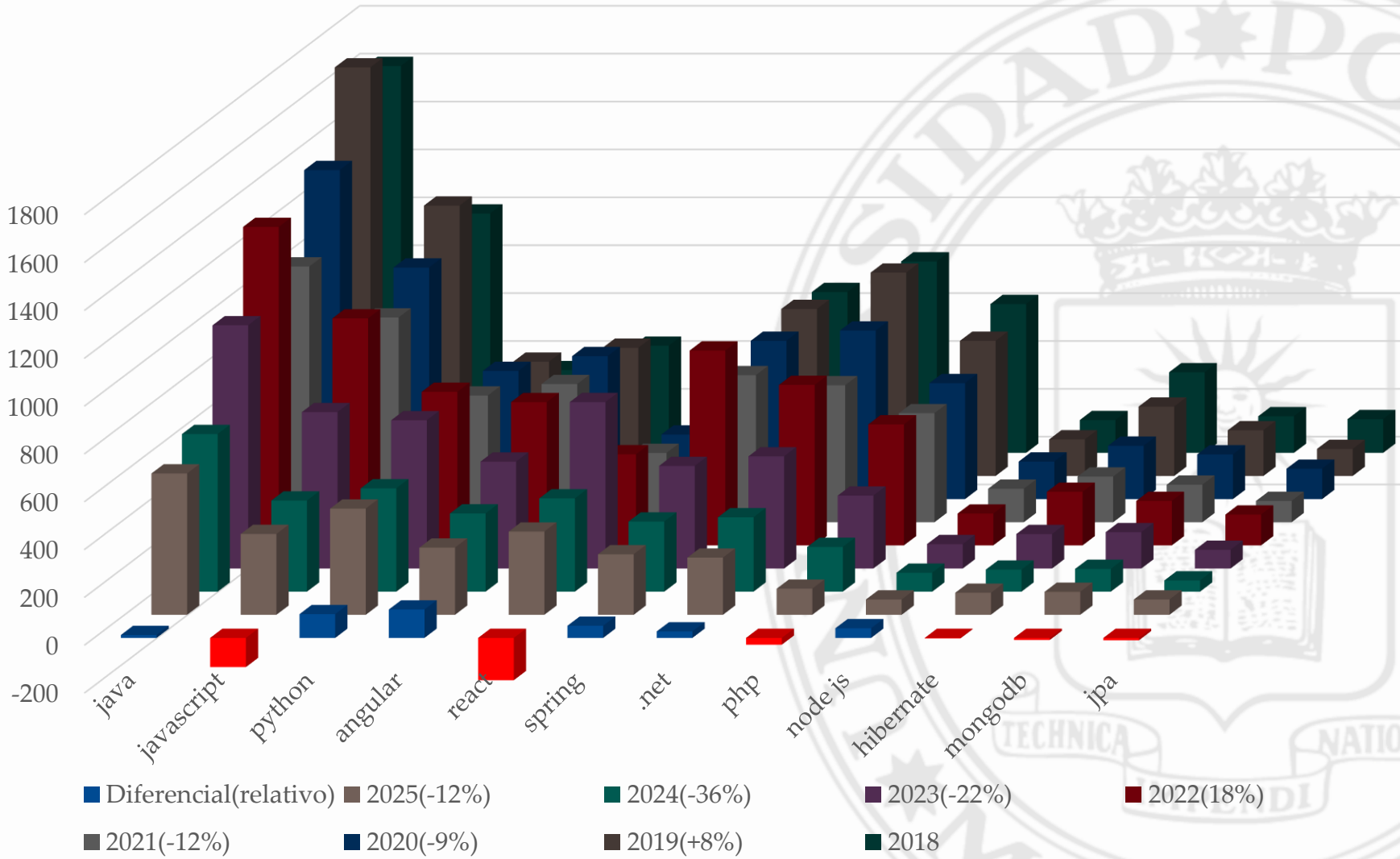
Back-end con Tecnologías de Código Abierto -BETCA-

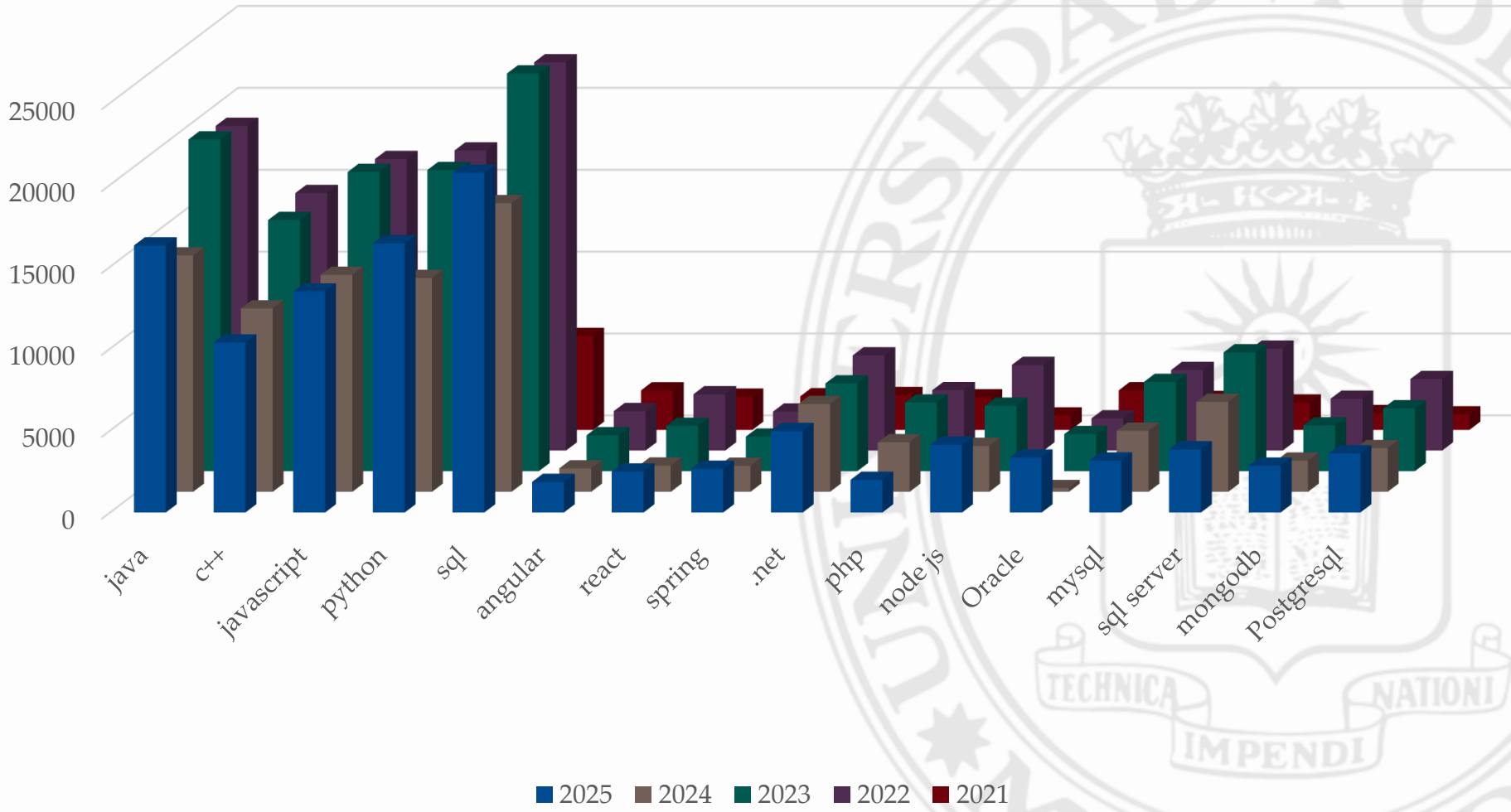
Spring

Jesús Bernal Bermúdez

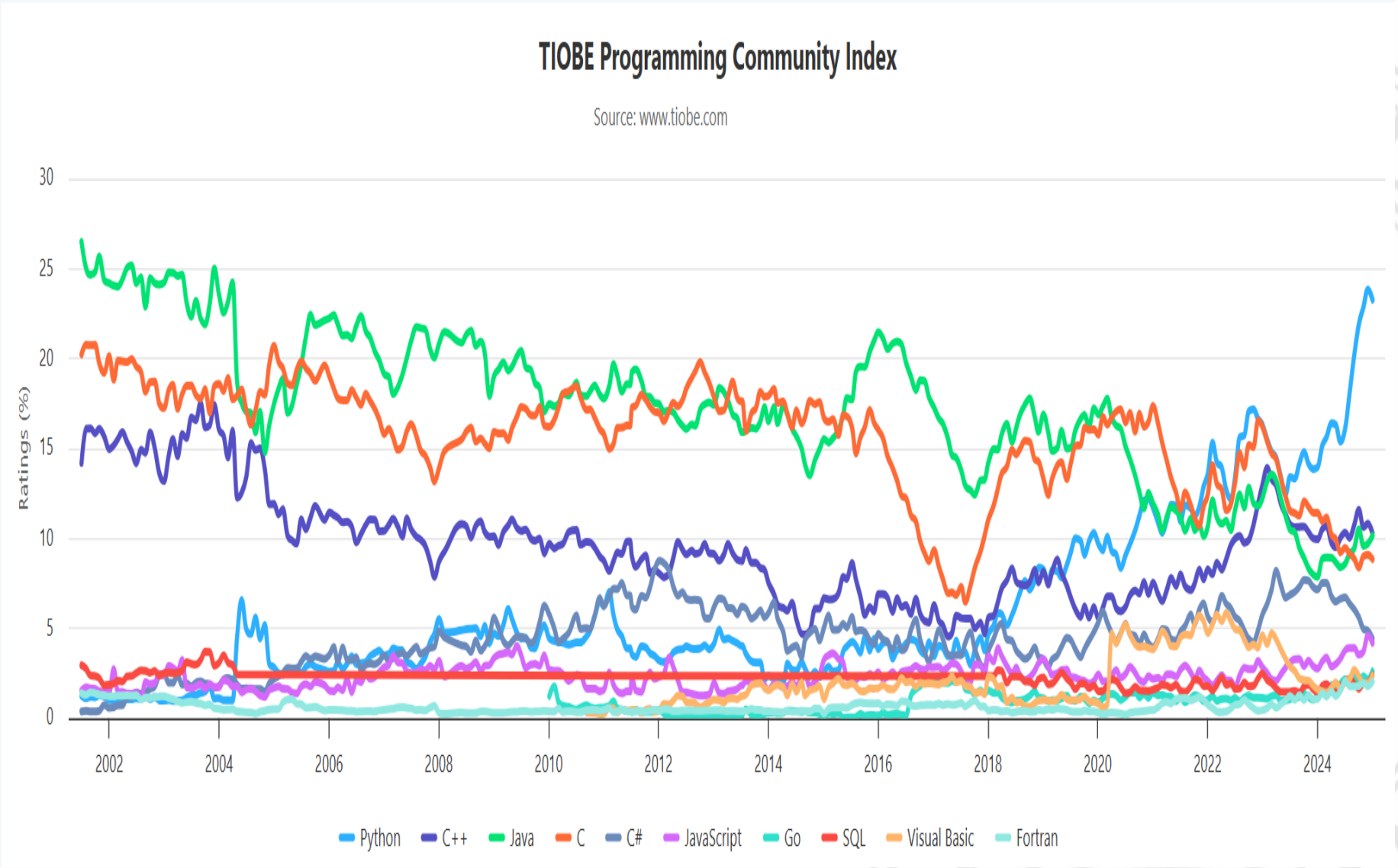
Tendencia del Mercado
InfoJobs (03-02-2025)



Tendencia del Mercado
Linkedin (03-02-2025)



Tendencia del Mercado
TIOBE (03-02-2025)



BETCA

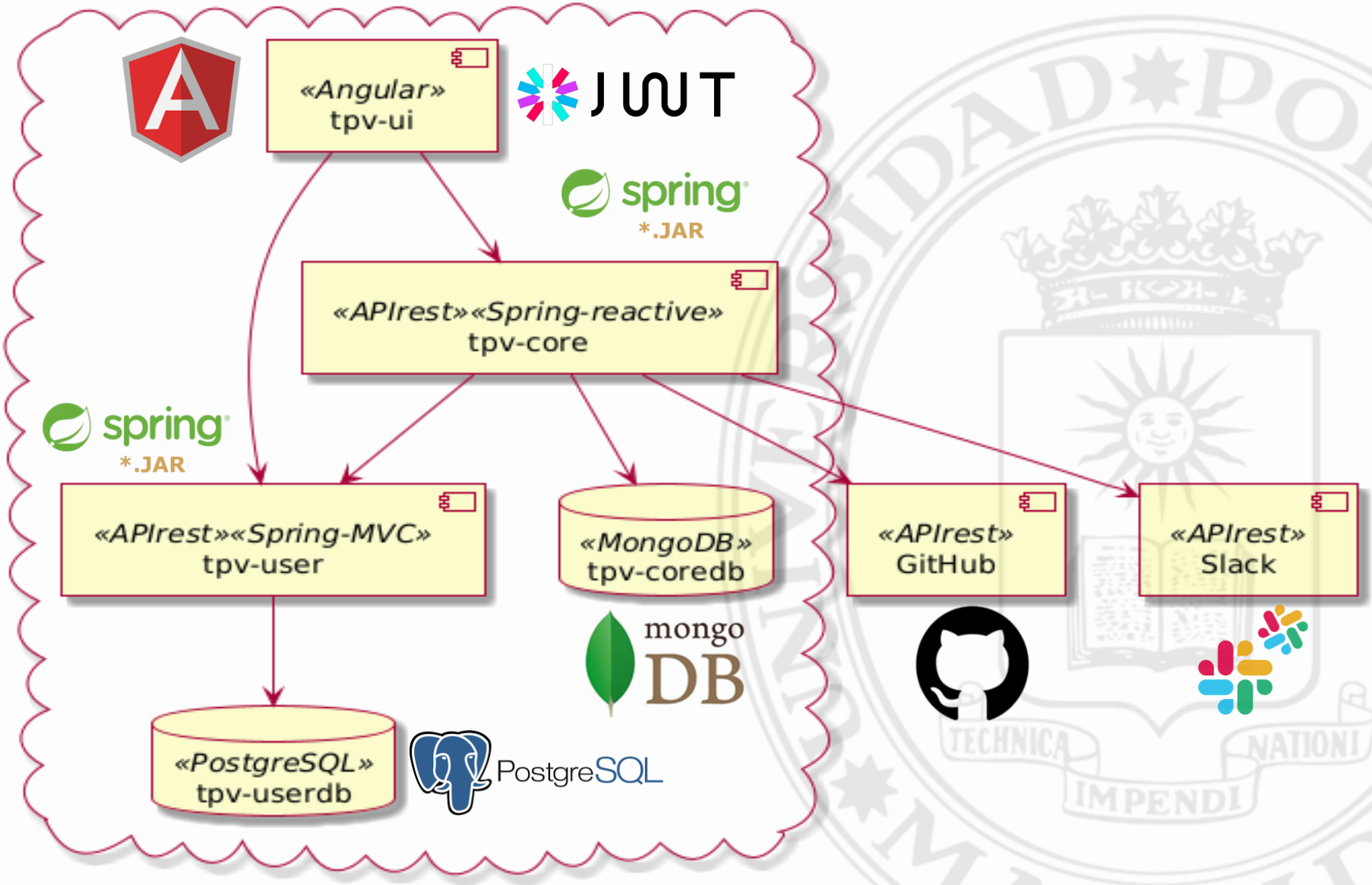
Back-end con Tecnologías de Código Abierto

