Primero de todo he hecho un git clone con el directorio del proyecto UOC Boilerplate. Seguidamente he borrado la carpeta .git. Despues con el comando git init la he creado de nuevo y con git commit -m "mensaje" he hecho mi primer commit. Con el comando git remote add origin seguido de la URL de mi directorio de github he vinculado mi trabajo en local con el repositorio online. Seguidamente con git push -u origin master he actualizado mi repositorio online con el local.

En este proyecto ya el packaga.json por ese motivo no teníamos que crearlo con npm start, pero si que debiamos de hacer npm install para instalar todas las dependencias. Con el comando npm run dev nos elimina el historial del ejecutado anteriormente y nos abre un puerto en local. Con el comando npm run build nos prepara toda la página minificada en la carpeta dist preparada para subirla a producción.

Para la publicación en Netlify le das al botón ADD A NEW PROJECT, seguidamente lo linkeas con github en mi caso y luego sigues las instrucciones hasta llegar a donde tienes que decidir la branch que en mi caso es la master y el Dir que es donde npm run build te crea la página es decir en dist. Y, por último, en build commando el comando de producción es decir npm run build. Despues cada vez que hagas un push se actualizará automaticamente.

Luego para instalar el paquete de tailwind he escrito por línea de comandos: npm install tailwindcss --save-dev y seguidamente con el comando npx tailwind init he generado el documento base de configuración.

En el documento css colocamos

@tailwind base;

@tailwind components;

//Todo nuestros componentes con @apply

@tailwind utilities;

// seguido de nuestras utilidades

He tenido problemas instalandoel plugin el de transforms al final he visto que era mejor crear unas utilidades en mi css. Tampoco he logrado usar la queries de Tailwind (@screen md{}). Lo he podido sustituir por las queries normales de css.

* ¿Qué diferencias hay entre el enfoque de tipo CSS semántico (el que usaste en las otras PEC) al respecto del CSS de utilidades? ¿Cómo afectó esto a tu proceso de desarrollo? ¿Y a tu código?

El css semántico te obliga a estructurar más tu CSS y siempre pensar en hacer clases más genéricas y luego otras más específicas. La manera de elegir los nombres aunque si que es verdad que es menos flexible para otros proyectos o otros casos te ayuda a con un vistazo ver de donde viene cada cosa. El CSS de utilidades vas colocando clases muy especificas en cada elemento y esto te puede llevar a, en el futuro cuando el html sea más largo a cometer errores. Por otro lado, es más rápido y en poco tiempo puedes montar una página en css y no te requiere tantos conocimientos de css ya que los nombres de las clases son intuitivas.

* ¿Qué diferencias encontraste entre usar una libería de componentes y una libería de utilidades?

Componentes te ata más a una estructura y por consecuencia te limita pero por otro lado es más fácil, se requieren menos conocimientos, y es más rápido. Trabajar con utilidades es más versátil ya que cada clase es muy específica pero necesita más tiempo y conociminetos

* ¿Qué componentes decidiste extraer y por qué?

He extraído básicamente los componentes que se repetían, es decir, el navegador(.nav, .nav\_item) y las clases para dar el efecto polaroid y de giro(.cardFlip, .cardFlip\_\_box, .cardFlip—face, .cardFlip—back, .btn—cardflip, .coverImage—polaroid). En los atributos que se repetían en diferentes componentes he creado utilidades y en los que solamente los usaba una vez los he llamado directamente con css.