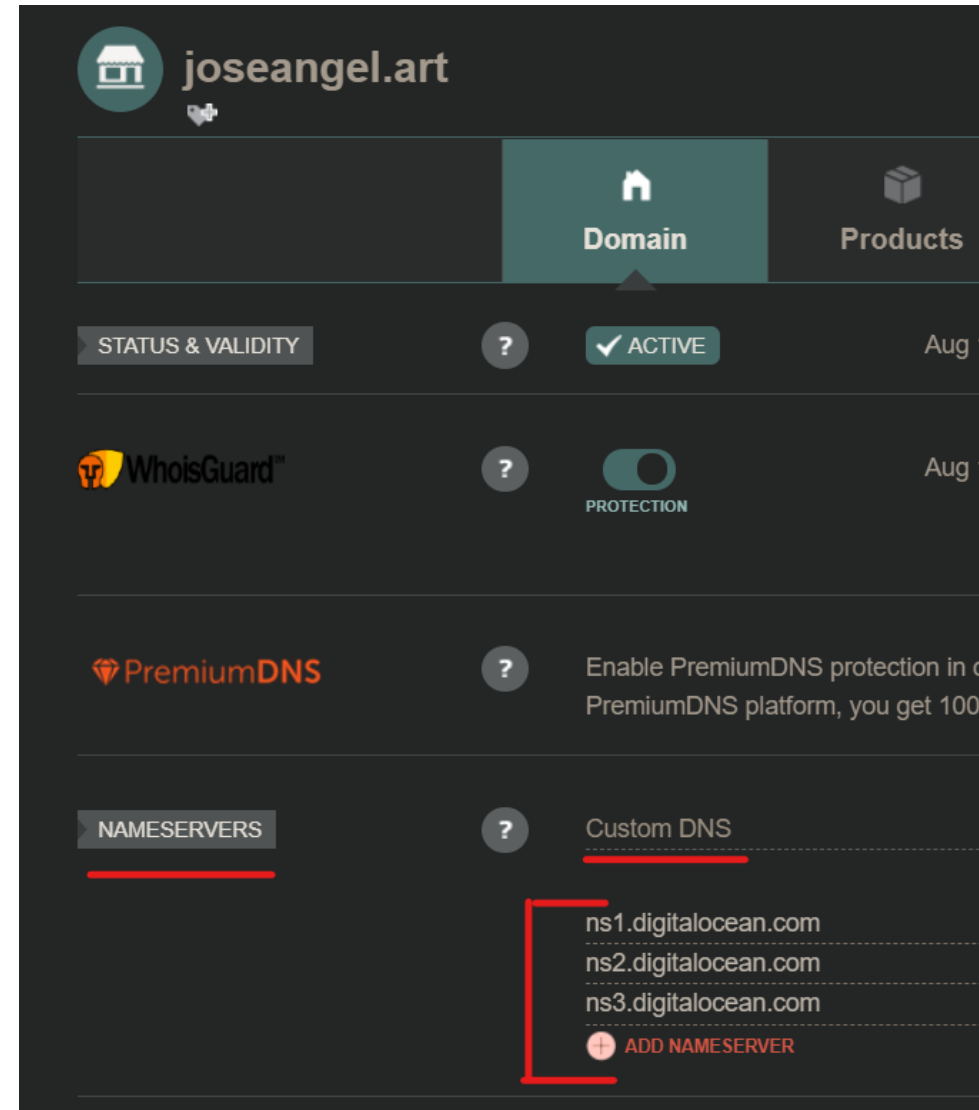
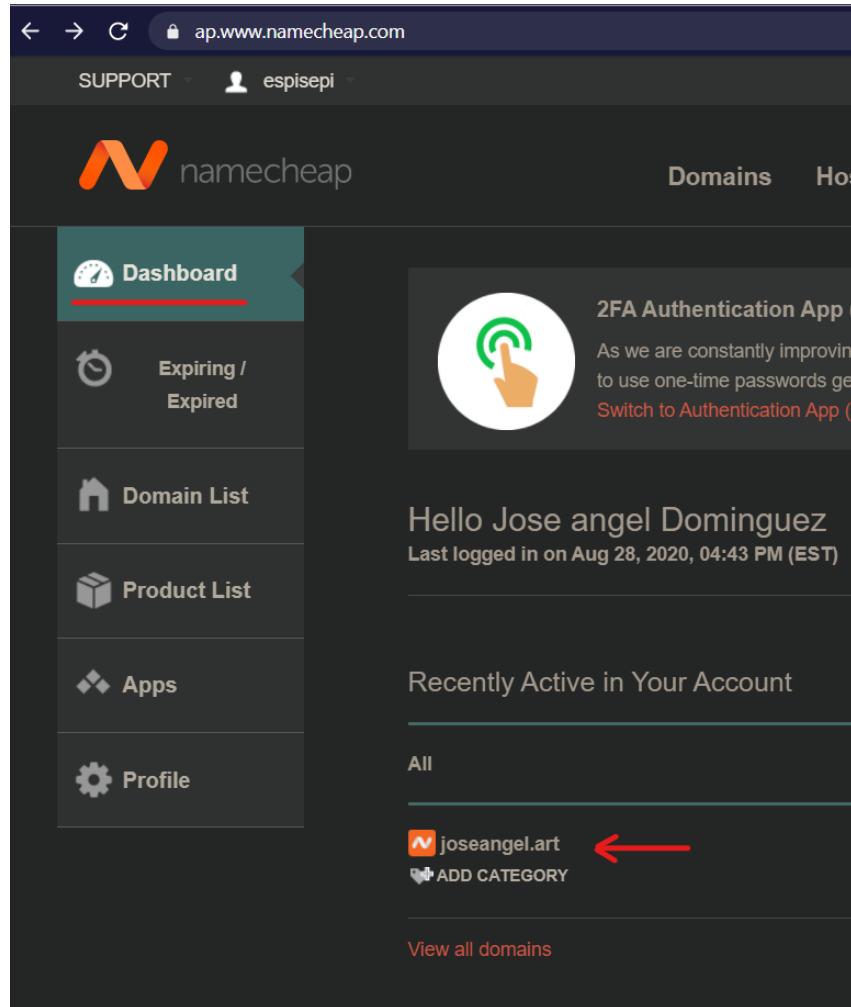
```

4
5  apache:
6    image: php:7-apache
7    container_name: apache
8    restart: always
9    stdin_open: true
10   tty: true
11   volumes:
12     - ./public:/var/www/html
13
14  nginx:
15    image: linuxserver/letsencrypt
16    ports:
17      - 80:80
18      - 443:443
19    volumes:
20      - ./docker/nginx/config:/config
21      - ./docker/nginx/nginx.conf:/config/nginx/site-confs/default
22      - ./docker/nginx/ssl.conf:/config/nginx/ssl.conf
23    container_name: nginx
24    restart: unless-stopped
25    environment:
26      - PUID=1000
27      - PGID=1000
28      - TZ=Europe/London
29      - URL=joseangel.art ←
30      - SUBDOMAINS=www,
31      - VALIDATION=http
32      - STAGING=false #optional

