

Descripción

Supervisión

Sprint 7 – Proyecto

¡Felicidades! Completaste la sección sobre herramientas de desarrollo de software. Ahora es el momento de aplicar los conocimientos y habilidades adquiridos al realizar un proyecto: construir y desplegar un panel de control de una aplicación web en un servicio en la nube.

Una vez que hayas terminado el proyecto, recuerda enviar tu trabajo al equipo de supervisión de proyectos para su evaluación. Te dará su opinión en 48 horas. Utiliza los comentarios para realizar cambios y luego envía la nueva versión al revisor.

Es posible que recibas más comentarios sobre esta nueva versión. Esto es totalmente normal. No es nada raro que pases por varios ciclos de comentarios y revisiones.

Tu proyecto se considerará completado una vez que el revisor del proyecto lo apruebe.

Descripción del proyecto

El objetivo de este proyecto es proporcionarte más posibilidades de practicar las tareas habituales de la ingeniería de software. Gracias a estas tareas, podrás aumentar y complementar tus habilidades en el campo de los datos y te convertirás en un candidato más atractivo para posibles empleadores.

Las tareas incluyen la creación y gestión de entornos virtuales de Python, el desarrollo de una aplicación web y su despliegue en un servicio en la nube que la hará accesible al público.

En este proyecto, te proporcionamos un conjunto de datos de anuncios de venta de coches. Sin embargo, en este proyecto, el enfoque no se centrará en el conjunto de datos ni en el análisis, por lo que eres libre de elegir cualquier dataset que desees.

Observa el aspecto que podría tener tu solución final:

Demo de la aplicación web que construirás en este proyecto



Sprint 7: Herramientas de desarrollo de software



proyecto

Paso 1. Configuración

1. Crea una cuenta en github.com. Si ya tienes una cuenta o creaste una mientras avanzabas en el capítulo [Git y GitHub](#), puedes omitir este paso.
2. Crea un nuevo repositorio git con un archivo `README.md` y un archivo `.gitignore` (hay que elegir la plantilla de Python). Si necesitas repasar, consulta [esta lección](#).

3. Si no lo has hecho, crea una cuenta en render.com. En [esta lección](#) se analizaron las aplicaciones web y de renderizado. Al crear una cuenta de Render, elige la opción "Github" y sigue los pasos de registro. Esto es exactamente lo que necesitas para vincular tu cuenta de Render a tu cuenta de GitHub.
4. Este proyecto implica realizar un análisis exploratorio de datos. Para lograrlo, necesitarás tener instalados los paquetes `pandas` y `plotly-express`. Anteriormente analizamos `plotly` en [esta lección](#). `plotly-express` está diseñado con valores predeterminados razonables y opciones configurables para tipos de gráficos comunes, lo cual lo convierte en un excelente punto de partida para principiantes en comparación con solo `plotly`. Si eres principiante en esto, no te preocupes, ¡te guiaremos a través del proceso!

Además, necesitarás el paquete `streamlit` para desarrollar una aplicación web.

Crea un nuevo entorno virtual y asígnale un nombre significativo que esté relacionado con el conjunto de datos con el que trabajarás, por ejemplo, podrías llamarlo `vehicles_env`.

Asegúrate de haber instalado al menos los siguientes paquetes en el entorno:
`pandas`, `streamlit` y `plotly-express`.

► Si tienes dificultades para encontrarlos, estos son los comandos para instalar las librerías:

Si necesitas un resumen sobre cómo crear un entorno virtual e instalar paquetes, consulta [la lección sobre este tema](#).

5. **Instala VS Code** si aún no lo has hecho. Clona el repositorio de tu proyecto desde GitHub y ábrelo como un proyecto en VS Code. Este será el directorio de tu proyecto. Configura el intérprete de Python como el utilizado por el entorno virtual que creaste anteriormente.
6. En aras de la simplicidad, en lugar de guardar tu entorno en el archivo `requirements.txt` del directorio del proyecto, crea manualmente el archivo `requirement.txt` y añade tres librerías ahí sin especificar las versiones:

```
pandas  
plotly_express  
streamlit
```

Paso 2. Descarga del archivo de datos

1. **Descarga el conjunto de datos de anuncios de coches** (`vehicles_us.csv`) o encuentra tu propio dataset en formato CSV.
2. Coloca el conjunto de datos en el directorio del proyecto.

