**Angular desde CERO con BD PostGreSQL**

Pasos para Instalar Angular:

* Dirigirse a la página de angular ( <https://angular.io/cli>)
* Primero también entrar a la pagina de node.js (<https://nodejs.org/en/>)
  + Una vez instalado se debe ejecutar los siguientes comandos:
    - Get-ExecutionPolicy Restricted
    - set-executionpolicy remotesigned
* Una vez instalado node.js, puedes usar node y npm
  + Ver la versión de node: node -v
  + Ver la versión de npm: npm -v
* Instalar angular con la siguiente orden:
  + npm install -g @angular/cli (Permite crear Proyecto, generar aplicaciones y código de biblioteca)
* Crear un Proyecto en angular con la siguiente orden:
  + ng new nuevoproyecto
* Ejecutar proyecto:
  + ng serve –o O ng s --o
* Crear un componente en angular:
  + ng g c nomcomponente --skip-tests o skiptests (version anterior a 14)
* Instalar bootstrap:
  + npm i bootstrap@5.3 .0-alpha1
* Una vez instalado bootstrap se debe hacer parte en el archivo “angular.json”, en la sección “styles”:
  + node\_modules/bootstrap/dist/css/bootstrap.min.css
* Se usa Formulario Reactivo, para ello se debe escribei en el archivo “app.module.ts” en la sección imports: lo siguiente
  + FormsModule y ReactiveFormsModule

Crear repositorios Locales y en la Nube

Repositorio Local:

* Ir a consola de comandos dentro del proyecto
  + git init

Crear una copia del repositorio:

* git add . (guarda en la memoria)
* git commit -m “Nombre de la versión”

Subir a la nube el repositorio:

**Creación de Login con Angular y Node**

***Api Rest***: es un estándar un conjunto de reglas especificaciones que nos permite transmitir datos entre un cliente y un servidor, entre servidor y servidor, mediante el protocolo http (es decir, GET, POST, PUT y LIT, etc en formato json).

**Creación de BackEnd – instalación de paquetes y configuraciones iniciales**

1. Inicializar proyecto (se debe tener instalado el NodeJs)
   1. npm init -y
2. Instalar paquetes:
   1. express (framework del api rest).
   2. bcrypt (permite encryptar las contraseñas)
   3. cors (para que maneje puertos del fronend y backend)
   4. dotenv (permite configurar las variables de entorno)
   5. jsonwebtoken (permite crear el token)
   6. **sequelize** (es un moderno ORM es un modelo de programación que permite mapear las estructuras de una base de datos relacional (SQL Server, Oracle, MySQL, Postgresql, etc)(link: https://sequelize.org/docs/v6/getting-started/).
   7. ***Postman*** es una plataforma API para construir y usar API. Postman simplifica cada paso del ciclo de vida de la API y agiliza la colaboración para que pueda crear mejores API, más rápido (link: https://www.postman.com/)
   8. npm install express bcrypt cors dotenv jsonwebtoken sequelize pg pg-hstore
3. Instalar Typescript:
   1. npm install typescript --save-dev
4. Instalar nodemon (escucha los cambios y actualiza los cambios):
   1. npm install nodemon --save-dev
5. Inicializar Proyecto de Typescript (crea un archivo de configuración):
   1. npx tsc –init (crea el archivo tsconfig.json, en la linea cambiar “outDir”: “./dist”)
6. Crear la carpeta “src” donde vamos a contener toda la aplicación y codigo
   1. Se crea el archivo index.ts
7. Transquilar codigo “ts a js” para eso se escribe:
   1. npx tsc (crea la carpeta dist con el archivo index.js)
8. Ejecutar archivo index.js
   1. node dist/index.js
9. Ejecutar con nodemon (ejecuta tipo servidor)
   1. npx nodemon dist/index.js (verifica y actualiza los cambios en la carpeta dist y archivo index.js)
10. Comando para actualizar archivo ts y archivo js
    1. npx tsc –watch (verifica y actualiza el archivo index.ts)
11. Configurar los script en el archivo “package.json”, para hacer correr “nodemom y typescript”
    1. “dev”: npx nodemon dist/index.js (ahora: **npm run dev**)
    2. “typescript”: npx tsc --watch (ahora: **npm run typescript**)