```

DESARROLLO DEL PROYECTO

06 Registro de un nombre de dominio



DESARROLLO DEL PROYECTO



07 Despliegue del proyecto en DigitalOcean

Create Droplets

Choose an image ?

Distributions Container distributions Marketplace Custom images

Q Docker

 Docker 19.03.12 on Ubuntu 20.04 [Details](#) 

Choose a plan

SHARED CPU

Basic General Purpose

Standard virtual machines with a mix of memory and compute resources. Best for small apps and dev/test environments.

Plan	Price	Resources
Basic	\$5/mo \$0.007/hour	1 GB / 1 CPU 25 GB SSD disk 1000 GB transfer
General Purpose	\$10/mo \$0.015/hour	2 GB / 1 CPU 50 GB SSD disk 2 TB transfer
Advanced	\$15/mo \$0.022/hour	1 GB / 3 CPUs 60 GB SSD disk 3 TB transfer

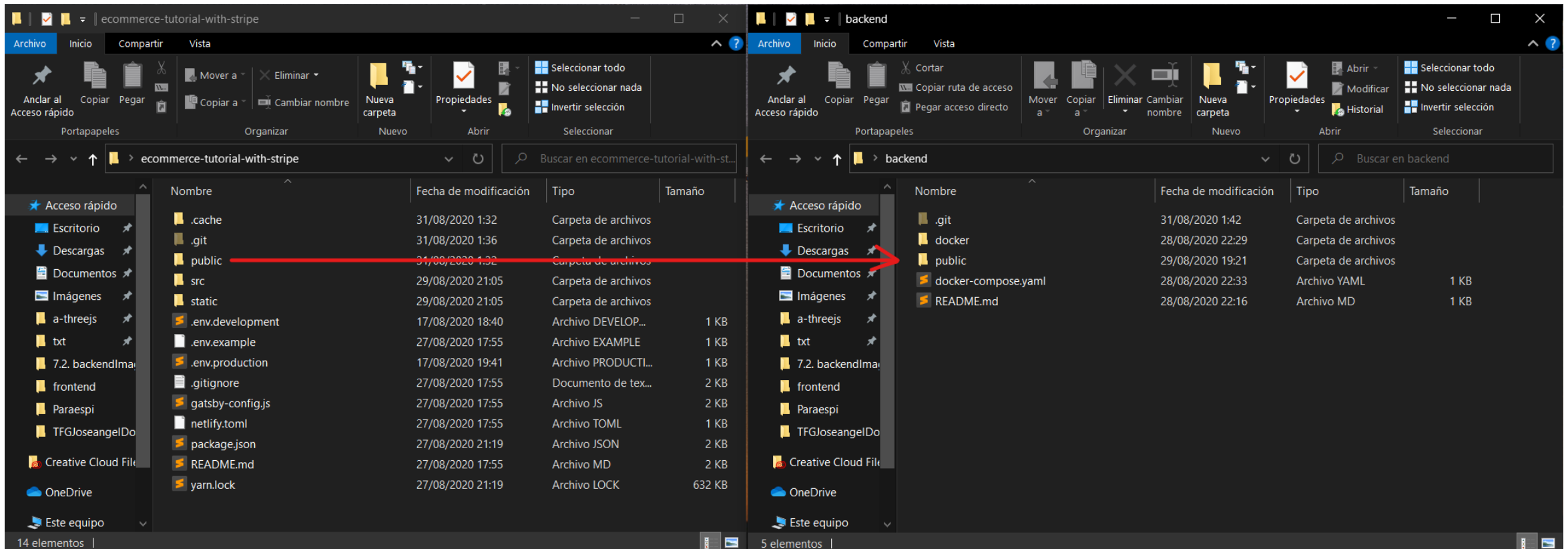
DNS records

Type	Hostname	Value
A	www.joseangel.art	directs to 164.90.215.243
A	joseangel.art	directs to 164.90.215.243
NS	joseangel.art	directs to ns3.digitalocean.com.
NS	joseangel.art	directs to ns1.digitalocean.com.
NS	joseangel.art	directs to ns2.digitalocean.com.

DESARROLLO DEL PROYECTO

07 Despliegue del proyecto en DigitalOcean

```
PS C:\Users\Jose Angel Dominguez\Desktop\ecommerce-tutorial-with-stripe> gatsby build
success open and validate gatsby-configs - 0.035s
success load plugins - 1.591s
success onPreInit - 0.051s
success delete html and css files from previous builds - 0.010s
success initialize cache - 0.010s
```



The screenshot displays two Windows File Explorer windows side-by-side. The left window is titled 'ecommerce-tutorial-with-stripe' and shows a directory listing with 14 elements. The right window is titled 'backend' and shows a directory listing with 5 elements. A red arrow points from the 'public' folder in the left window to the 'public' folder in the right window.