Paso 3. Análisis exploratorio de datos

1. Crea un directorio llamado `notebooks` en el directorio de tu proyecto.

2. Crea un Jupyter notebook llamado `EDA.ipynb` en VS Code y guárdalo en el directorio `notebooks` de tu proyecto. Recuerda que `.ipynb` es una extensión de archivo para Jupyter Notebooks.
3. Abre el Jupyter notebook `EDA.ipynb` y experimenta con `plotly-express` para crear visualizaciones para el análisis exploratorio básico del conjunto de datos dentro del notebook. Estos son algunos ejemplos sobre:

► **Como construir un histograma mediante `plotly-express`.**

► **Cómo construir un gráfico de dispersión en esta librería.**

Si te interesa construir otros tipos de gráficos mediante `plotly-express`, en [esta página](#) puedes encontrar una lista de ejemplos de los gráficos básicos más comunes. La página incluye ejemplos, así como el código necesario para implementarlos. Sin embargo, no dediques demasiado tiempo a explorar el conjunto de datos, ya que el enfoque de este proyecto es diferente. Nuestro objetivo es crear una aplicación web y, aunque sería genial tener muchos gráficos y diagramas en la aplicación, en esta etapa no es necesario. En su lugar, vamos a enfocarnos en el trabajo integral necesario para implementar una aplicación de este tipo. Exactamente de esto se tratan los siguientes pasos.

Paso 4. Desarrollo del cuadro de mandos de la aplicación web

1. Crea un archivo `app.py` en la raíz del directorio de tu proyecto. Para crear un archivo `.py`, haz clic en "New File" (Nuevo archivo) en VS Code y guárdalo en el directorio del proyecto con el nombre deseado y la extensión `.py`.
2. Importa `streamlit` como `st`, `pandas` y `plotly_express` al principio del archivo.
3. Lee el archivo CSV del conjunto de datos en un `DataFrame`. El código será el mismo que tenías en el Jupyter Notebook al explorar el conjunto de datos.
4. Ahora, vamos a crear el contenido de nuestra aplicación basada en Streamlit. Esto es lo que queremos que incluyas:
 - Al menos un encabezado (puedes utilizar `st.header()` para hacerlo. En la **lección de aplicaciones web** te mostramos cómo crear un encabezado).
 - Crea un botón que, al hacer clic en él, construya un histograma `plotly-express`. Para hacerlo, considera utilizar las funciones `st.write()` y `st.plotly_chart()`.

Este es un ejemplo de cómo puedes hacerlo.

```

import pandas as pd
import plotly.express as px
import streamlit as st

car_data = pd.read_csv('vehicles-us.csv') # leer los datos
hist_button = st.button('Construir histograma') # crear un botón

if hist_button: # al hacer clic en el botón
    # escribir un mensaje
    st.write('Creación de un histograma para el conjunto de datos de vehículos')

    # crear un histograma
    fig = px.histogram(car_data, x="odometer")

    # mostrar un gráfico Plotly interactivo
    st.plotly_chart(fig, use_container_width=True)

```

- Agrega otro botón que, al hacer clic en él, construya un gráfico de dispersión `plotly-express`. Nuevamente, considera utilizar las funciones `st.write()` y `st.plotly_chart()`.

