Software recomendado

- VS Code https://code.visualstudio.com/download
    - Extensiones de VS Code: ESLint, GitLens, Git Graph, Git File History, Git History, Better Comments, Angular Language Service, Angular Snippets, CSS Formatter, dotENV, Excel Viewer, htmltagwrap, indent-rainbow, Live Share, Live Share Audio, Live Share Extension Pack, npm , npm Intellisense, Python, Tabnine, vscode-pdf, Nx Console
- Postman https://www.postman.com/downloads/
- Node LTS https://nodejs.org/download/release/v16.19.0/
- Python https://www.python.org/downloads/
- Chrome, Firefox
- DBeaver Community (cliente de BBDD SQL) https://dbeaver.io/download/
- MobaXTerm (recomendado) https://mobaxterm.mobatek.net/download-home-edition.html, Cyberduck https://cyberduck.io/download/, WinSCP https://winscp.net/eng/download.php (conexiones SFTP)
- Cmder (consola avanzada para Windows) https://cmder.net/
- Spotlight Studio (documentación OpenAPI): https://github.com/stoplightio/studio/releases
- Docker Desktop: Windows https://docs.docker.com/desktop/windows/install/ https://docs.docker.com/desktop/linux/install/
- RunJS (Playground de JavaScript y TypeScript) https://runjs.app/
- Studio 3T (cliente de MongoDB) https://studio3t.com/download/
- CCleaner (limpieza de equipo, opcional) https://www.ccleaner.com/es-es/download
- Driver Booster (actualización de controladores, opcional) https://www.iobit.com/es/driver-booster.php

Itinerario

1. Curso de Git, GitHub y SourceTree: https://bosonit-my.sharepoint.com/:f:/p/fernando_merino/EgANGgIuzPpLgKBKS0eJtOkBFDPgjJ-6ff7YDw1mkTCoDA
2. Convención de trabajo en Git y Gitlab https://bosonit.sharepoint.com/:w:/s/ELLIoT-FullStackWeb/ER1Mam7lJeNFrqo4VtCr3cEBOtoGDb0frS3rQIOUoE5aHw?e=whlfAq
3. Certificación completa en JS de freeCodeCamp https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/
4. Principios SOLID en JS: https://hackernoon.com/understanding-solid-principles-in-javascript-w1cx3yrv
5. Curso de TypeScript - Aprender typescript desde cero https://www.youtube.com/watch?v=IJ_mpJRaHmc&list=RDCMUC3QuZuJr2_EOUak8bWUd74A
6. Decoradores en Typescript (no se trata este concepto en el curso anterior):
    - https://bosonit-my.sharepoint.com/:v:/p/fernando_merino/EXRLZ7MdSY1DnFUTLdgsx6UBYMwJB6cFZXW9mzKH-vvcEw?e=fM4mpq
    - https://www.typescriptlang.org/docs/handbook/decorators.html
    - https://diegoboscan.com/decoradores-en-typescript/
7. Principios SOLID en TS: https://blog.bitsrc.io/solid-principles-in-typescript-153e6923ffdb
8. Build your first Angular app: https://scrimba.com/learn/yourfirstangularapp
9. Curso Angular en Español - Tutorial de Angular 12 desde cero https://www.youtube.com/watch?v=i-oYrcNtc2s
10. Principios SOLID en Angular:
    - https://medium.com/geekculture/solid-angular-a-guide-to-writing-better-code-a919a4af1bf2

    - https://indepth.dev/posts/1414/angular-and-solid-principles
11. Learn Angular Material - Full Tutorial https://www.youtube.com/watch?v=jUfEn032IL8
12. Convención en Git y Gitlab y arquitectura de Angular en proyectos de Elliot Convención en git y Gitlab y arquitectura en Angular.mp4
13. Node.js and Express.js - Full Course: https://www.youtube.com/watch?v=Oe421EPjeBE
14. SemVer: Versionado semántico en Node y npm: https://bernardoayala.com/npm-versionado/
15. Cómo actualizar paquetes de npm de forma segura:
    - Artículo: https://chrispennington.blog/blog/how-to-update-npm-packages-safely-with-npm-check-updates/
    - Vídeo: https://www.youtube.com/watch?v=0XQXGx3lLaU
16. PostgreSQL & Nodejs REST API, Sequelize, Babel (Versión antigua 2019) https://www.youtube.com/watch?v=sA3t4d1v7OI
17. Sequelize REST API (usando Postgres) (NUEVA VERSIÓN 2022): https://youtu.be/3xiIOgYdbiE
18. Convención de nombres en un API REST https://restfulapi.net/resource-naming/
19. Buenas prácticas API REST
    - https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9
    - https://elbauldelprogramador.com/buenas-practicas-para-el-diseno-de-una-api-restful-pragmatica/
20. SEQUELIZE (Documentación de gestión de asociación entre tablas, importante lectura)
21. Associations https://sequelize.org/docs/v6/core-concepts/assocs/
22. Advanced Association Concepts: https://sequelize.org/docs/v6/category/advanced-association-concepts/
23. Las 6+2 formas normales de las bases de datos relacionales: https://picodotdev.github.io/blog-bitix/2018/02/las-6-plus-2-formas-normales-de-las-bases-de-datos-relacionales/
24. Curso de Introducción a ELLIoT Backend: https://bosonit-my.sharepoint.com/:f:/p/fernando_merino/EuVEHW7ikohFiwhYTtZ9lOgBz13x6BJ1wkujl1aWkIzPEw
25. Tutorial de ELLIoT Backend:
    - Sesión 1: https://bosonit-my.sharepoint.com/:v:/p/fernando_merino/EauXsC9cCm5Cg1XJOU8t7CgBeeYkeemfwWWA4o4FXrd8HA?e=6JCJye
    - Sesión 2: https://bosonit-my.sharepoint.com/:v:/p/fernando_merino/EbFN6yaOPkNMki9mqKmHIMgB_lwZT3Yoq6OhyIuC9wZJlA?e=gLaGGE
    - Sesión 3: https://bosonit-my.sharepoint.com/:v:/p/fernando_merino/EfbFv34biy9HoK8O2FHiUvAB5fzD1JTarzKhtg2zxF0MsQ?e=nRZxAh
    -
26. Curso Docker desde Cero https://bosonit-my.sharepoint.com/:f:/p/fernando_merino/ErCuNclIm_ZLpeR9jTlpnbUBn_UiySjCf1J4_rHrlC26OA
27. Desplegar con Gitlab, Portainer y Nginx: Desplegar con Gitlab, Portainer y Nginx.mp4
28. Organización inicial de un proyecto: Organización inicial de un proyecto.mp4
29. Configurar proxy de Grafana en backend: Configurar proxy de Grafana en backend.mp4