Aplicación Web

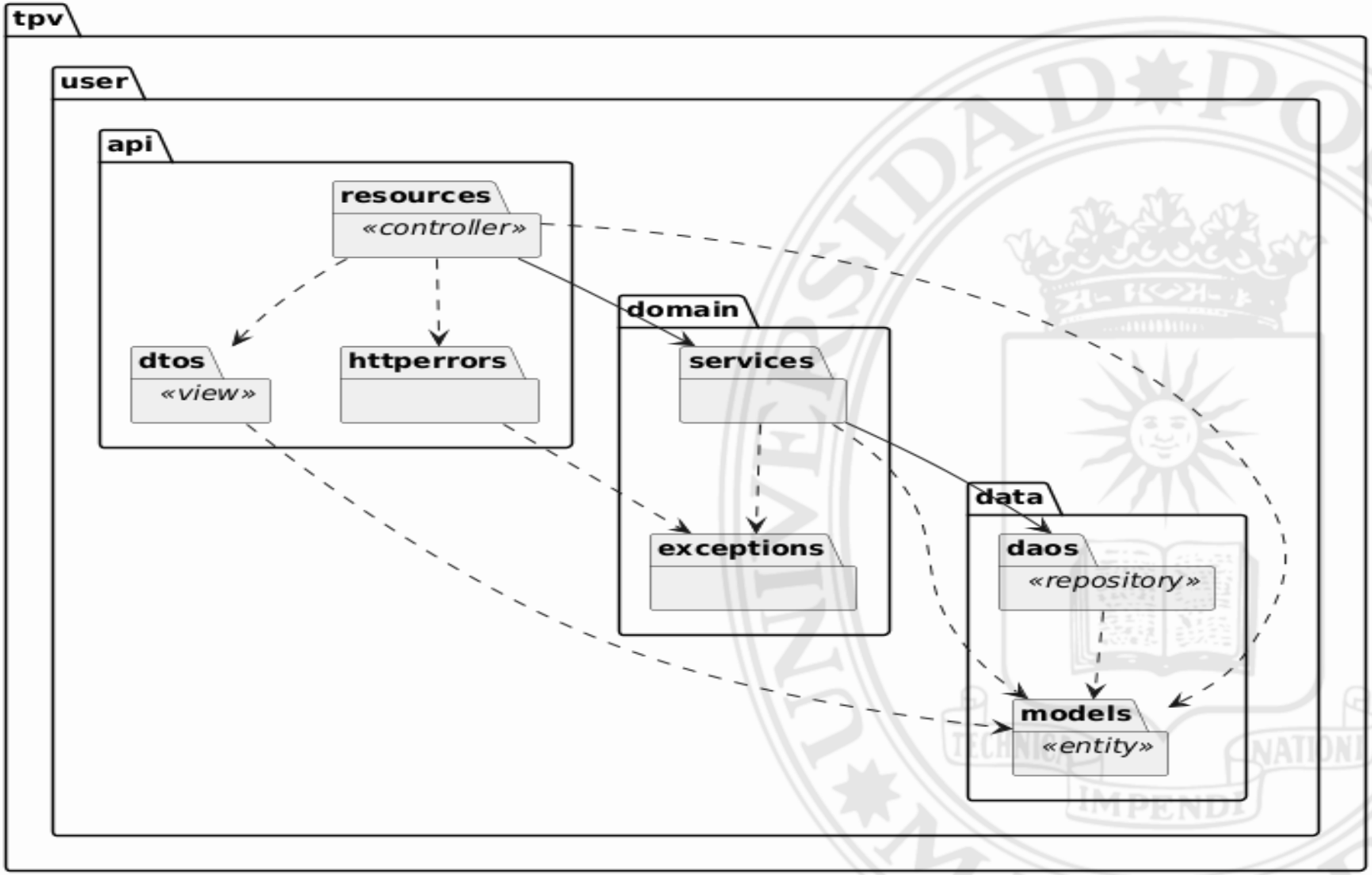
Full-Stack

- Front-end: *Typescript-Angular (reactivo)*
- Back-end: *Java-Spring: MVC (funcional)*
- Back-end: *Java-Spring: Rective-Funcional*
- Database: *PostgreSQL & Mongodb*

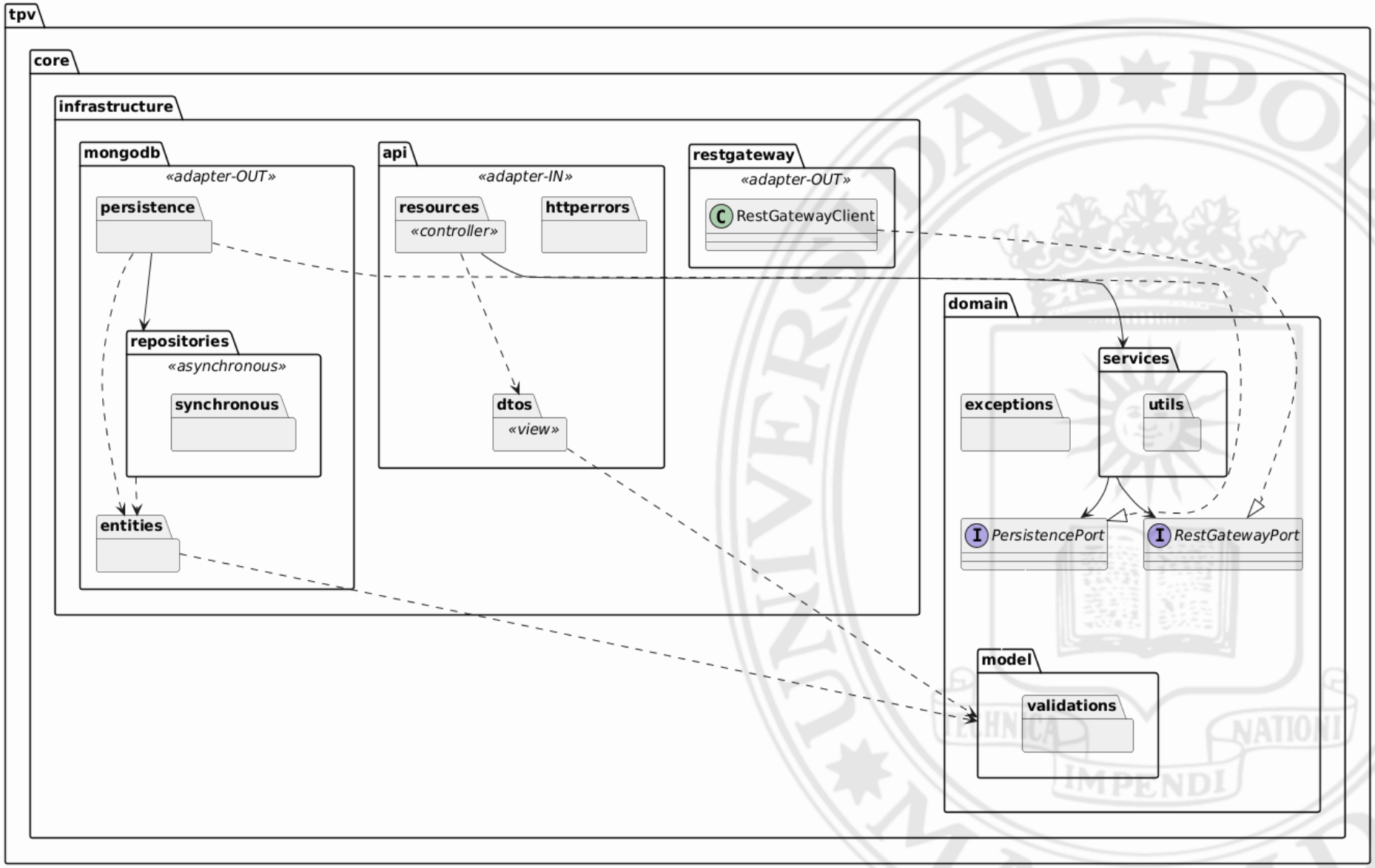
BETCA
TPV

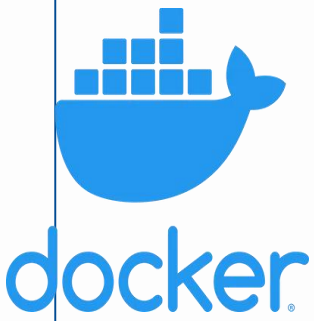

«Angular»
tpv-ui
JWT
spring
*.JAR
«APIrest»«Spring-reactive»
tpv-core
spring
*.JAR
«APIrest»«Spring-MVC»
tpv-user
«MongoDB»
tpv-coredb
mongo
DB
«APIrest»
GitHub
«APIrest»
Slack
«PostgreSQL»
tpv-userdb
PostgreSQL

BETCA
TPV-USER



BETCA
TPV


tpv
core
infrastructure
mongodb
«adapter-OUT»
persistence
repositories
«asynchronous»
synchronous
entities
api
«adapter-IN»
resources
«controller»
httperrors
dtos
«view»
restgateway
«adapter-OUT»
RestGatewayClient
domain
services
exceptions
utils
PersistencePort
RestGatewayPort
model
validations



Docker

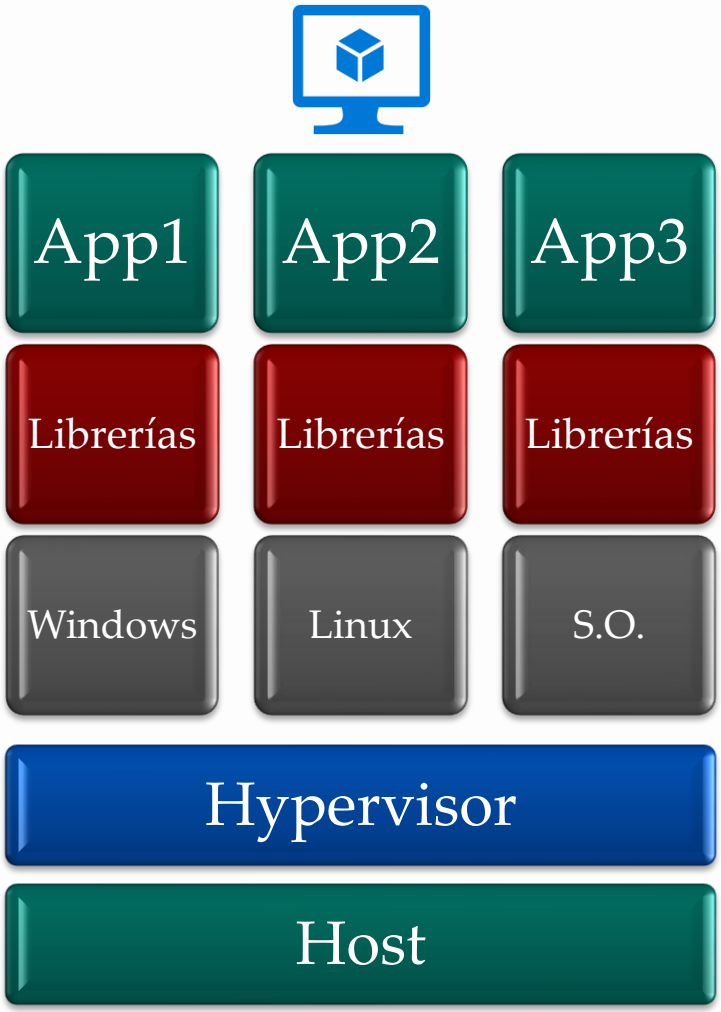
- Docker es un proyecto de código abierto que automatiza el **empaquetamiento, distribución y ejecución** de aplicaciones dentro de contenedores de software.
- Es una plataforma que simplifica el desarrollo y despliegue de aplicaciones al hacerlas portables, eficientes y escalables.
- Es una herramienta clave en ***DevOps***.
- ***https://www.docker.com/***