Esto es opcional, pero si quieres un desafío extra, considera reemplazar los botones por casillas de verificación, las cuales están disponibles en `streamlit` a través de `st.checkbox()`. Puedes pedirle al usuario o la usuaria que seleccione una casilla de verificación correspondiente a un histograma o un diagrama de dispersión y

luego generar un gráfico basado en la casilla de verificación seleccionada. Este es un ejemplo simple de cómo funcionan las casillas de verificación en `streamlit`:

```
import streamlit as st

# crear una casilla de verificación
build_histogram = st.checkbox('Construir un histograma')

if build_histogram: # si la casilla de verificación está seleccionada
    st.write('Construir un histograma para la columna odómetro')
    ...
```



5. Asegúrate de actualizar el archivo `README` cuando hayas terminado. Este debe incluir una breve descripción del proyecto, donde se explique para qué sirve la aplicación web y qué tipo de funcionalidad proporciona.
6. Para hacer que Streamlit sea compatible con Render, añade un archivo de configuración de Streamlit al repositorio de tu proyecto en `streamlit/config.toml` con el siguiente contenido:


```
[server]
headless = true
port = 10000

[browser]
serverAddress = "0.0.0.0"
serverPort = 10000
```

Le dirá a Render que busque en el sitio correcto para escuchar tu aplicación de Streamlit cuando la aloje en sus servidores.

7. Recuerda confirmar y empujar todos los cambios a tu repositorio cuando hayas terminado tu trabajo. Si no lo haces, nada funcionará correctamente.

Notas a considerar:

- Mientras desarrollas tu aplicación añadiendo un nuevo componente de Streamlit, puedes ejecutar el comando `streamlit run app.py` desde la terminal para ver el resultado.
- Es una buena práctica confirmar y enviar tu trabajo a un repositorio remoto en GitHub a medida que vas alcanzando ciertos objetivos en el desarrollo de la aplicación (por ejemplo, añades un componente que funciona y la aplicación se ejecuta sin errores). ¡Así que no te olvides de escribir un mensaje de confirmación significativo!

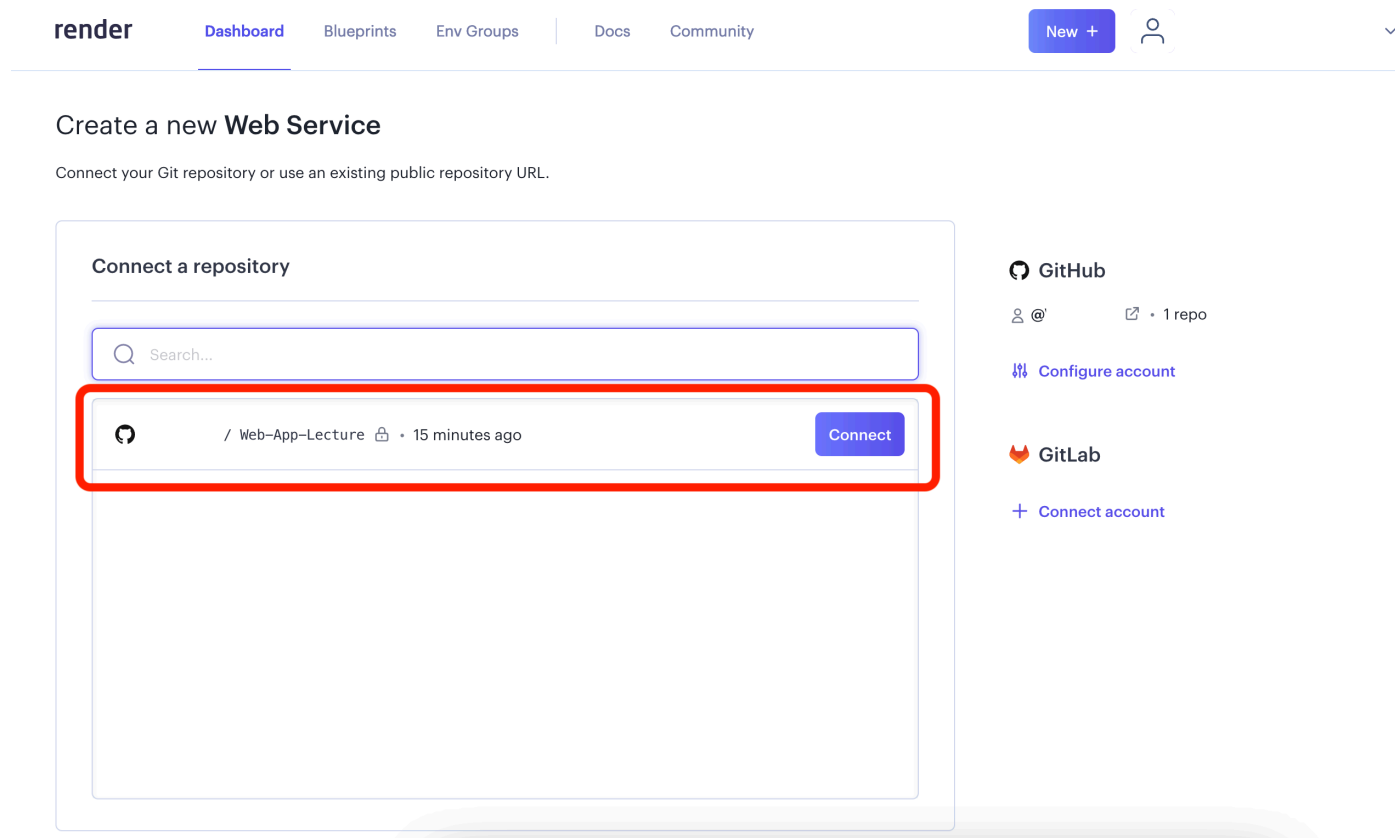
Paso 5. Despliegue de la versión final de la aplicación en Render