**Estructura de Carpetas y Lanzamiento del servidor**

1. Se crean dentro la carpeta “src”, las siguientes subcarpetas:
   1. db, models, routes, controllers
2. Se crea el archivo .env donde se declaran las variables de entorno

**Configuración de rutas**

1. Instalar json viewer (<https://chrome.google.com/webstore/detail/json-formatter-viewer-and/infnlhnhibphpaljmnnadaldibggkokb/related>)

**Creación de modelos y getproducts**

1. Utilizar sequelize para el modelo de datos, existen 3 maneras de usar la función “sync” de sequelize:
   1. tabla.sync(): Esto crea la tabla si no existe (y no hace nada si ya existe).
   2. tabla.sync(forcé: true): Esto crea la tabla, soltándola primero si ya existía.
   3. tabla.sync(alter: true): Esto verifica cuál es el estado actual de la tabla en la base de datos (qué columnas tiene, cuáles son sus tipos de datos, etc.) y luego realiza los cambios necesarios en la tabla para que coincida con el modelo.

**Creación de Usuarios**

1. Ctrl – space: invoca listado
2. Ctrl - .: invoca librería o clase
3. Instalar bcrypt para encriptar el password (npm i --sabe-dev @types/bcrypt )
4. Cuando sale una “promesa” se pone “await”

**Login Usuario**

1. Ingresar a jwt para generar el token (<https://jwt.io/>), la importancia de generar el token es para verificar si la persona es la persona, tiene tres (3) partes:
   1. Cabecera
   2. Payload data
   3. Firma de verificación
2. Instalar jwt (npm i --sabe-dev @types/jsonwebtoken)

**Proteger Rutas**

1. Sea visible para usuarios que tengan token valido

**Creación del Front Instalación de Bootstrap, estructura del proyecto**

1. Crear el proyecto en angular:
   1. ng new nom\_proy
   2. (yes)(css)
2. Hacer correr el servidor en angular:
   1. ng serve --o
3. Configurar Bootstrap:
   1. Instalar boostrap: npm i [bootstrap@5.3.0-alpha1](mailto:bootstrap@5.3.0-alpha1)
   2. Hacer una referencia e el archivo “angular.json” en la sección “styles”, copiar la ruta: (node\_modules/Bootstrap/dist/css/bootstrap.min.css)
4. Crear la estructura de carpetas dentro de “app”:
   1. components, services, shared, utils, interfaces
   2. crear componentes con: ng g c components/login

**Login HTML**

1. manejar flexbox (<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>)
2. colocar logo con (<https://www.flaticon.es/>)
3. Para crear archivos estáticos se crea dentro de la carpeta “assets”