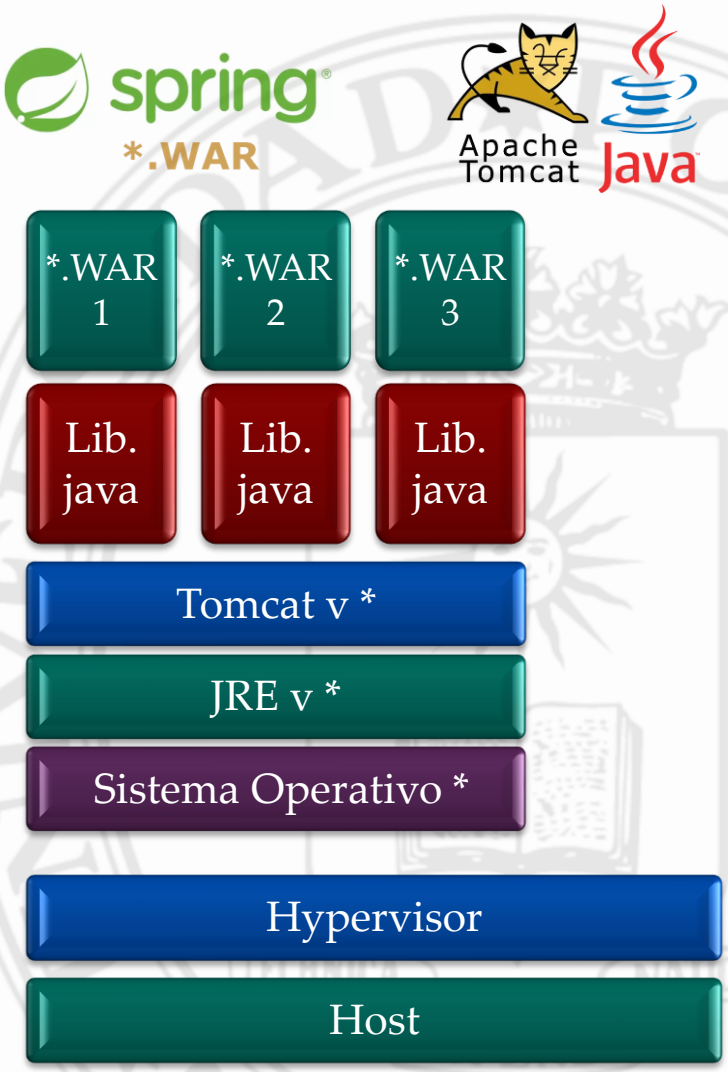
Docker Hub

- Repositorio de imágenes de Docker
- ***https://hub.docker.com/***

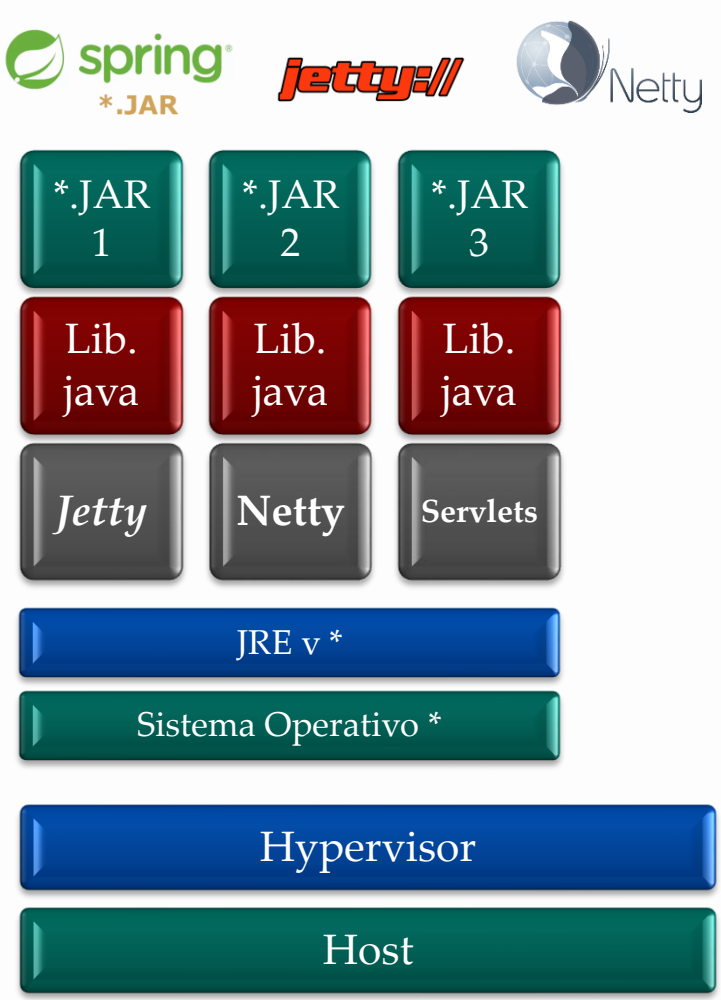
BETCA

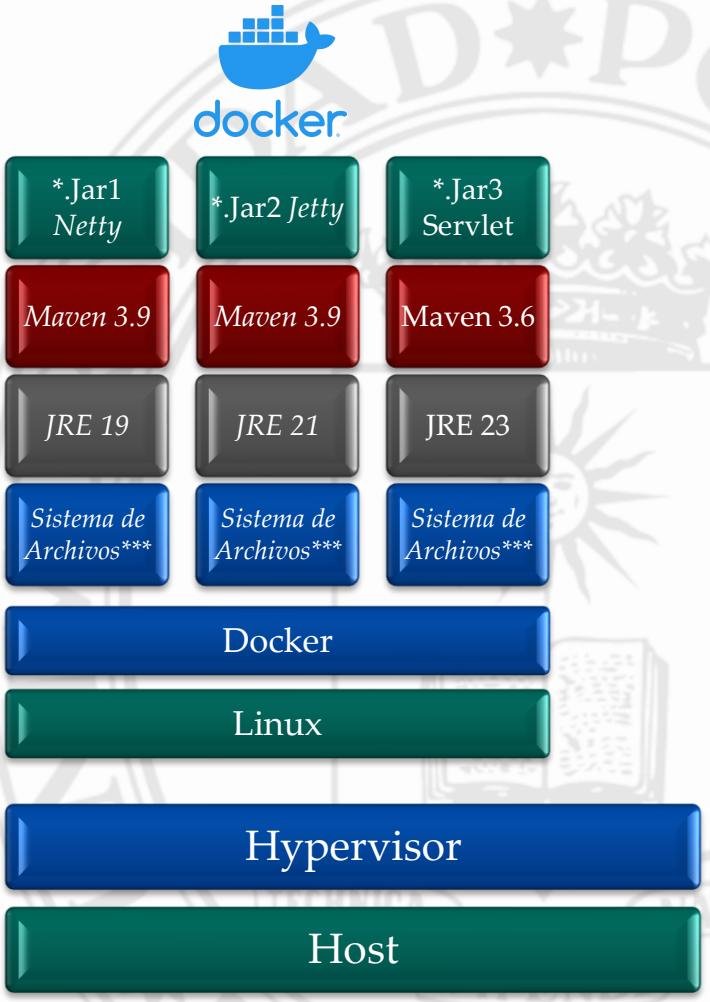
Host VS Virtual Machine & WAR



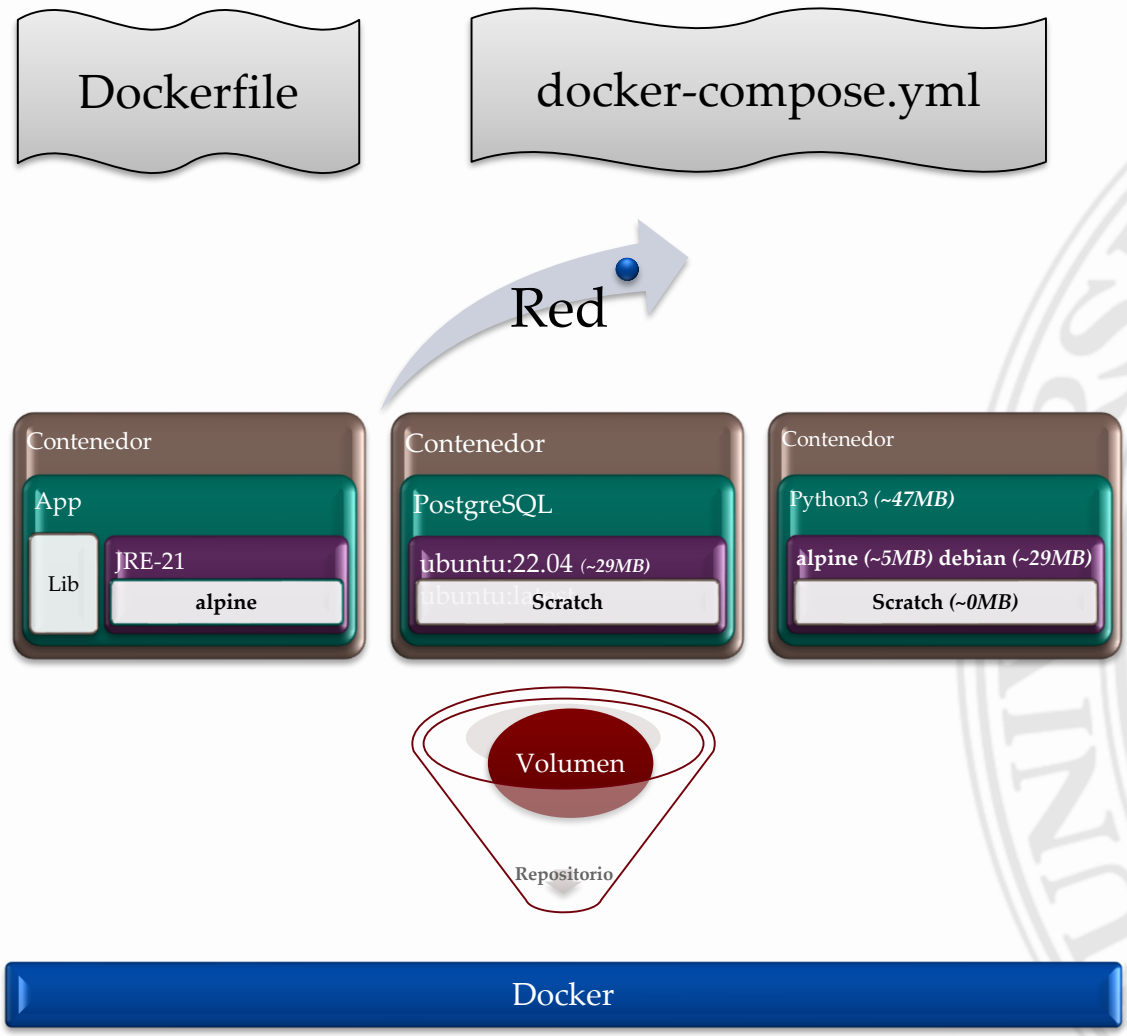


BETCA
Jar & Docker





BETCA
Docker. Conceptos



Docker

- **Imagen**. Es una plantilla inmutable que contiene lo necesario para ejecutar una aplicación (*código, dependencia, configuración*…). Se construyen a partir de un fichero *Dockerfile* de tipo texto.
  - *Imagen* **scratch**. Es el origen
- **Contenedor**. Unidad de software empaquetado para su ejecución de una imagen. Puede contener *variables de entorno*. Se puede limitar el uso de memoria y CPU.
- **Red**. Docker pude crear redes para que los contenedores se puedan comunicar entre si.
- **Volumen**. Son almacenamientos persistentes de datos y sobreviven al ciclo de vida de los contenedores.
- **Docker compose**. Es una herramienta para definir y ejecutar múltiples contenedores, se define mediante el archivo *docker-compose.yml*.
- ***Kubernetes***. Es una herramienta para para administrar los contenedores en producción.

BETCA
Commandos

Instalación

- https://www.docker.com/
- https://hub.docker.com/

Comandos

• docker *comando* --help	# Muestra la ayuda del comando indicado
• docker pull postgres[:15.10]	# Se baja una imagen, si no se pone versión sería *latest*
• docker images	# Muestra las imágenes
• docker image rm <id>	# Borra una imagen
• docker build -t <imagen:version> .	# Construye una imagen a partir de *Dockerfile* --tag, o -f filename
• docker create --name<name> <image>	# Crea un contenedor a partir de una imagen
• docker ps [-a]	# Muestra los contenedores --all
• docker rm <id>	# Borra un contenedor
• docker start -a -i <id>	# Ejecuta un contenedor, -a --attach, -i:--interactive
• docker stop <id>	# Para un contenedor en ejecución
• docker run -d -p 8080:8080 <image>	# Construye un contenedor y ejecuta a partir de imagen [-it] –interactive & --tty
• docker logs [-f] <id>	# Muestra los logs de un contenedor
• docker compose up	# Construye, (re)crea, inicia y adjunta a partir de *docker-compose.yml*
• docker compose down	# Detiene y elimina
• docker volumen ls	# Lista los volúmenes existentes
• docker network ls	# Lista las redes
• …	

BETCA
Dockerfile

Fichero: *Dockerfile*, en la raíz del proyecto

- Tipo txt

```
# Etapa1
FROM maven:3.9.9-eclipse-temurin-21 AS build
WORKDIR /app
COPY pom.xml ./
RUN mvn dependency:go-offline –B
COPY src ./src
RUN mvn clean package –DskipTests

# Etapa2
FROM eclipse-temurin:21-jre-alpine
WORKDIR /app
COPY --from=build /app/target/*.jar app.jar
EXPOSE 8081
CMD ["java","-jar","app.jar"]
```

Parte de una imagen que combina maven(3.9.9) & JDK(21)

Establece el directorio de trabajo dentro del contenedor

Copia el *pom.xml* de la ruta raíz del proyecto a la carpeta de trabajo del contenedor

Ejecuta el comando maven para que se descargen las dependencias en el contenedor

Solo copia los fuentes del proyecto al contenedor

Ejecuta el comando maven para crear el *.jar

Parte de una imagen JRE(21)

Establece el directorio de trabajo dentro del contenedor

Copia el *.jar generado del contenedor de construcción

Habilita el puerto 8081 para que sea escuchado

Establece un comando para cuando se inicialice el contenedor: java –jar app.jar

BETCA
YAML

YAML

- Key: Value
- Structure: “whitespace” o “{“
- List “-” o “[*, *, *]”
- Comments “#”
- Single-line “>” o Multi-line “|”

test-sonar.yml

```yaml
name: DevOps - Tests
on:
  push:
    branches: ['develop', 'release-*']
jobs:
  test:
    name: Test - Unit & Integration & SonarCloud Scan
    runs-on: ubuntu-latest # macos-latest macos-11 wi
    steps:
      - uses: actions/checkout@v2
        with: {fetch-depth: 0}
      - uses: actions/setup-java@v1
        with:
          java-version: 11
      - name: Unit & Integration Tests
        run: | # -B: --batch-mode
          mvn clean
          mvn -B verify
      - name: Sonar
        if: success() # always() failure() success()
        run: >
          mvn -B verify -DskipTests sonar:sonar
          -Dsonar.projectKey=es.upm.miw:iwvg-devops
          -Dsonar.organization=miw-upm-github
          -Dsonar.host.url=https://sonarcloud.io
          -Dsonar.login=$SONAR_TOKEN
```

BETCA

docker-compose.yml

```yaml
version: '3.9'
services:
  database:
    image: postgres:15.10
    container_name: postgres-db
    restart: always
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: tpv
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
    labels:
      - "es.upm.miw.betca-tpv-user=database"
  app:
    build: .
    container_name: java-app
    restart: always
    depends_on:
      - database
    ports:
      - "8081:8081"
    environment:
      SPRING_DATASOURCE_URL: jdbc:postgresql://database:5432/tpv
      SPRING_DATASOURCE_USERNAME: postgres
      SPRING_DATASOURCE_PASSWORD: postgres
      SPRING_JPA_HIBERNATE_DDL_AUTO: update
    labels:
      - "es.upm.miw.betca-tpv-user=app"
volumes:
  pgdata:
    name: "postgres-data"
```