Nombre	Fecha de modificación	Tipo	Tamaño
.cache	31/08/2020 1:32	Carpeta de archivos	
.git	31/08/2020 1:36	Carpeta de archivos	
public	31/08/2020 1:32	Carpeta de archivos	
src	29/08/2020 21:05	Carpeta de archivos	
static	29/08/2020 21:05	Carpeta de archivos	
.env.development	17/08/2020 18:40	Archivo DEVELOP...	1 KB
.env.example	27/08/2020 17:55	Archivo EXAMPLE	1 KB
.env.production	17/08/2020 19:41	Archivo PRODUCTI...	1 KB
.gitignore	27/08/2020 17:55	Documento de tex...	2 KB
gatsby-config.js	27/08/2020 17:55	Archivo JS	2 KB
netlify.toml	27/08/2020 17:55	Archivo TOML	1 KB
package.json	27/08/2020 21:19	Archivo JSON	2 KB
README.md	27/08/2020 17:55	Archivo MD	2 KB
yarn.lock	27/08/2020 21:19	Archivo LOCK	632 KB

Nombre	Fecha de modificación	Tipo	Tamaño
.git	31/08/2020 1:42	Carpeta de archivos	
docker	28/08/2020 22:29	Carpeta de archivos	
public	29/08/2020 19:21	Carpeta de archivos	
docker-compose.yaml	28/08/2020 22:33	Archivo YAML	1 KB
README.md	28/08/2020 22:16	Archivo MD	1 KB

DESARROLLO DEL PROYECTO

07 Despliegue del proyecto en DigitalOcean

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Jose Angel Dominguez\Desktop>scp -r backend root@164.90.215.243:/root/
config                                100% 377      8.5KB/s   00:00
description                          100% 73       1.7KB/s   00:00
HEAD                                 100% 23       0.5KB/s   00:00
applypatch-msg.sample               100% 478     11.1KB/s   00:00
commit-msg.sample                   100% 896     20.9KB/s   00:00
fsmonitor-watchman.sample           100% 4655    105.7KB/s   00:00
post-update.sample                  100% 189      4.5KB/s   00:00
pre-applypatch.sample                100% 424     10.0KB/s   00:00
pre-commit.sample                   100% 1643    37.7KB/s   00:00
pre-merge-commit.sample              100% 416      9.8KB/s   00:00
pre-push.sample                     100% 1348    31.5KB/s   00:00
pre-rebase.sample                   100% 4898    113.5KB/s   00:00
```

```
C:\Users\Jose Angel Dominguez\Desktop>ssh root@164.90.215.243
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