1. Accede a tu cuenta en render.com y crea un nuevo servicio web:

The screenshot shows the Render.com dashboard. At the top, there's a navigation bar with 'render' logo, 'Dashboard', 'Blueprints', 'Env Groups', 'Docs', and 'Community'. On the right, there's a 'New +' button and a user profile icon. Below the navigation bar, the 'Overview' section is titled 'Get up and running in minutes'. It features eight service cards arranged in a 2x4 grid:

- Static Sites**: Static sites are automatically served over a global CDN. Add a custom domain and get free, fully-managed SSL. [?](#)
 Button: New Static Site
- Web Services**: Web services include zero-downtime deploys, persistent storage and PR previews. Scale up and down with ease. [?](#)
 Button: New Web Service (highlighted with a red arrow)
- Private Services**: Private services are only accessible within your Render network and can speak any protocol. [?](#)
 Button: New Private Service
- Background Workers**: Background workers are suitable for long running processes like consumers for queues and streaming. [?](#)
 Button: New Worker
- Cron Jobs**: With cron jobs you can schedule any command or script to run on a regular interval. [?](#)
 Button: New Cron Job
- PostgreSQL**: Fully-managed hosted PostgreSQL with internal and external connectivity, and automated daily backups. [?](#)
 Button: New PostgreSQL
- Redis**: A cloud based in-memory key value datastore. Render offers fully managed hosted Redis instances. [?](#)
 Button: New Redis
- Blueprints**: A Blueprint specifies your Infrastructure as Code in a single file. Use it to set up all your services at once. [?](#)
 Button: New Blueprint


2. Crea un nuevo servicio web enlazado a tu repositorio de Github:



3. Configura el nuevo servicio web Render añadiendo un **Build Command** que instale todo lo necesario para iniciar tu app, incluyendo `streamlit` y todos los paquetes de `requirements.txt`. Utiliza el siguiente comando:

```
pip install --upgrade pip && pip install -r requirements.txt
```

Añade a tu **Start Command**: `streamlit run app.py`. Debería verse así:

render [Dashboard](#) [Blueprints](#) [Env Groups](#) [Docs](#) [Community](#) [New +](#) 

You are deploying a web service for `/Web-App-Lecture.`

You seem to be using **Python**, so we've autofilled some fields accordingly. Make sure the values look right to you!

Name
A unique name for your web service.

lecture 8

Environment
The runtime environment for your web service.

Python 3

Region
The [region](#) where your web service runs.

Oregon (US West)

Branch
The repository branch used for your web service.

main

Build Command
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

`$ pip install streamlit & pip install -r requirements.txt`

Start Command
This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.