Versión de Docker utilizada

Contenedor

Indica que cuando el contenedor de error, o se reinicie el host, se arranque de nuevo

Facilita la organización y gestión de contenedores

Contenedor

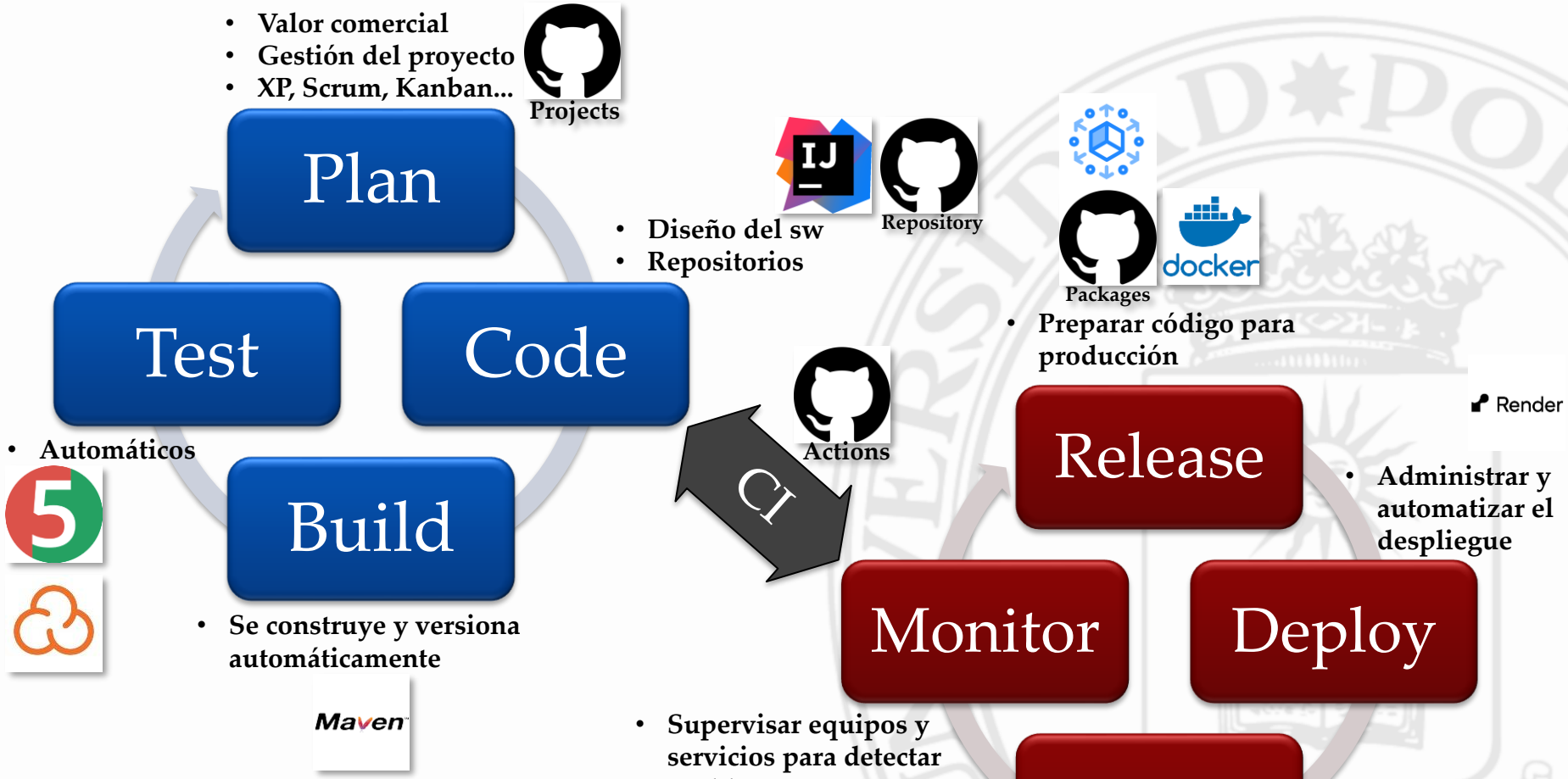
Se construye a partir del fichero Dockerfile

Nos indica que hasta que no se inicie el contenedor de BD, este no puede arrancar

Puentea el puerto interno 8081 al puerto de expuesto del contenedor 8081

Se configura Spring

BETCA
DevOps.



BETCA

Entrega Continua *(Continuous Delivery)*

Práctica del desarrollo del software para liberar código en ciclos cortos

El objetivo es que las mejoras del desarrollo sean visibles para el cliente lo antes posible.

Despliegue Continuo

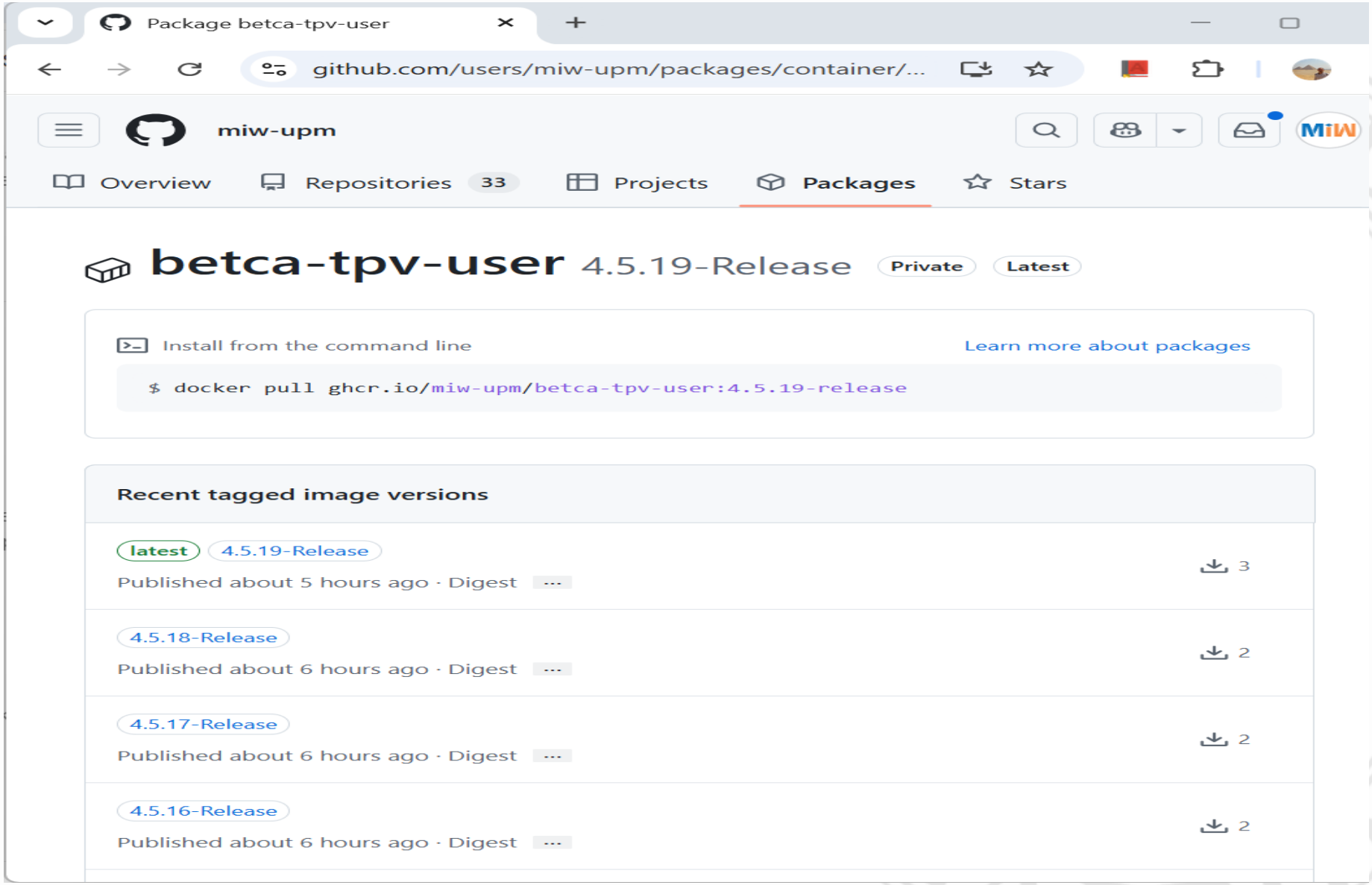
(Continuous Deployment) resulta cuando las entregas se despliegan automáticamente

GitHub. *Packages Repository*

GitHub Packages Repository

- **GitHub Packages** es un servicio de alojamiento y gestión de *paquetes de software* que permite a los desarrolladores almacenar y compartir paquetes de manera privada o pública.
- Los paquetes se integran perfectamente con *APIs de GitHub, GitHub Actions* y *webhooks* (notificaciones automáticas entre sistemas por HTTP) para crear un flujo de trabajo ***DevOps*** completo.
- ***GitHub Packages*** soporta varios registries de paquetes comunes, como *npm, RubyGems, Apache Maven, Gradle,* ***Docker*** *y NuGet (.NET).*

BETCA
Packages Repository



BETCA

Despliegue Continuo

```yaml
name: CD
on:
  push:
    branches:
      - master
jobs:
  cd:
    name: Build & Push GitHub Package & Deploy on Render Docker Image
    runs-on: ubuntu-22.04
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0
      - uses: actions/setup-java@v4
        with:
          distribution: zulu
          java-version: 21
      - name: Get version from pom.xml
        id: get-version
        run: |
          echo "REPOSITORY=${{ github.repository }}" >> $GITHUB_ENV
          echo "VERSION=$(mvn help:evaluate -Dexpression=project.version -q -DforceStdout)" >> $GITHUB_ENV
          echo "PACKAGE_PATH=ghcr.io/${{ github.repository }}" >> $GITHUB_ENV
      - name: Show NAMES
        run: |
          echo $REPOSITORY
          echo $VERSION
          echo $PACKAGE_PATH
```

Fichero ***yaml*** para el ***Despliegue Continuo*** disparado por la rama ***master***

Se generan tres variables de entorno: REPOSITORY, VERSION & PACKAGE_PATH (*miw-upm/betca-tpv-user, 4.5.19-Release, ghcr.io/miw-upm/betca-tpv-user*)

BETCA

Despliegue Continuo

1. Se crea un token de acceso personal en el perfil de la cuanta de GitHub con los permisos: *write:packages & read:packages delete:packages & repo*.
2. Se crea una variable de entorno en el repositorio con el token: *DOCKER_REGISTRY_TOKEN*

```
- name: Login to GitHub Container Registry
  uses: docker/login-action@v3
  with:
    registry: ghcr.io
    username: ${{ github.actor }}
    password: ${{ secrets.DOCKER_REGISTRY_TOKEN }}
- name: Build and Tag and Push Docker image
  run: |
    docker build  -t $REPOSITORY:$VERSION .
    docker tag $REPOSITORY:$VERSION $PACKAGE_PATH:$VERSION
    docker tag $REPOSITORY:$VERSION $PACKAGE_PATH:latest
- name: Push Docker image to GitHub Container Registry
  run: |
     docker push $PACKAGE_PATH:$VERSION
     docker push $PACKAGE_PATH:latest
```

Se construye a partir del fichero Dockerfile que esta en la raíz del proyecto

Se crean dos etiquetas

Se sube las dos referencias, la versión creada y la etiquetada como latest

BETCA
Despliegue Continuo

Fichero ***yaml*** para el ***Despliegue Continuo*** disparado por la rama ***master***

Si la petición es incorrecta, nos devuelve un error. Pero no se controla si la app se levanta adecuadamente

Se llama a un end-point del APIRest de **Render**, para indicarle que se realice un nuevo despliegue

```
- name: Deploy on Render
  run: |
    curl --fail -X POST "https://api.render.com/deploy/${{ secrets. DEPLOY_HOOK_TOKEN}}" || { echo "Deployment failed"; exit 1; }
    echo "Deployment succeeded"
```

1. En **Render**, en la *Settings* del repositorio, se debe mirar el *Deploy Hook* y copiar el *token* de acceso.
2. Crear en el repositorio, en *Secrets and variable*, en *Actions*, un *new repository secret* con el token

Si algo sale mal en la llamada, da por fallido esta etapa del despliegue

BETCA

Render. *https://render.com/*

Soporta **Despliegue Continuo:** ***GitHub, GitLab & Bitbucket***

Se pueden crear distintos tipos de servicios: ***Web estática, Servicio Web, DataBase (PostgreSQL, MySQL, MongoDB)…***

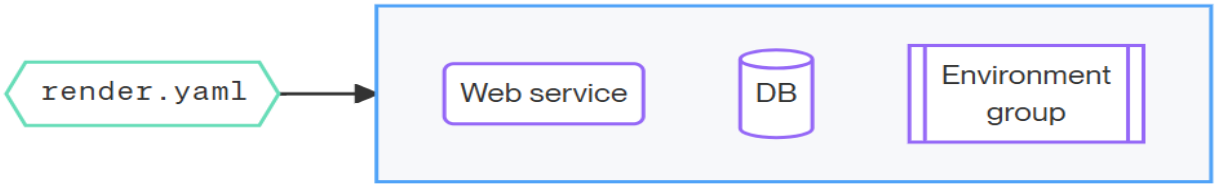
Soporta varios lenguajes: ***Node.js, Python, Ruby, Go… (~~Java~~) & Docker image***

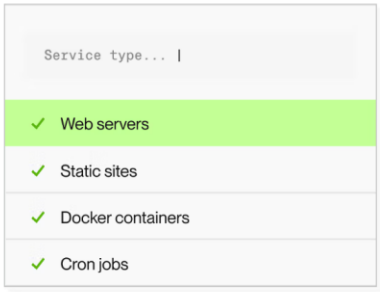
BETCA

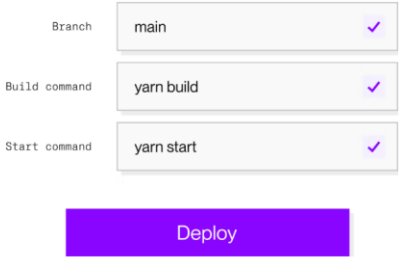
Render

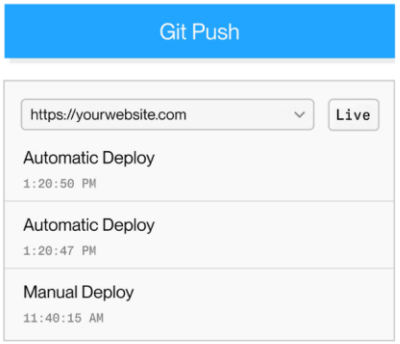
Hobby	Professional	Organization	Enterprise
For hobbyists, students, and indie hackers.	For small teams and early-stage startups.	For larger teams with complex needs.	For ultimate power and customization.
$0 USD per month	$19 USD per user/month	$29 USD per user/month	Custom pricing
+ Compute Costs	+ Compute Costs	+ Compute Costs	+ Compute Costs
Get Started	Get Started	Get Started	Contact Sales