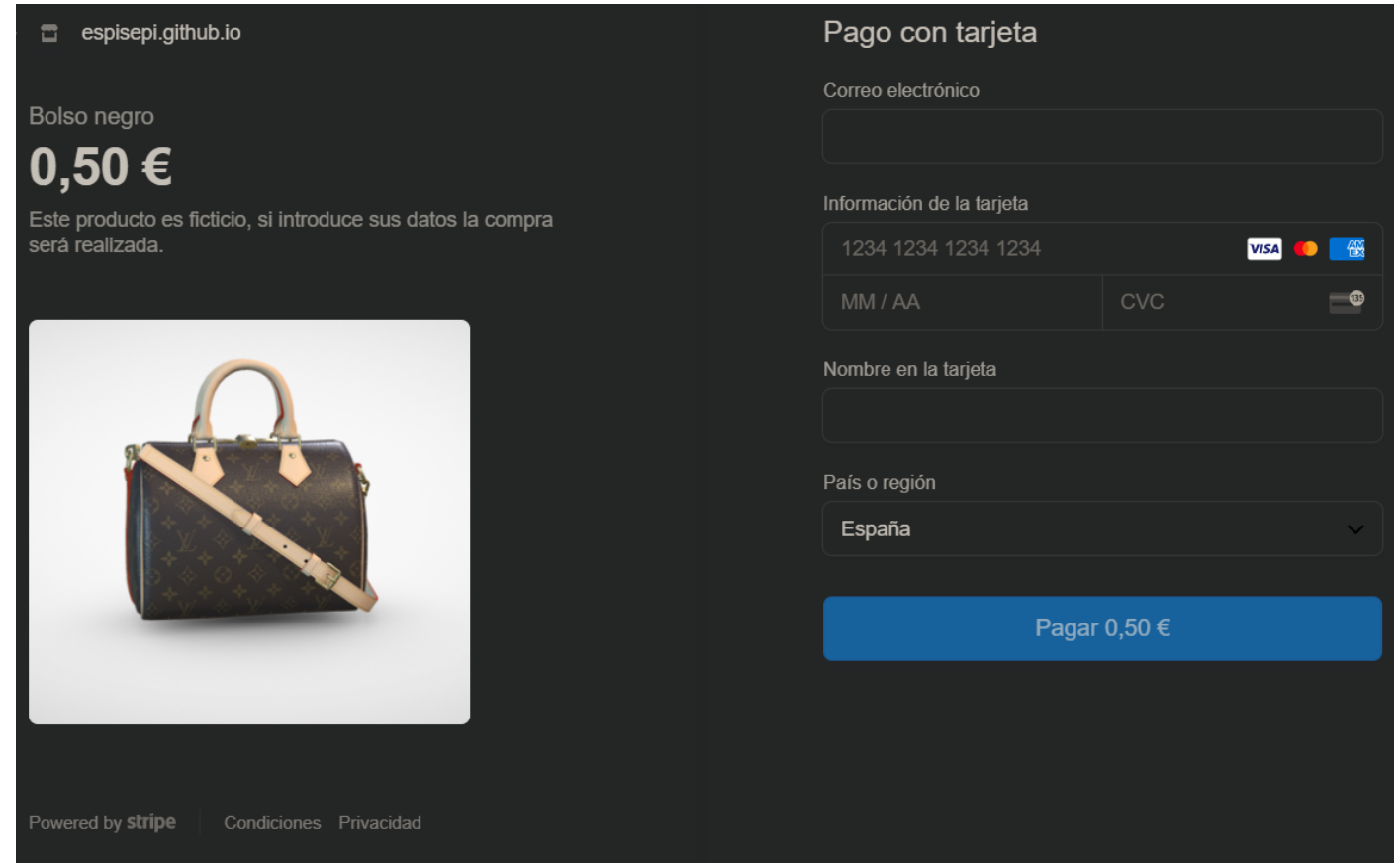
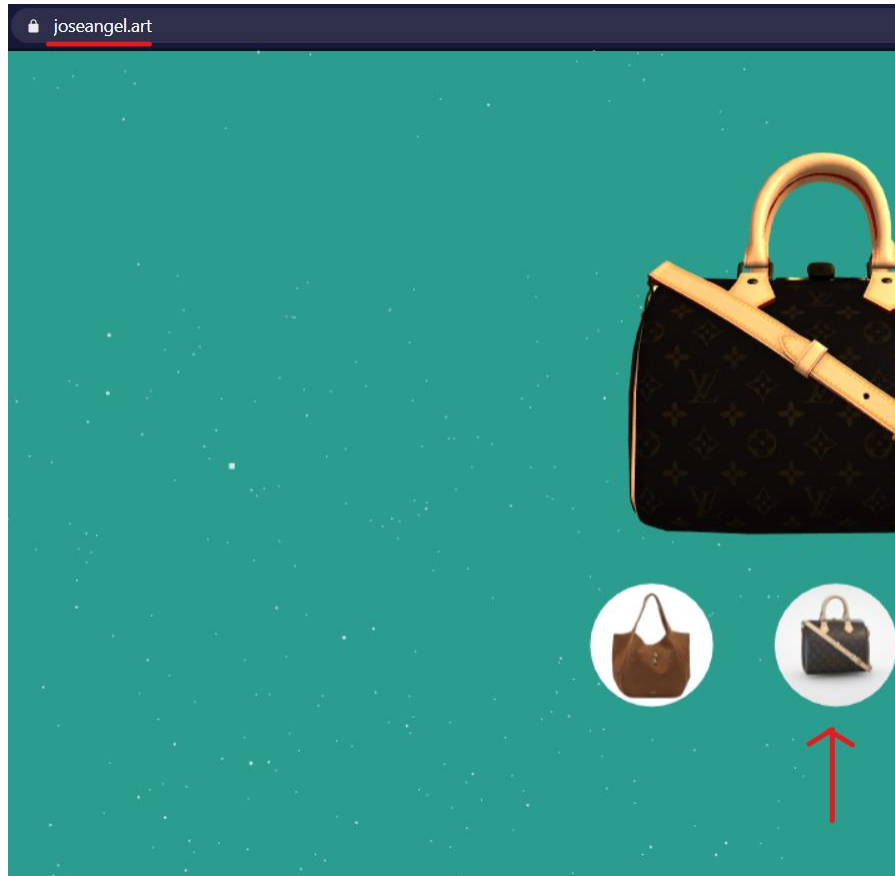
System information as of Sat Aug 29 17:38:06 UTC 2020
```