Cursos

GIT

Curso de Git, GitHub y SourceTree: https://bosonit-my.sharepoint.com/:f:/p/fernando_merino/EgANGgIuzPpLgKBKS0eJtOkBFDPgjJ-6ff7YDw1mkTCoDA

JAVASCRIPT

Certificación completa en JS de freeCodeCamp https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/

TYPESCRIPT

Curso de TypeScript - Aprender typescript desde cero https://www.youtube.com/watch?v=IJ_mpJRaHmc&list=RDCMUC3QuZuJr2_EOUak8bWUd74A

FRONTEND:

Certificación completa en diseño web responsive (HTML, CSS) de freeCodeCamp: https://www.freecodecamp.org/learn/2022/responsive-web-design/

Build your first Angular app: https://scrimba.com/learn/yourfirstangularapp

Curso Angular en Español - Tutorial de Angular 12 desde cero https://www.youtube.com/watch?v=i-oYrcNtc2s

Angular Material: https://www.youtube.com/watch?v=jUfEn032IL8

BACKEND:

Node.js and Express.js - Full Course: https://www.youtube.com/watch?v=Oe421EPjeBE

PostgreSQL & Nodejs REST API, Sequelize, Babel (Versión antigua 2019) https://www.youtube.com/watch?v=sA3t4d1v7OI

Sequelize REST API (usando Postgres) (NUEVA VERSIÓN 2022): https://youtu.be/3xiIOgYdbiE

Curso de Node, Express y MongoDB https://bosonit-my.sharepoint.com/:f:/p/fernando_merino/EjnCBV40mMtKlEYgfRNHRrUBd0aXttIw25T95deYGdMNKQ

Curso de Introducción a ELLIoT Backend: https://bosonit-my.sharepoint.com/:f:/p/fernando_merino/EuVEHW7ikohFiwhYTtZ9lOgBz13x6BJ1wkujl1aWkIzPEw

DEVOPS

Curso Docker desde Cero https://bosonit-my.sharepoint.com/:f:/p/fernando_merino/ErCuNclIm_ZLpeR9jTlpnbUBn_UiySjCf1J4_rHrlC26OA

Tutoriales
BACKEND:

PostgreSQL & Nodejs REST API, Sequelize, Babel (Versión antigua 2019) https://www.youtube.com/watch?v=sA3t4d1v7OI&t=3761s

Sequelize REST API (usando Postgres) (NUEVA VERSIÓN 2022): https://youtu.be/3xiIOgYdbiE

Guías
SEQUELIZE (Documentación de gestión de asociación entre tablas, importante lectura)

Associations: https://sequelize.org/docs/v6/core-concepts/assocs/

Advanced Association Concepts: https://sequelize.org/docs/v6/category/advanced-association-concepts/

Conventional commits: https://www.conventionalcommits.org/en/v1.0.0/

Markdown: https://www.markdownguide.org/basic-syntax/

Git Flow (Nota: nosotros usamos una versión más simple de este sistema acorde a nuestras necesidades, explicada en nuestro convenio de git y, por tanto, no instalamos git-flow): https://cleventy.com/que-es-git-flow-y-como-funciona/

Artículos
Las 6+2 formas normales de las bases de datos relacionales

https://picodotdev.github.io/blog-bitix/2018/02/las-6-plus-2-formas-normales-de-las-bases-de-datos-relacionales/

How to speed up Sequelize with complicated includes

https://maciek.cloud/how-to-speed-up-sequelize-with-complicated-includes/

Angular Parent to Child and Child to Parent communication from router-outlet

https://medium.com/@sujeeshdl/angular-parent-to-child-and-child-to-parent-communication-from-router-outlet-868b39d1ca89

Arquitectura de 3 módulos en Angular

https://tomastrajan.medium.com/how-to-build-epic-angular-app-with-clean-architecture-91640ed1656

Introduction to "reflect-metadata" package and its ECMAScript proposal

Para empezar a introducirse en el manejo de metadatos en clases y su aplicación con decoradores en la programación orientada a aspectos (AOP), base de Angular y NestJS, que permiten una programación mucho más avanzada, cómoda y versátil, pero que hay que entender.