Blueprints are Render's infrastructure-as-code (IaC) model for defining, deploying, and managing multiple resources with a single YAML file:



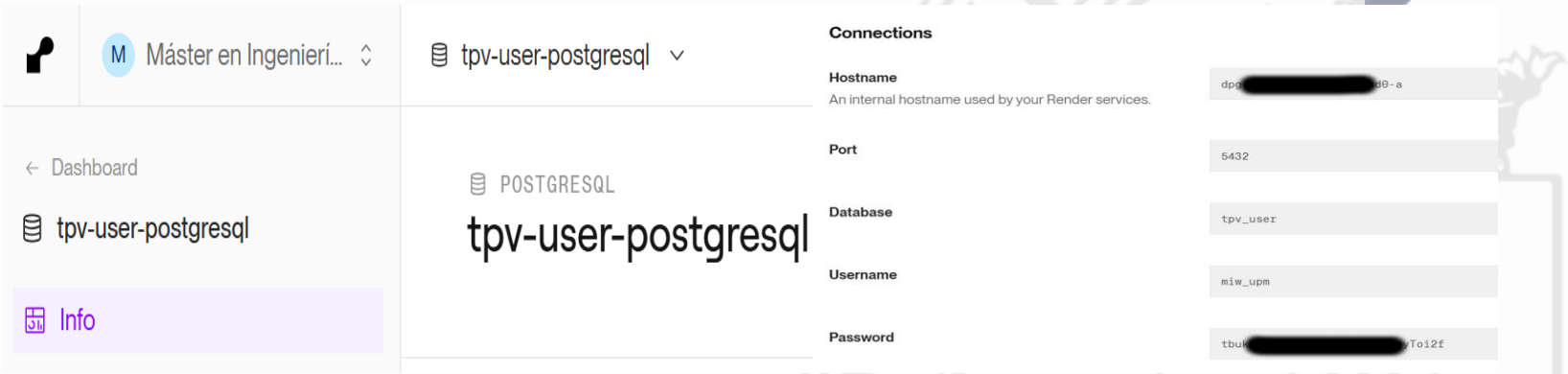






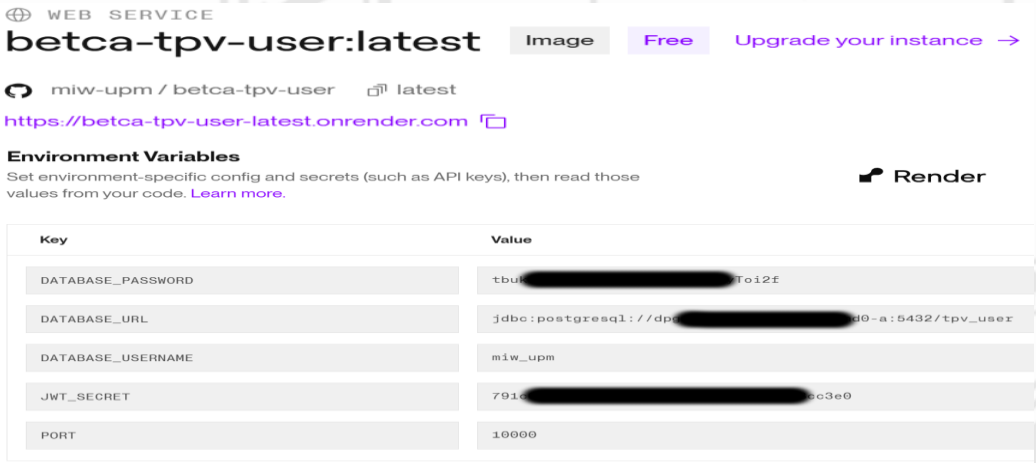
BETCA
New PostgreSQL

Name → Database → Region → Version PostgreSQL → Plan Options *Free $0 / month*



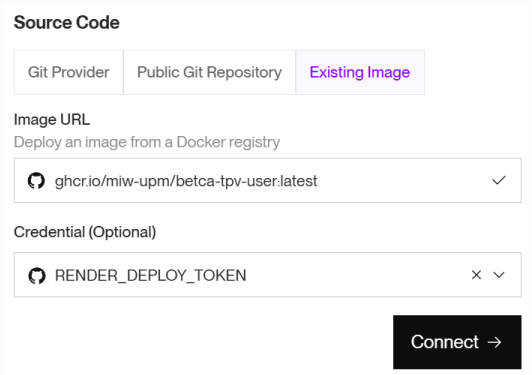
application-prod.properties

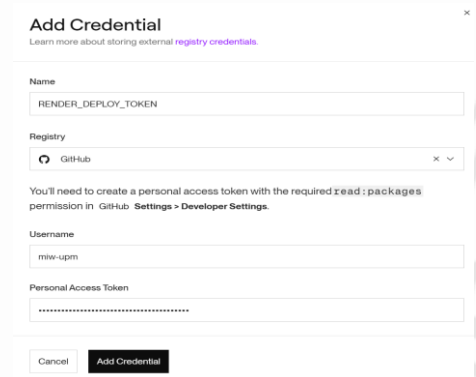
```
server.port=10000
# PostgreSQL ----------------------------------------
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=${DATABASE_URL}
spring.datasource.username=${DATABASE_USERNAME}
spring.datasource.password=${DATABASE_PASSWORD}
#JWT -----------------------------------------------------
miw.jwt.secret=secret-${JWT_SECRET}
```

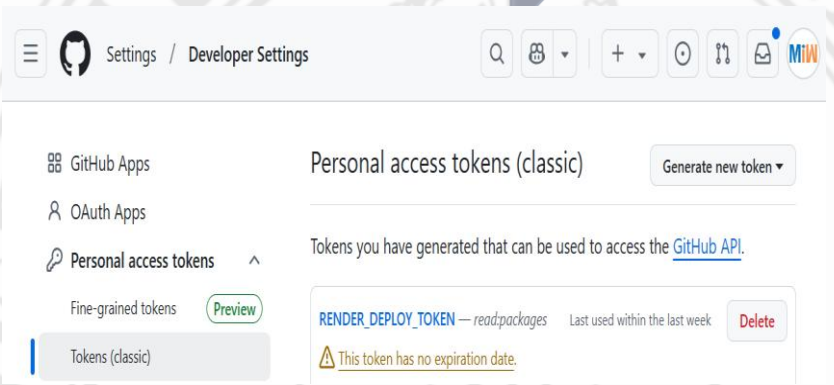


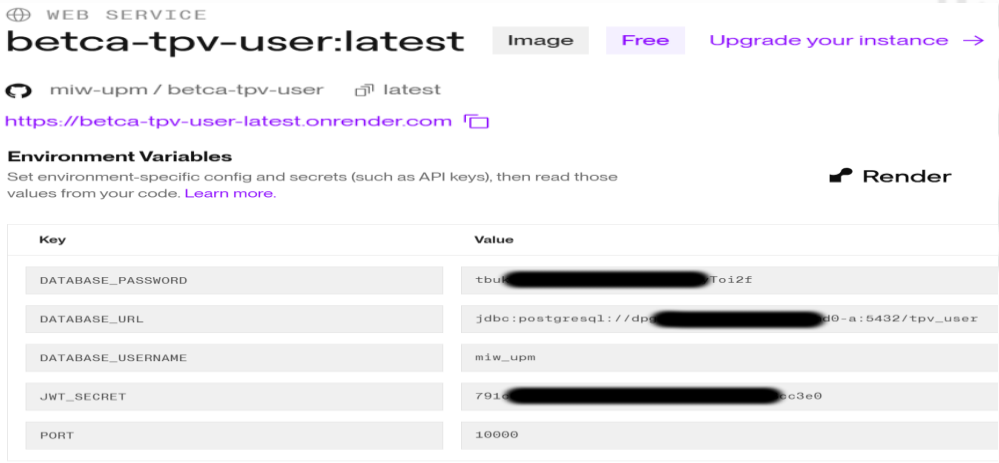
BETCA
New Web Service

Source Code → Name *(único)* → Region *Frankfurt (EU Central)* → Instance Type *Free $0 / month* → Enviroment Variables → Health Check Path */actuator/health*









BETCA
MongoDB. Atlas

https://www.mongodb.com/es/atlas

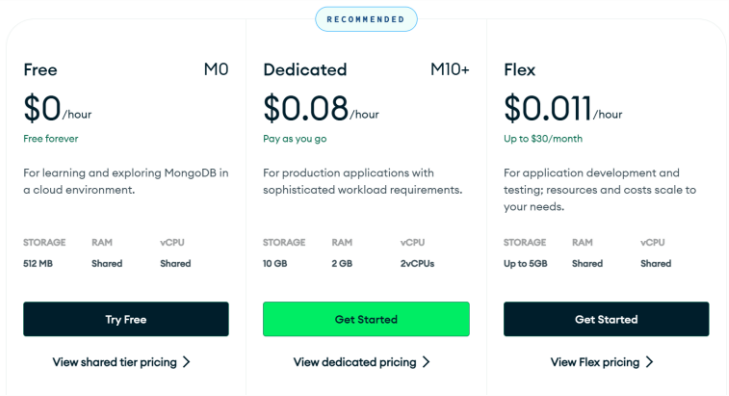
Dispone de un plan gratuito.

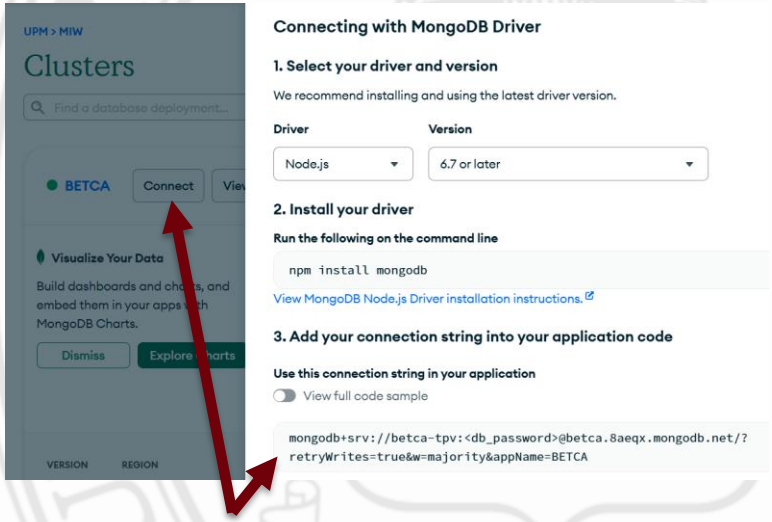
Soporta BD multi-cloud.

BETCA
MongoDB. Atlas

Create Organization *(UPM)* → Create Project *(MIW)* → Create Cluster:
- Plan *(Free)*, Name: BETCA, Provider: AWS.
- Regin: Paris, User & Password

→ User Rol: **atlasAdmin**@admin

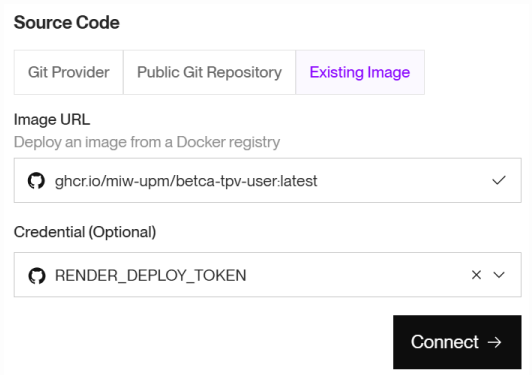


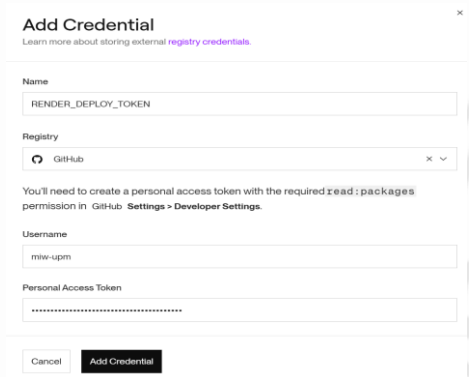


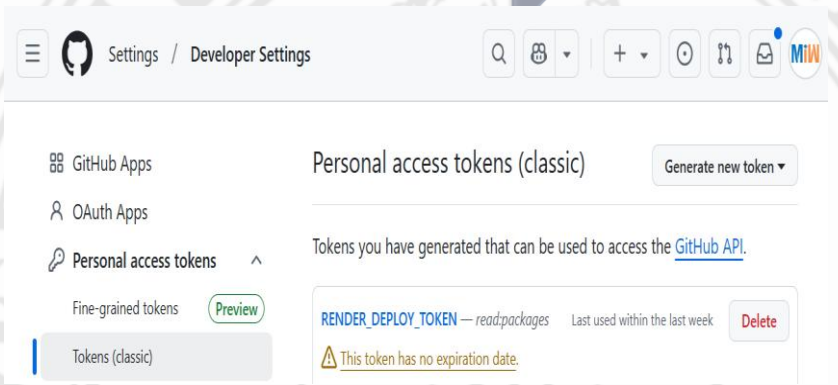
BETCA

Render. Web Service

Source Code → Name *(único)* → Region *Frankfurt (EU Central)* → Instance Type *Free $0 / month* → Enviroment Variables → Health Check Path */actuator/health*