```
root@joseangelMachine:~# cd backend
root@joseangelMachine:~/backend# ls
README.md  docker  docker-compose.yaml  public
root@joseangelMachine:~/backend# docker-compose up -d
Creating network "backend_default" with the default driver
Pulling apache (php:7-apache)...
7-apache: Pulling from library/php
bf5952930446: Pull complete
a409b57eb464: Pull complete
3192e6c84ad0: Pull complete
43553740162b: Pull complete
```

DESARROLLO DEL PROYECTO

07

Despliegue del proyecto en DigitalOcean



ÍNDICE DE CONTENIDOS

1	CONTEXTO Y OBJETIVOS PRINCIPALES
2	METODOLOGÍA, PLANIFICACIÓN Y COSTES
3	ANÁLISIS DE LAS TECNOLOGÍAS EMPLEADAS
4	DESARROLLO DEL PROYECTO
5	CONCLUSIONES

CONCLUSIONES

Bloqueo CORS en la petición de imágenes al servidor de Stripe

Error al realizar la build con Gatsby

PROBLEMAS
ENCONTRADOS
DURANTE EL
DESARROLLO

Problemas con el evento onClick en smartphones / tablets

Error al dirigirse a la página Checkout de Stripe

CONCLUSIONES

BIEN

- Se han cumplido los objetivos del proyecto.
- Se ha cumplido con la fecha de entrega.
- Se han solucionado todos los problemas encontrados durante el desarrollo.

MAL

- No se ha cumplido con la planificación inicial.
- Se ha trabajado más horas de las planificadas.
- La solución al bloqueo CORS es una solución temporal. Dependemos de una aplicación externa.

CONCLUSIONES

Si comenzase de nuevo...

Aumentaría el margen de tiempo para la iteración correspondiente al desarrollo Frontend.



Aumentaría el margen de tiempo entre la finalización del proyecto y la fecha de entrega.



Tendría más en cuenta el desconocimiento de nuevas tecnologías a la hora de planificar los tiempos.



CONCLUSIONES

Trabajos futuros:

Desplegar nuestra propia aplicación Cors-anywhere.

```
README.md

build passing coverage 100%

CORS Anywhere is a NodeJS proxy which adds CORS headers to the proxied request.

The url to proxy is literally taken from the path, validated and proxied. The protocol part of the url is validated and defaults to "http". If port 443 is specified, the protocol defaults to "https".

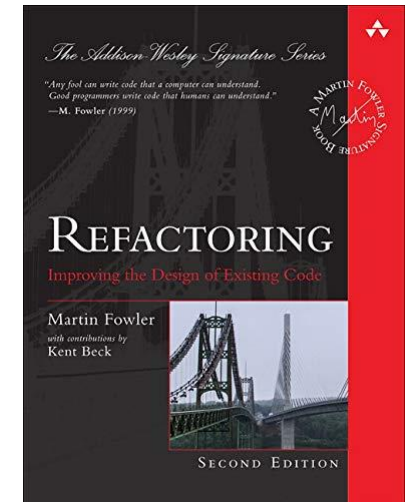
This package does not put any restrictions on the http methods or headers, except for cookies. Credentials is disallowed. The app can be configured to require a header for proxying a request to avoid a direct visit from the browser.

Example

// Listen on a specific host via the HOST environment variable
var host = process.env.HOST || '0.0.0.0';
// Listen on a specific port via the PORT environment variable
var port = process.env.PORT || 8080;

var cors_proxy = require('cors-anywhere');
cors_proxy.createServer({
  originWhitelist: [], // Allow all origins
  requireHeader: ['origin', 'x-requested-with'],
  removeHeaders: ['cookie', 'cookie2']
}).listen(port, host, function() {
  console.log('Running CORS Anywhere on ' + host + ':' + port);
});
```

Mejorar la organización del código y aplicar Refactoring donde fuese necesario.



Añadir la funcionalidad de carrito de compras.



PREGUNTAS