`$ streamlit run app.py`

4. Despliega en Render y espera a que el build se ejecute con éxito:

The screenshot shows the Render dashboard for a service named "lecture 8". The top navigation bar includes "render", "Dashboard", "Blueprints", "Env Groups", "Docs", "Community", a "New +" button, and a user profile icon. The service details section shows "lecture 8" with a "Python 3" tag, a "Free Plan" badge, and a repository link "r/Web-App-Lecture" with a "main" branch. There are "Connect" and "Manual Deploy" buttons. Below this, a sidebar on the left lists various tabs: Events, Logs, Disks, Environment, Shell, PRs, Jobs, Sharing, Metrics, Scaling, and Settings. The main content area shows a notification about slow builds, a log entry for August 30, 2022 at 1:59 PM with a status of "In progress", and a search bar for logs. The log output is displayed in a dark-themed window, showing the build process from generating the container image to uploading the build. The final line of the log, "Aug 30 02:04:55 PM ==> Build successful 🎉", is highlighted with a red rectangle. Below the log window is a "Scroll to bottom" button.

render Dashboard Blueprints Env Groups Docs Community New +

WEB SERVICE

lecture 8 Python 3 Free Plan r/Web-App-Lecture main Connect Manual Deploy

https://lecture-8.onrender.com

Events

Builds too slow? Upgrade to a paid plan to go faster. Learn more about free plan limits.

Logs

August 30, 2022 at 1:59 PM In progress

Disks

08da094 streamlit app

Environment

Shell Search logs Go Maximize Scroll to top

PRs

Jobs

Sharing

Metrics

Scaling

Settings

```
2.28.1 rich-12.5.1 scipy-1.7.3 semver-2.13.0 six-1.16.0 smmap-5.0.0 streamlit-1.12.2 toml-0.10.2 toolz-0.12.0 tornado-6.2 typin
g-extensions-4.3.0 tzdata-2022.2 tzlocal-4.2 urllib3-1.26.12 validators-0.20.0 watchdog-2.1.9 zipp-3.8.1
Aug 30 02:00:52 PM WARNING: You are using pip version 20.1.1; however, version 22.2.2 is available.
Aug 30 02:00:52 PM You should consider upgrading via the '/opt/render/project/src/.venv/bin/python -m pip install --upgrade pi
p' command.
Aug 30 02:00:53 PM ==> Generating container image from build. This may take a few minutes...
Aug 30 02:04:00 PM ==> Uploading build...
Aug 30 02:04:54 PM ==> Build uploaded in 34s
Aug 30 02:04:55 PM ==> Build successful 🎉
Aug 30 02:04:55 PM ==> Deploying...
```

Scroll to bottom

5. Comprueba que tu aplicación sea accesible a través de la siguiente URL:

`https://<APP_NAME>.onrender.com/` .

Nota: pueden pasar varios minutos después de un despliegue exitoso hasta que la aplicación esté disponible online en un nivel gratuito. También hay que tener en cuenta que las aplicaciones se quedan "dormidas" después de estar inactivas durante unos minutos. Si es así, simplemente carga y actualiza tu aplicación unas cuantas veces para que se despierte.

Si actualizas tu repositorio de GitHub, para implementar la versión más reciente en Render, haz clic en "Manual Deploy" → "Latest Commit" (Implementación manual → Última confirmación).

¿Cómo puedo enviar mi proyecto?

Deberás enviar un enlace a tu repositorio de GitHub; también, añade la URL de tu aplicación en Render al `README.md` de tu proyecto.

¿Cómo será evaluado mi proyecto?

Hemos recopilado los criterios de evaluación del proyecto. Léelos detenidamente antes de enviarlo.

Esto es lo que buscan los revisores de proyecto cuando evalúan tu proyecto:

- ¿Contiene el repositorio del proyecto al menos los siguientes archivos?

```
$ tree
.
├── README.md
├── app.py
├── vehicles_us.csv
├── requirements.txt
├── notebooks
│   └── EDA.ipynb
├── .streamlit
│   └── config.toml
```

- ¿Se puede acceder a la aplicación web a través de un navegador?
- ¿Contiene la aplicación web los siguientes puntos?
 - Al menos un encabezado con texto.
 - Al menos un histograma.
 - Al menos un gráfico de dispersión.
 - Al menos un botón o una casilla de verificación.