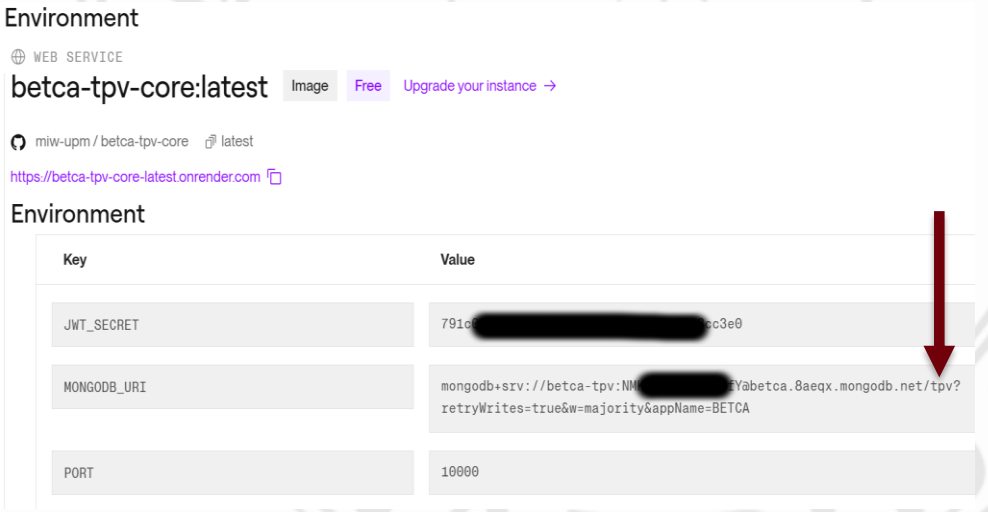






application-prod.properties

```
server.port=10000
# Database MONGODB external ------------------------------
spring.data.mongodb.uri=${MONGODB_URI}
#JWT ----------------------------------------------------
miw.jwt.secret=secret-${JWT_SECRET}
```



BETCA
Angular. CD

Despliegue en Render

A través de Imagen Docker

Desde el repositorios de GitHub

BETCA
Angular. CD

```
# Etapa de construcción
FROM node:22.13.1-alpine AS build
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm ci
COPY . .
RUN npm run build-prod
```

Parte de una imagen con Node con Alpine

Copia el *package.json*de la ruta raíz del proyecto a la carpeta de trabajo del contenedor

Se instala las dependencias

Se construye en entorno producción

```
# Etapa de runtime con un servidor ligero de ficheros (Nginx)

FROM nginx:stable-alpine
WORKDIR /usr/share/nginx/html
COPY --from=build /app/dist/betca-tpv-angular/browser/ ./
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Puerto HTTP abierto

```
# Etapa de runtime, express un servidor mas pesado
FROM node:22.13.1-alpine
WORKDIR /app
COPY --from=build /app/dist/betca-tpv-angular /app
RUN npm install -g serve
EXPOSE 10000
CMD ["serve", "-s", "-l", "10000", "/app"]
```

Establece un comando para cuando se inicialice el contenedor: *RUN serve -s -l 10000*

BETCA

Angular. *Despliegue Continuo*

```yaml
name: CD
on:
  push:
    branches:
      - master
jobs:
  cd:
    name: Build & Push GitHub Package & Deploy on Render Docker Image
    runs-on: ubuntu-22.04
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0
      - uses: actions/setup-node@v4
        with:
          node-version: 22.13.1
      - name: Get version from package.json
        id: get-version
        run: |
          echo "REPOSITORY=${{ github.repository }}" >> $GITHUB_ENV
          echo "VERSION=$(node -pe "require('./package.json').version")" >> $GITHUB_ENV
          echo "PACKAGE_PATH=ghcr.io/${{ github.repository }}" >> $GITHUB_ENV
```

BETCA

Angular. *Despliegue Continuo*

```
- name: Show NAMES
  run: |
    echo $REPOSITORY
    echo $VERSION
    echo $PACKAGE_PATH

- name: Login to GitHub Container Registry
  uses: docker/login-action@v3
  with:
    registry: ghcr.io
    username: ${{ github.actor }}
    password: ${{ secrets.DOCKER_REGISTRY_TOKEN }}

- name: Build and Tag and Push Docker image
  run: |
    docker build  -t $REPOSITORY:$VERSION .
    docker tag $REPOSITORY:$VERSION $PACKAGE_PATH:$VERSION
    docker tag $REPOSITORY:$VERSION $PACKAGE_PATH:latest

- name: Push Docker image to GitHub Container Registry
  run: |
     docker push $PACKAGE_PATH:$VERSION
     docker push $PACKAGE_PATH:latest