Es interesante seguir los enlaces del artículo a sus otros artículos para ir uniendo cabos.

Truco: Para leer artículos de Medium sin límites hay que usar la ventana privada o de incógnito

https://medium.com/jspoint/introduction-to-reflect-metadata-package-and-its-ecmascript-proposal-8798405d7d88

Artículos de interés y explicaciones sobre arquitectura del código en proyectos

Os comparto unos artículos interesantes y unas explicaciones para entender mejor el por qué y el cómo organizar el código en capas, para que resulte más fácil entender el por qué separamos en controlador, servicio y repositorio las capas del backend.

Los principios en los que se basa son los de la arquitectura de tres capas y la arquitectura hexagonal. Estos son los artículos:

- Arquitectura de 3 capas

https://www.codementor.io/@evanbechtol/node-service-oriented-architecture-12vjt9zs9i
https://dev.to/santypk4/bulletproof-node-js-project-architecture-4epf

- Arquitectura hexagonal https://khalilstemmler.com/articles/enterprise-typescript-nodejs/clean-nodejs-architecture/

El resumen rápido es:

- El controlador recibe los datos y parámetros del cliente y se los pasa al servicio a través de uno o más métodos (según la complejidad de la petición) para poder obtener una determinada información con la que generar una respuesta a enviar. Es una implementación específica. en nuestro caso es un controlador HTTP que usa Express como dependencia para facilitar su creación.
- El servicio es donde reside la lógica de negocio de la aplicación. Si sustituimos controlador y repositorio por otras implementaciones, la lógica de negocio no se ve afectada y sigue pudiendo funcionar mientras se pueda conectar a cualquier controlador que le pase los argumentos que necesita, y pueda conectarse a una capa de persistencia de cualquier tipo que funcione usando los mismos métodos y argumentos de comunicación. Este es el motivo por el que las interfaces (y la intención de introducir TypeScript en backend) son útiles para definir claramente de que propiedades y métodos se compone un tipo, y así sustituir una implementación por otra sea como montar y desmontar un Lego (cambio un controlador HTTP por un controlador de consola de comandos o de MQTT, o cambio un repositorio de PostgreSQL por uno de MongoDB o uno que guarde directamente en RAM, como un simple array o Redis). Los servicios puedes usar otros servicios para delegar tareas específicas compartidas o compartibles entre servicios, como hashear contraseñas, generar tokens, acceder a otros recursos (p ej, el servicio de usuarios necesita hacer consultas al servicio de roles), etc.
- El repositorio es la capa encargada de interactuar con la capa de persistencia (BBDD, memoria, archivos...). Puede usar directamente los comandos directos de comunicación con ella, como el lenguaje SQL, o valerse de paquetes que le abstraigan el trabajo, como un ORM tipo Sequelize.

Respecto a la arquitectura hexagonal, nos basamos en un modelo hexagonal más simple que el general, porque hay que meter solo la complejidad necesaria en un diseño para no sobrecargar. La arquitectura serían solo dos capas:

- El núcleo central, que sería la capa de Dominio
- Una capa superior, que sería la capa de infraestructura Este sería un ejemplo: https://khalilstemmler.com/img/blog/clean-architecture/group.svg

El servicio sería el dominio o núcleo de la aplicación, donde va toda la lógica, mientras que controlador y repositorio son la infraestructura, implementaciones específicas para que la aplicación pueda comunicarse con el cliente (controlador HTTP con Express para comunicación con el cliente vía API) y con la capa de persistencia de datos (repositorio PostgreSQL con ORM Sequelize para comunicación con la capa de persistencia vía SQL)

Iremos organizando estas explicaciones en tutoriales más detallados, pero esta es la base y fundamentos que han de conocerse.

Para cualquier duda, no tenéis más que preguntar

Portainer
Una vez se ha aprendido a usar Docker en consola, se puede aprender a usar la aplicación web Portainer para gestionar contenedores Docker.

Se puede levantar en local en el ordenador con Docker. Estos son los comandos a ejecutar en consola (Windows, para otros SO preguntadme):

```
docker volume create portainer_data
docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data
portainer/portainer-ce:2.11.1
```

Una vez este corriendo, se puede acceder a el a través de https://localhost:9443. El navegador advertirá de que el certificado SSL no es seguro, simplemente hay que incluirlo como excepción. Después os pedirá usuario y contraseña a crear, se puede poner algo fácil como admin y 1234, porque es sólo para uso personal en local.