**Ruteo y registrar usuario HTML**

1. Crear ruteo a través del archivo “app-routing.module.ts”: en la sección const routes: Routes.

**Dashboard HTML**

**SignIn**

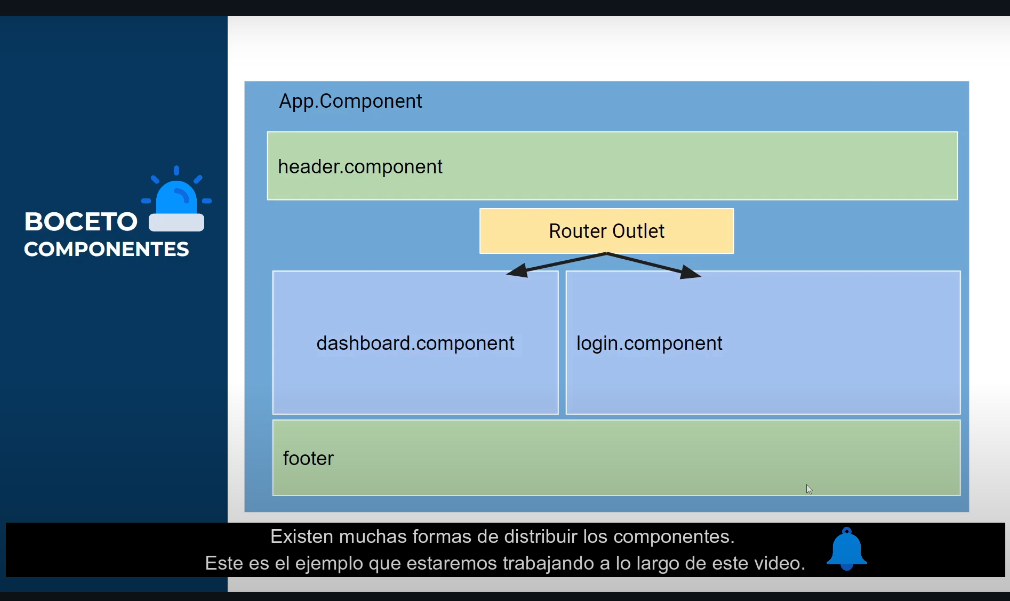
1. Services: el objetivo de los servicios son tres (3):
   1. Hacer las peticiones hacia el backend
   2. Reutilizar código
   3. Comunicación de datos mediante obsérvales
2. Crear services:
   1. ng g s services/user
3. Para hacer las peticiones HTML si o si se debe importar el modulo httpClientModule
4. Formularios reactivos (definicion).
5. Para usar ngModel, se debe importar un modulo
6. Utilizar para mensajes el toasts angular (link: <https://www.npmjs.com/package/ngx-toastr>).
   1. Instalar toastr para angular: npm install ngx-toastr --save.
   2. Instalar Tambien las animaciones: npm install @angular/animations --save
   3. Para usar estas animaciones se tiene que comocar en el archivo “angular.json” en la sección de “styles” lo siguiente: "node\_modules/ngx-toastr/toastr.css"
7. Configurar puertos para que se vean:
   1. Se debe instalar para declarar los cors: npm i --save-dev @types/cors