- name: Deploy on Render
  run: |
     curl --fail -X POST "https://api.render.com/deploy/${{ secrets.DEPLOY_HOOK_TOKEN }}" || { echo "Deployment failed"; exit 1; }
     echo "Deployment succeeded"
```

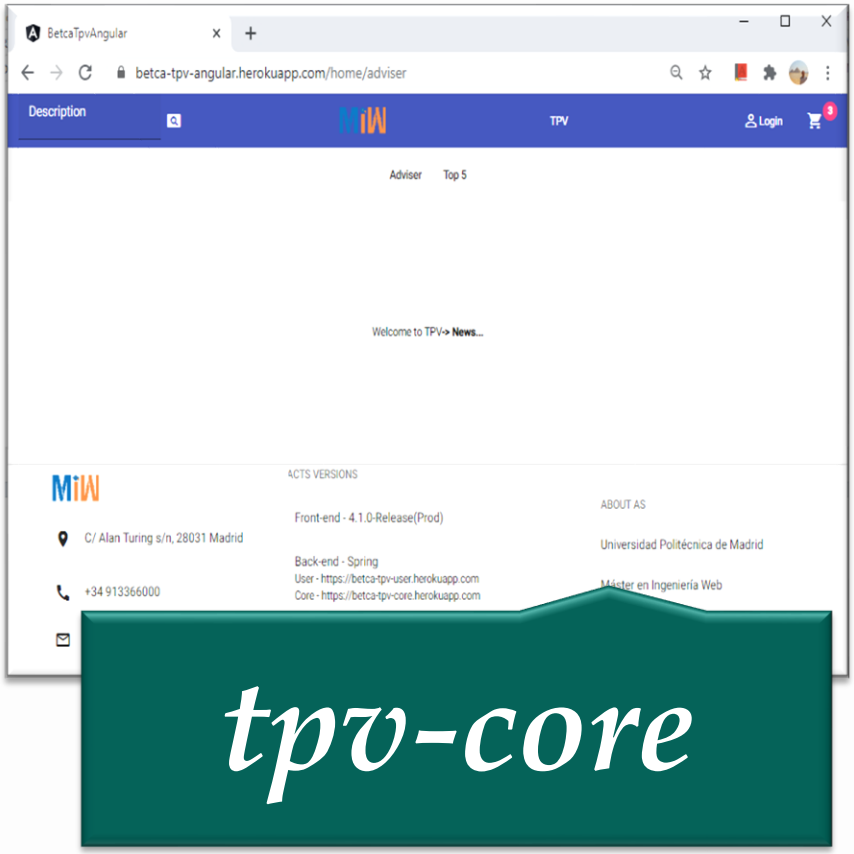
Lo mismo que para *.user, *.core…

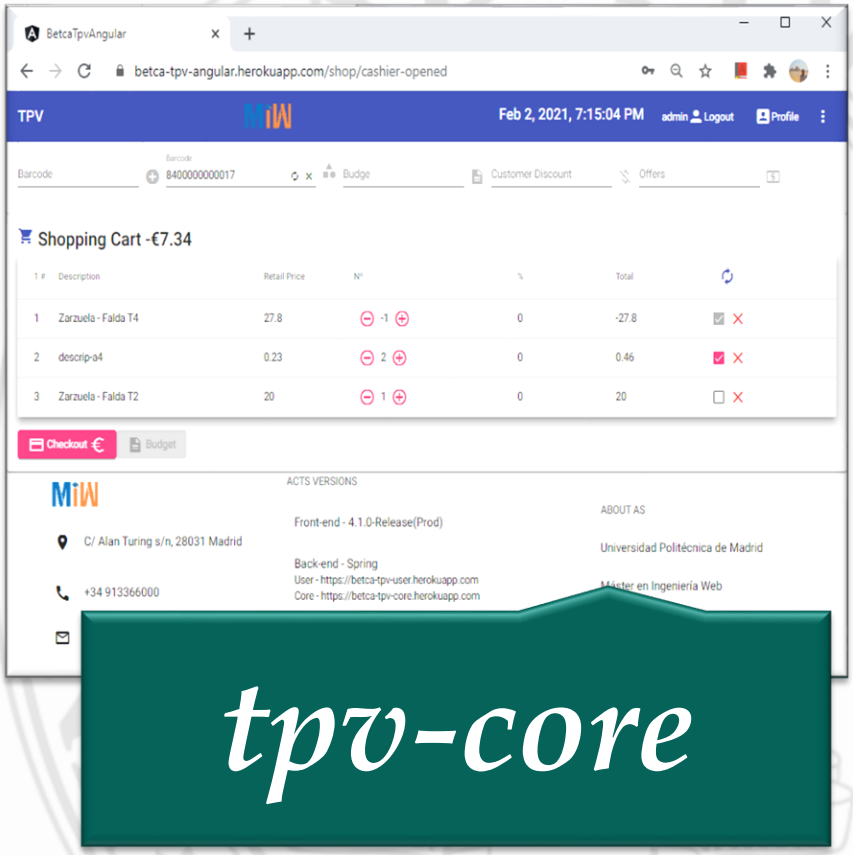
TPV

https://github.com/miw-upm/betca-tpv

- **Asignación de prácticas:** ***Wiki - Labels***
- **Gestión del proyecto:** ***TPV***
  - ***Commits: "miw-upm/betca-tpv#xxx"***
- **Enunciado:** ***README***
  - ***Story Requirements: UI***
- **Entrega**

TPV


BetcaTpvAngular
betca-tpv-angular.herokuapp.com/home/adviser
Description
TPV
Login
Adviser
Top 5
Welcome to TPV-> News...
C/ Alan Turing s/n, 28031 Madrid
+34 913366000
ACTS VERSIONS
Front-end - 4.1.0-Release(Prod)
Back-end - Spring
User - https://betca-tpv-user.herokuapp.com
Core - https://betca-tpv-core.herokuapp.com
ABOUT AS
Universidad Politécnica de Madrid
tpv-core


BetcaTpvAngular
betca-tpv-angular.herokuapp.com/shop/cashier-opened
TPV
Feb 2, 2021, 7:15:04 PM
admin
Logout
Profile
Barcode
8400000000017
Budge
Customer Discount
Offers
Shopping Cart -€7.34
Description
Retail Price
Total
1 Zarzuela - Falda T4 27.8 -1 0 -27.8
2 descrip-a4 0.23 2 0 0.46
3 Zarzuela - Falda T2 20 1 0 20
Checkout €
Budget
C/ Alan Turing s/n, 28031 Madrid
+34 913366000
ACTS VERSIONS
Front-end - 4.1.0-Release(Prod)
Back-end - Spring
User - https://betca-tpv-user.herokuapp.com
Core - https://betca-tpv-core.herokuapp.com
ABOUT AS
Universidad Politécnica de Madrid
tpv-core

Angular
UI

Angular Material (*Google*)

- https://material.angular.io/
- Github: ★24.5K

PrimeNG

- https://www.primefaces.org/primeng/
- Github: ★11.1K

NG Bootstrap

- https://ng-bootstrap.github.io
- Github: ★8.2K

Onsen UI

- https://onsen.io/
- Github: ★8.8K

NG Zorro

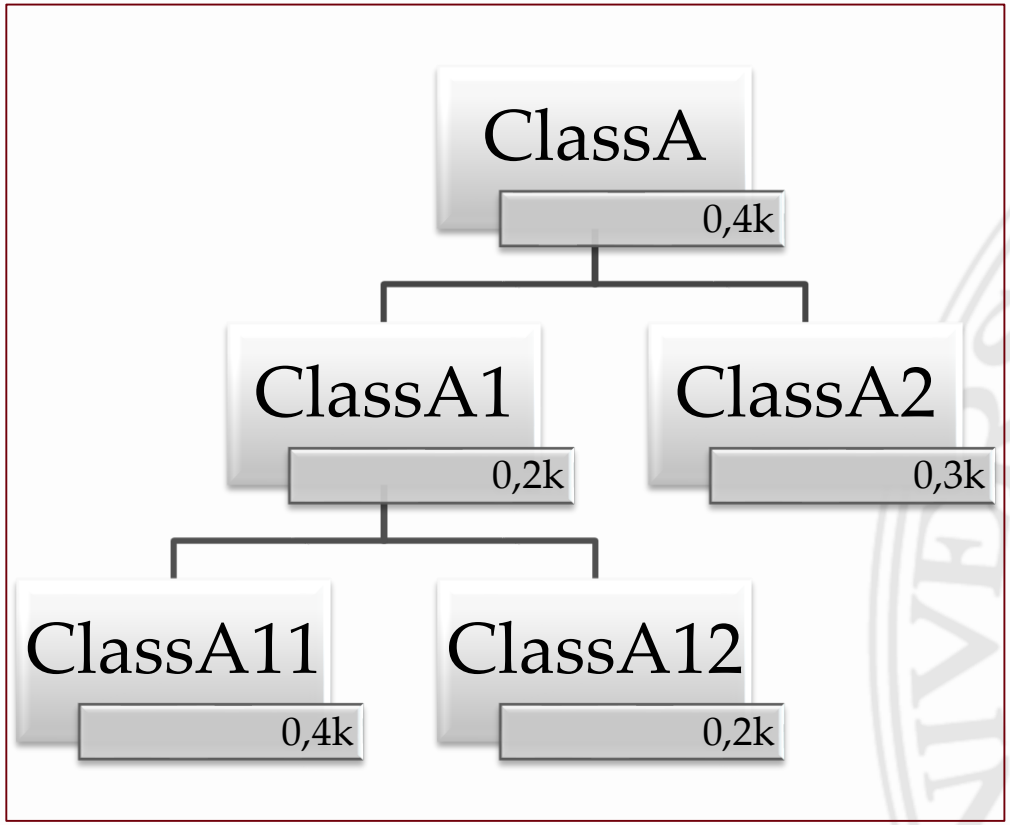
- https://ng.ant.design
- Github: ★8.9K

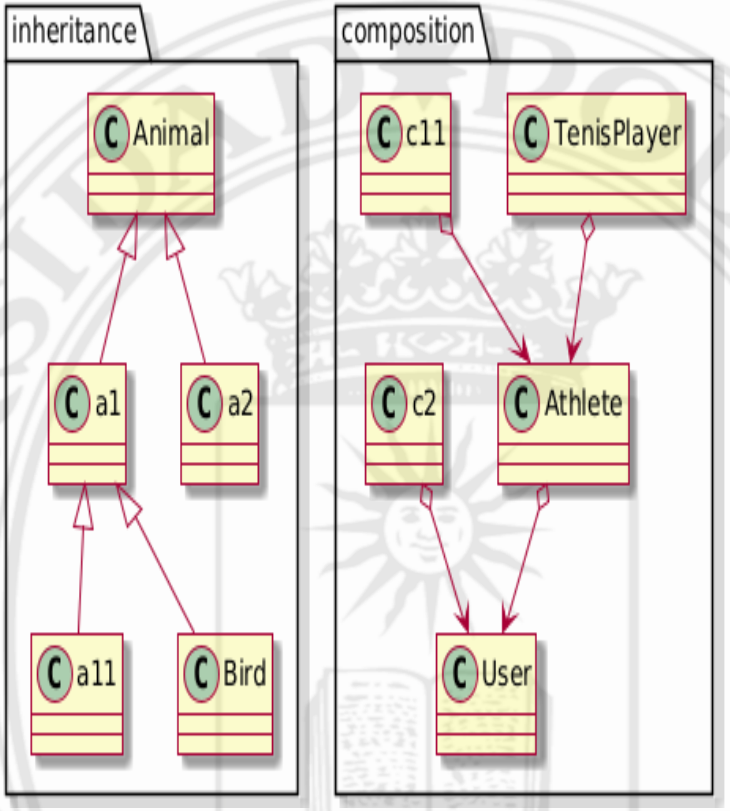
Clarity

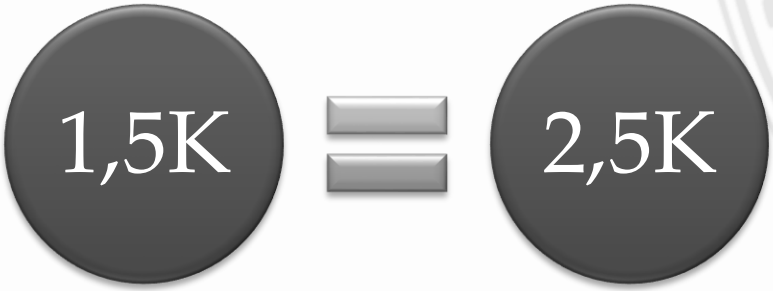
- https://clarity.design/
- Github: ★6.4K

Angular

Composición vs Herencia

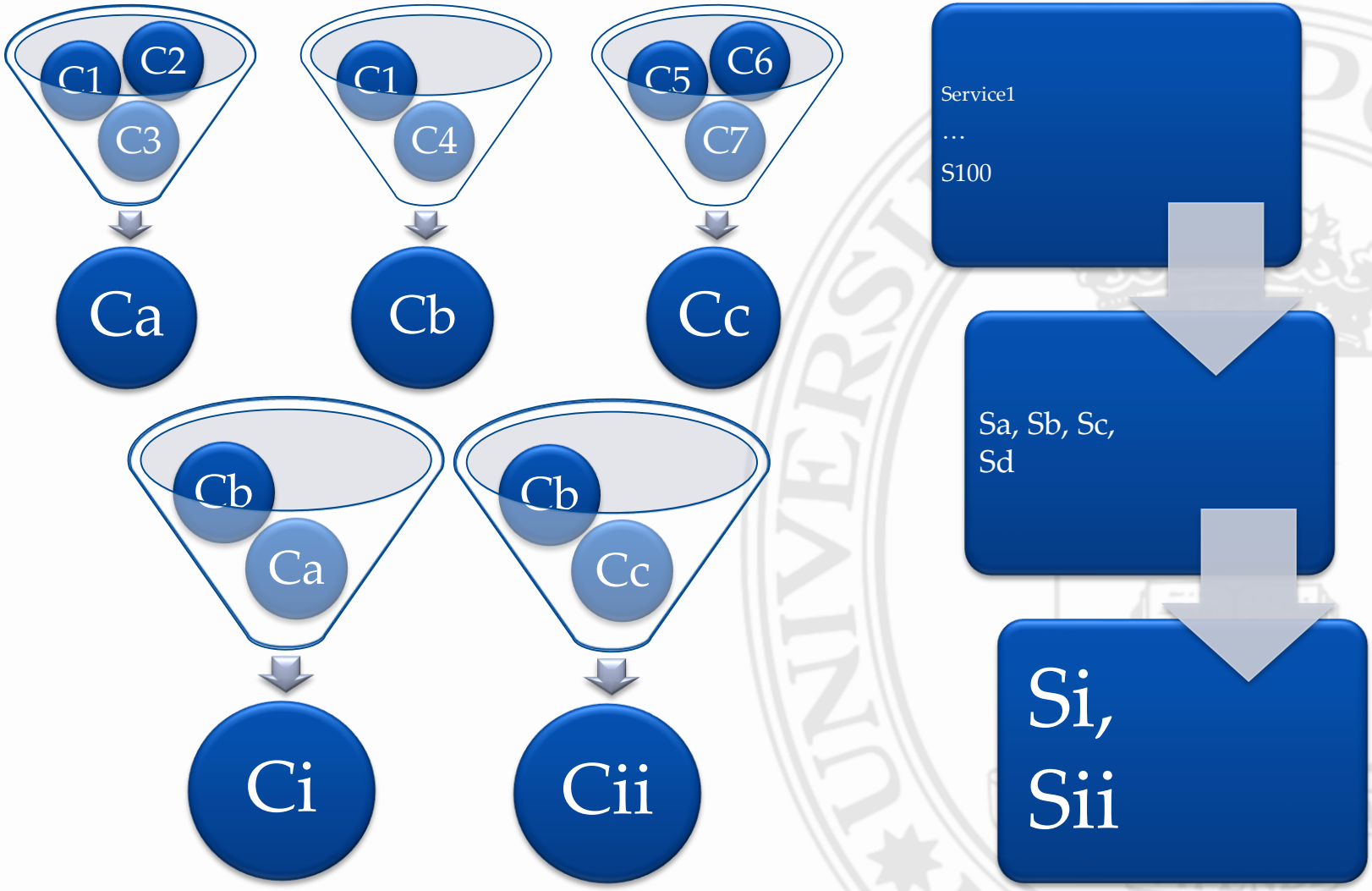




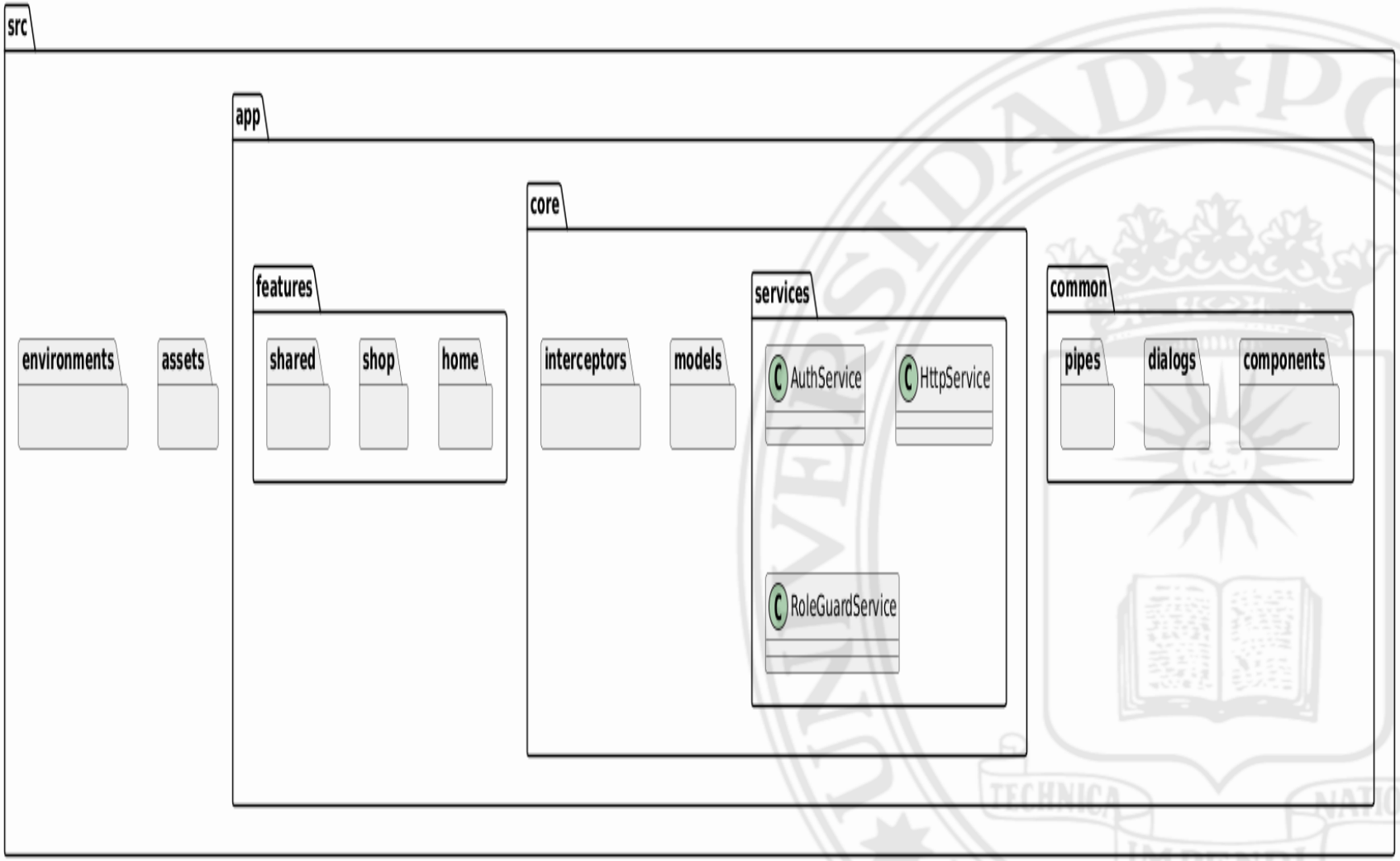


Angular

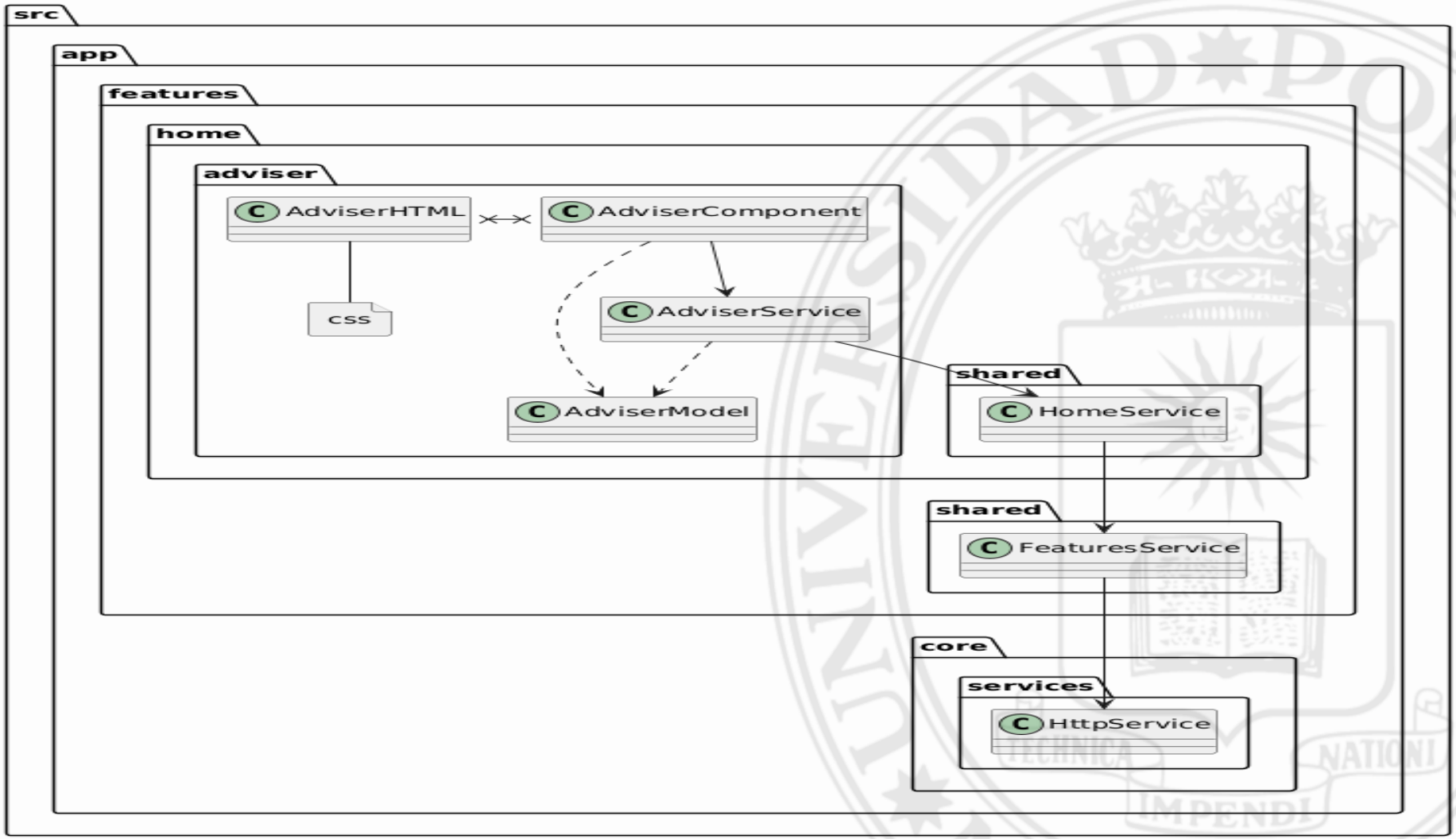
Componentes (Composición) & Servicios (Asociación)



Angular
Arquitectura



Angular
Arquitectura



Angular
View

Template: HTML*

```
@if (warning()) {
   <mat-icon color="warn">report_problem</mat-icon> Uncommit articles & phone not registered
}
```