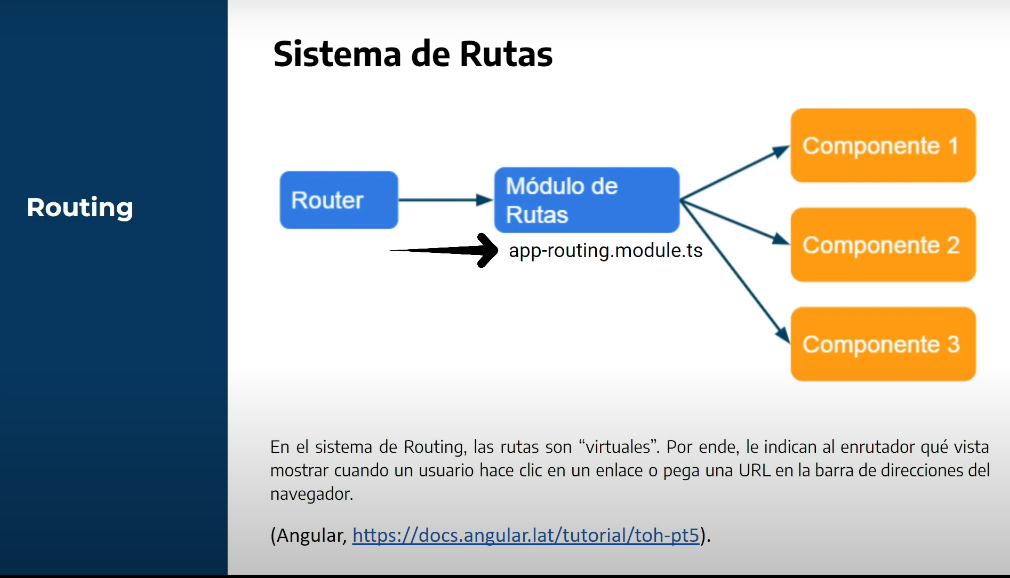
**SignIn Mensajes**

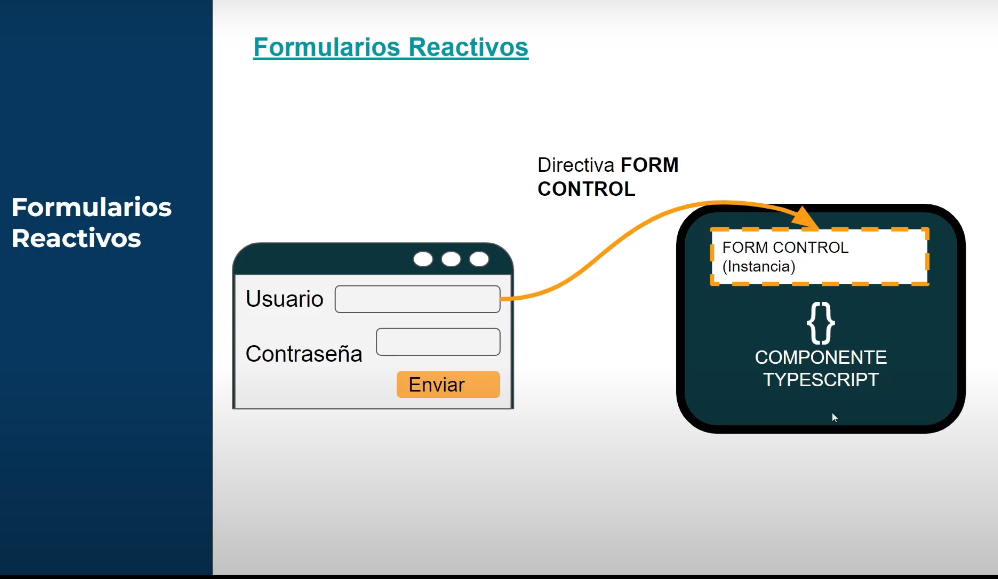
**Interceptor**

**Guard**

**List Products**

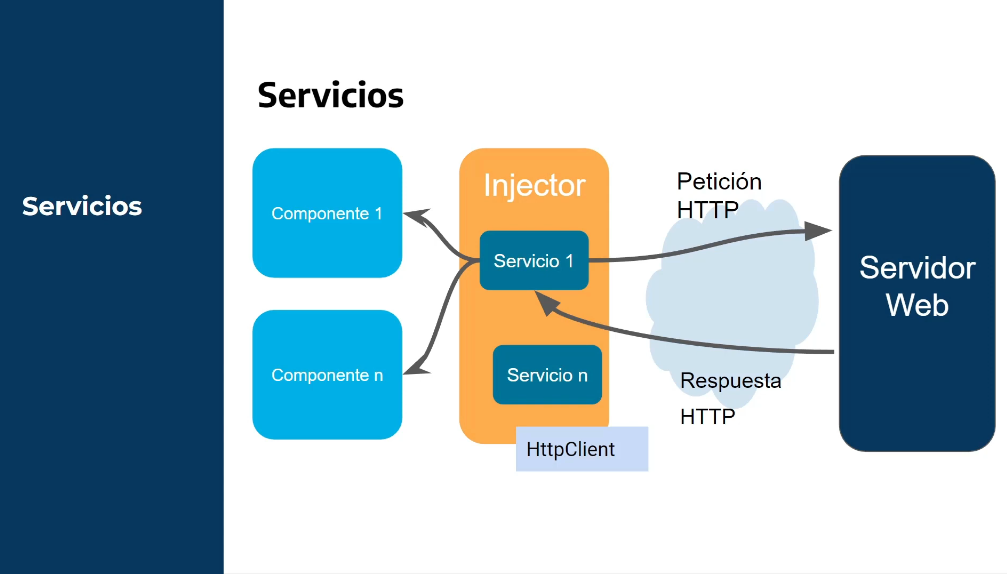


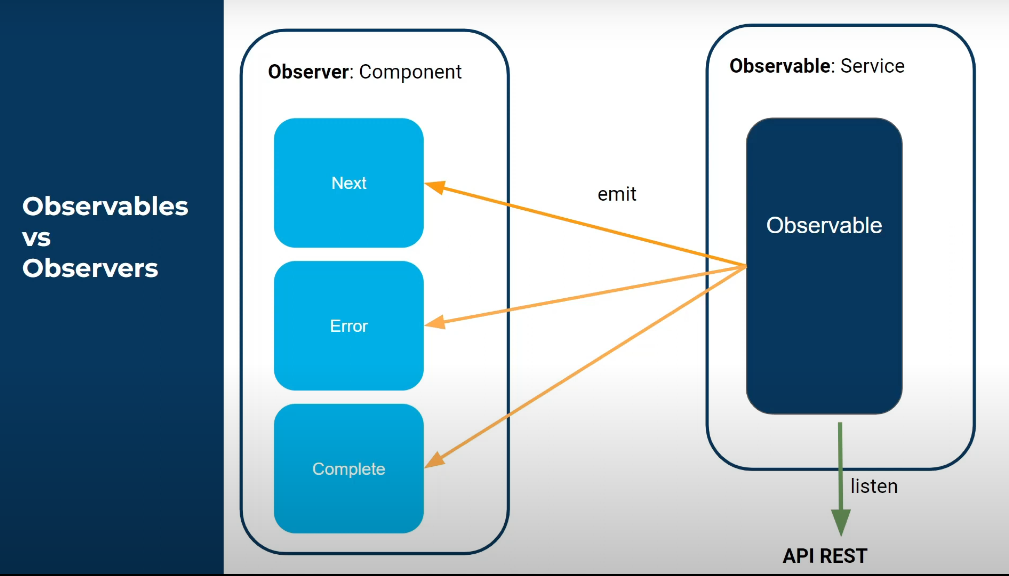


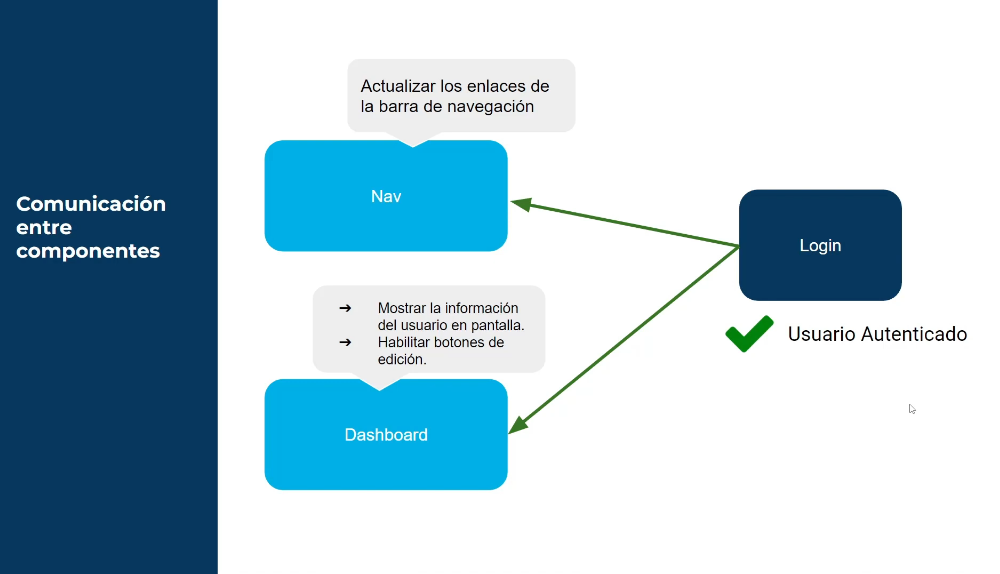


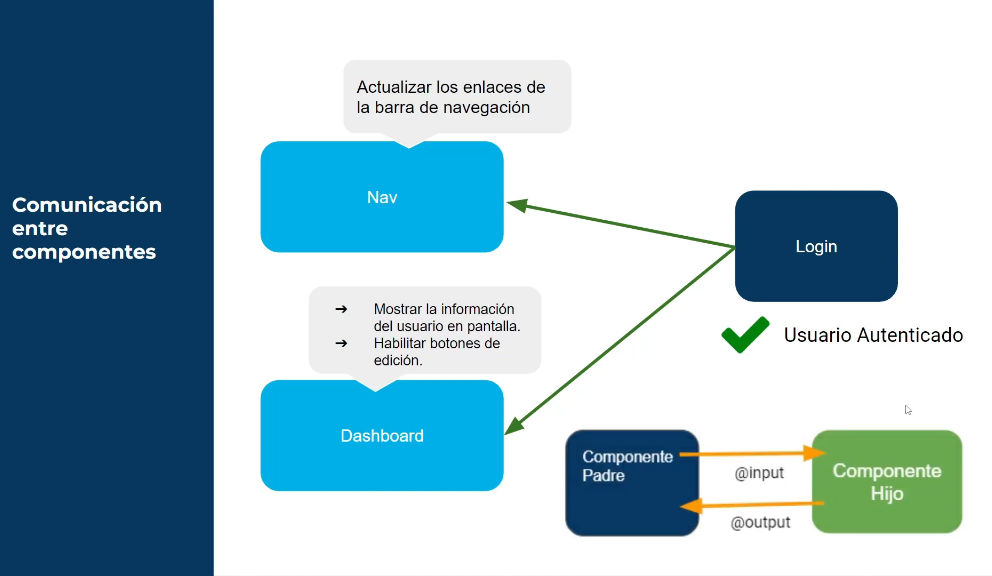


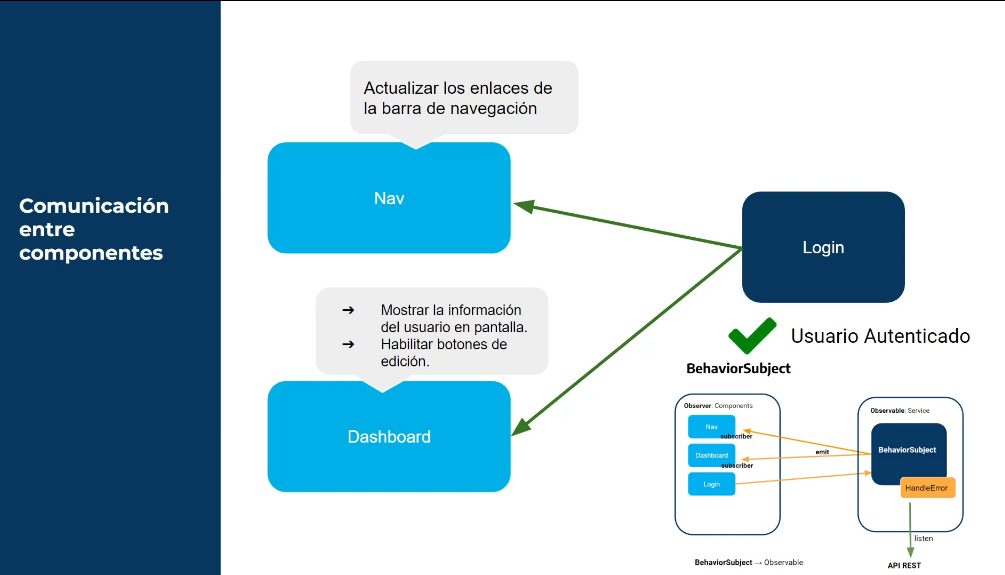


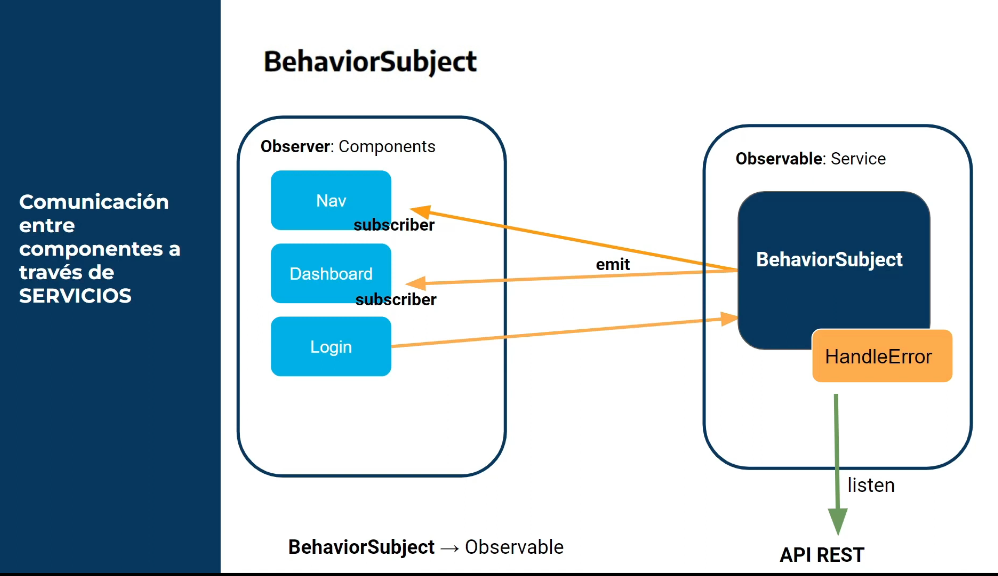


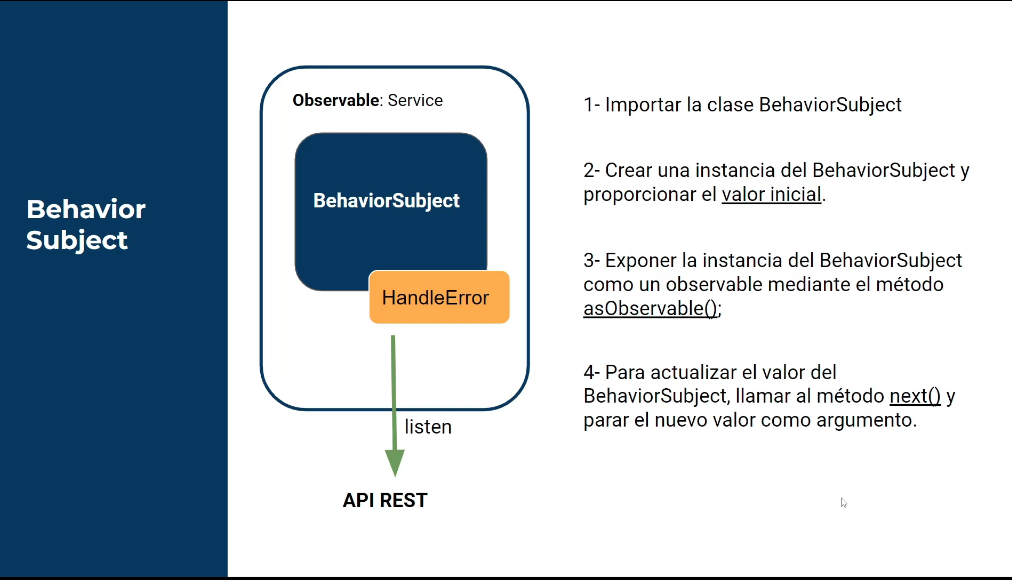












Armar un proyecto angular 17

* Archivo “app.component.js”, en este archivo se llama a los componentes:
  + Header
  + Footer
  + RouterOutlet
  + ReactiveFormsModule
  + HttpClientModule