```
<mat-checkbox [(ngModel)]="requestedInvoice" [disabled]="invalidInvoice()">
  <span> Create Invoice</span>
</mat-checkbox>
```

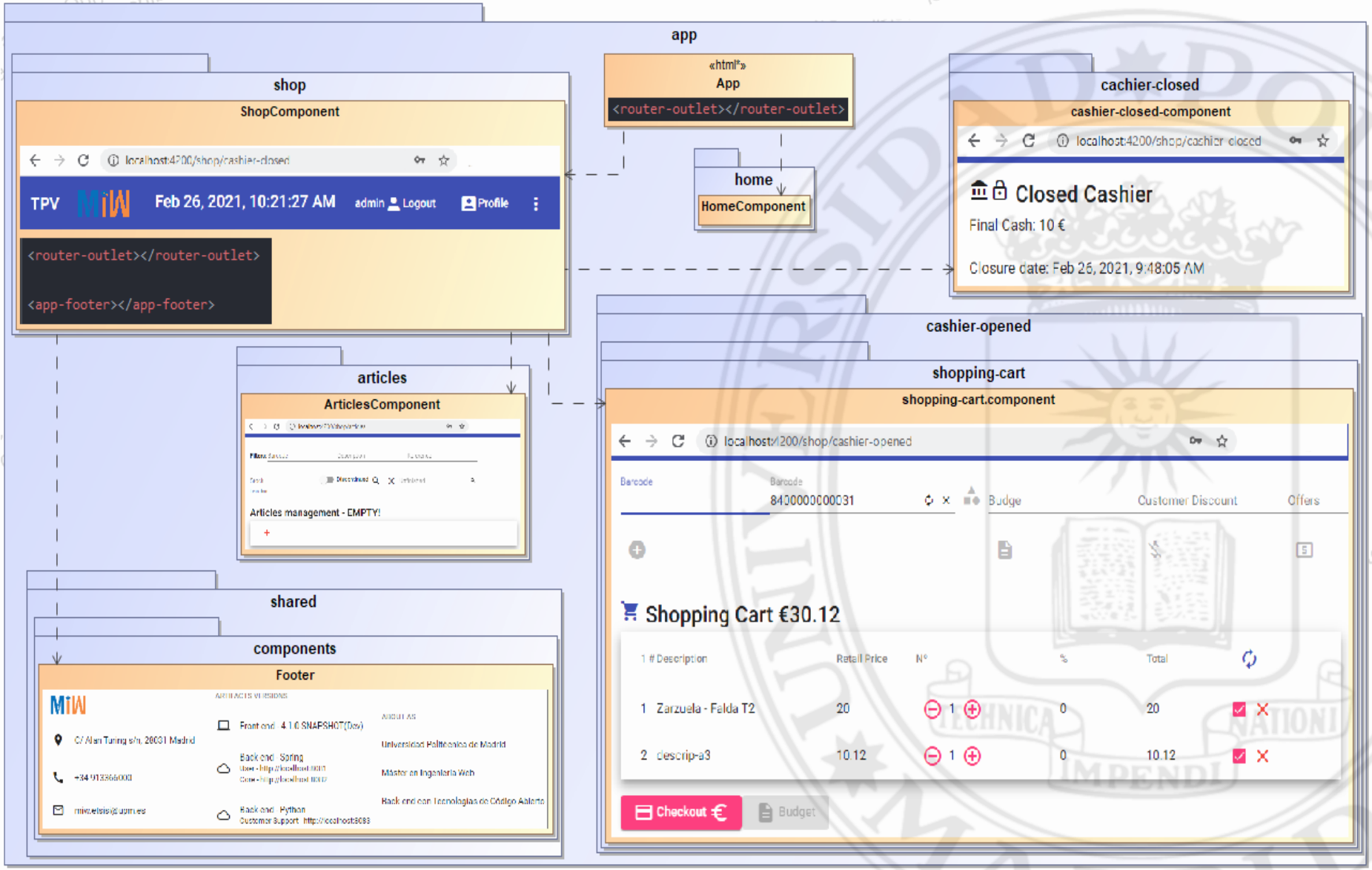
Component

```
warning(): boolean {
  return !this.managedMobile() && this.unCommitted();
}
```

```
invalidInvoice(): boolean {
  // TODO pendiente de calcular. Hace falta tener al usuario totalmente completado
  return true;
}
```

```
articles = of([]);
search(): void {    this.articles = this.articleService.search(this.articleSearch);  }
```

Angular
Arquitectura



Angular
Arquitectura

app.routes.ts

- Riesgo alto de conflicto

```
export const routes: Routes = [
  {path: '', pathMatch: 'full', redirectTo: 'home/adviser'},
  {
    path: 'home', component: HomeComponent,
    children: [
      {path: 'adviser', component: AdviserComponent}, // PUBLIC
      {
        path: 'complaints',
        component: ComplaintsComponent,
        canActivate: [RoleGuardService],
        data: {roles: [Role.CUSTOMER]}
      }
    ]
  },
  {
    path: 'shop', component: ShopComponent,
    canActivate: [RoleGuardService],
    data: {roles: [Role.ADMIN, Role.MANAGER, Role.OPERATOR]},
    children: [ // or path: 'shop/articles'
      {path: 'articles', component: ArticlesComponent},
      {path: 'cashier-closed', component: CashierClosedComponent},
      {path: 'cashier-opened', component: CashierOpenedComponent},
      {path: 'providers', component: ProvidersComponent},
      {path: 'tickets', component: TicketsComponent},
    ]
  }
];
```

Angular {→}

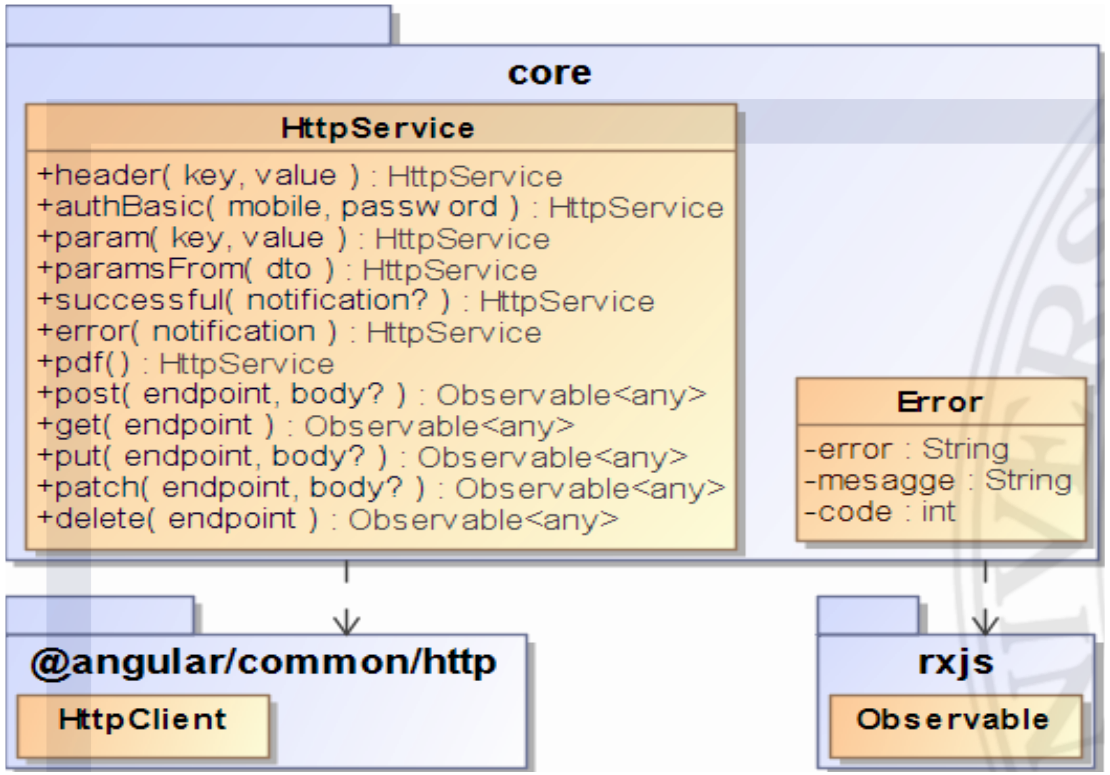
https://github.com/miw-upm/betca-tpv-angular

- ***https://github.com/miw-upm/betca-tpv***
- **Clonar el proyecto betca-tpv-user**
  - Crear imagen y contenedor con ejecución. Manejarlo con *Docker Desktop*
- **Clonar el proyecto betca-tpv-core**
  - Crear imagen y contenedor. Manejarlo con *Docker Desktop*
- **Clonar el proyecto *betca-tpv-angular*, instalar y abrir con *WebStorm***
  - *> git clone https://github.com/miw-upm/betca-tpv-angular*
  - *> cd betca-tpv-angular*
  - *> betca-tpv-angular> npm install*
- **Para ejecutar**
  - ***> ng serve***

Carpetas

- Comprender la arquitectura

Angular
HttpClient



httpService.paramsFrom(articleSearch).get(endpoint)

httpService.pdf().get(endpoint)

HttpService

- Patrón **Facade** & **Builder**. Simplificar llamadas al API Rest y ayudar a crear la llamada

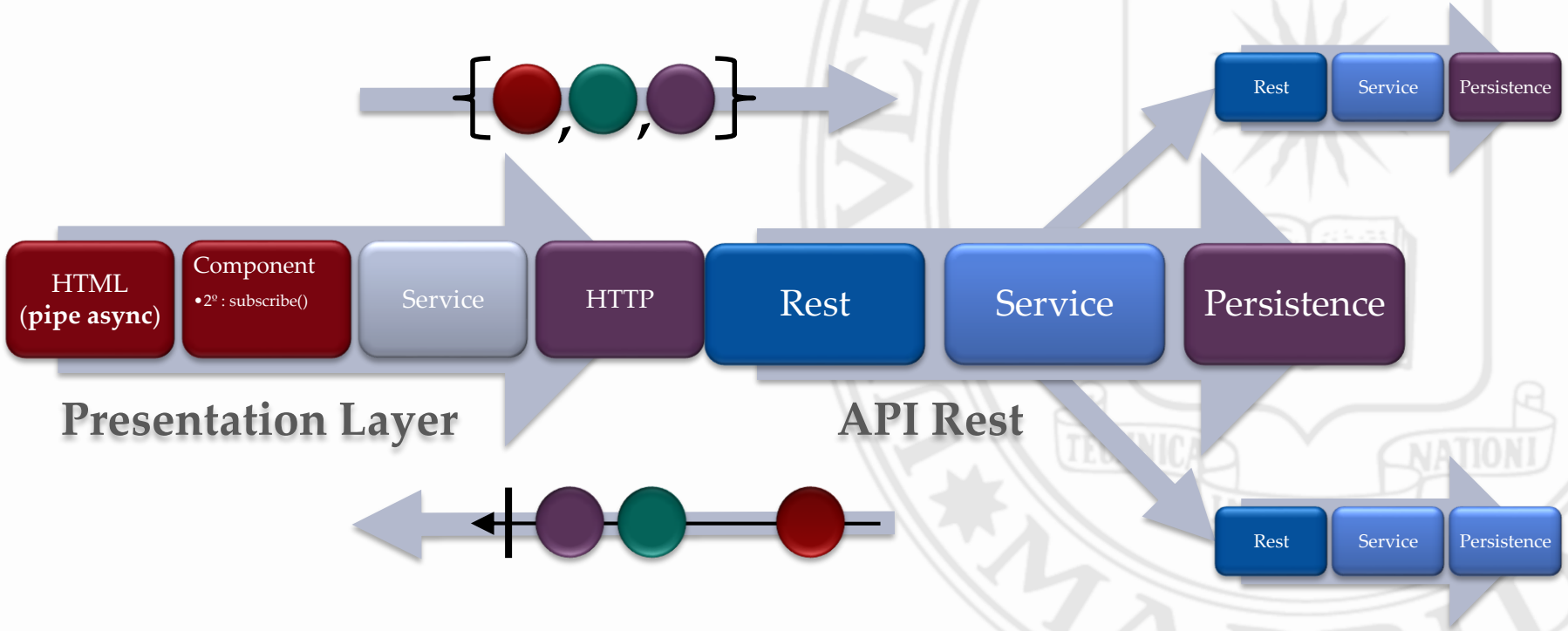
- Tratamiento de Seguridad: Basic Auth

- Tratamiento automático de errores con extracción del mensaje de error y presentación mediante **Snackbar**

- Tratamiento de **pdf** con presentación automática en ventana nueva

Angular
Flujo

Métodos que devuelven un tipo *Publisher* no *deberían* subscribirse porque ello *podría romper* la cadena del publicador



Angular
Flujo

ReadDetailDialogHTML

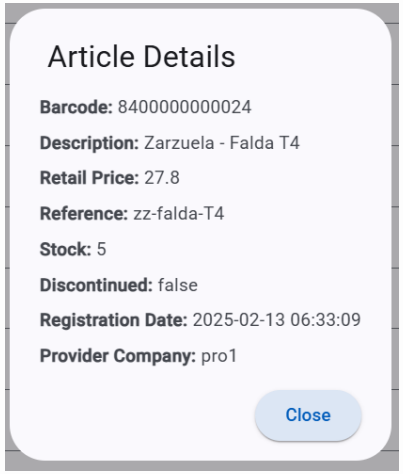
```
<mat-dialog-content>  <ng-container *ngIf="(object | async) as data">
    @for (def of labels(data); track def) {
      <mat-label><strong>{{ def | uppercaseWords }}: </strong>{{ data[def] }}</mat-label>
    }
  </ng-container> </mat-dialog-content>
```

ReadDetailDialogComponent

```
export class ReadDetailDialogComponent {
  title: string;  object: Observable<any>;
  constructor(@Inject(MAT_DIALOG_DATA) data: any) {
    this.title = data.title;  this.object = data.object;
  }
  labels(object): string[] {  return Object.getOwnPropertyNames(object);  }
}
```

ArticleComponent

```
read(article: Article): void {
  this.dialog.open(ReadDetailDialogComponent, {
    data: {  title: 'Article Details',  object: this.articleService.read(article.barcode)
    }    });  }
```



ArticleService

```
read(barcode: string): Observable<Article> {
  return this.httpService.get(EndPoints.ARTICLES + '/' + barcode);
}
```

HttpService

```
get(endpoint: string): Observable<any> {
  return this.http
    .get(endpoint, this.createOptions())
    .pipe(
      map(response => this.extractData(response)),
      catchError(error => this.handleError(error))
    );
}
```

Angular
Seguridad

HTTPS

Authorization

- Basic user:password$_{code64}$
- Bearer jwt
- Basic user:api-key$_{code64}$
- URL://*/api-key

Angular
Seguridad: JSON Web Token

JWT

- https://jwt.io
- https://auth0.com

Header$_{code64}$

- Se define el tipo.
  - HS256

Payload$_{code64}$

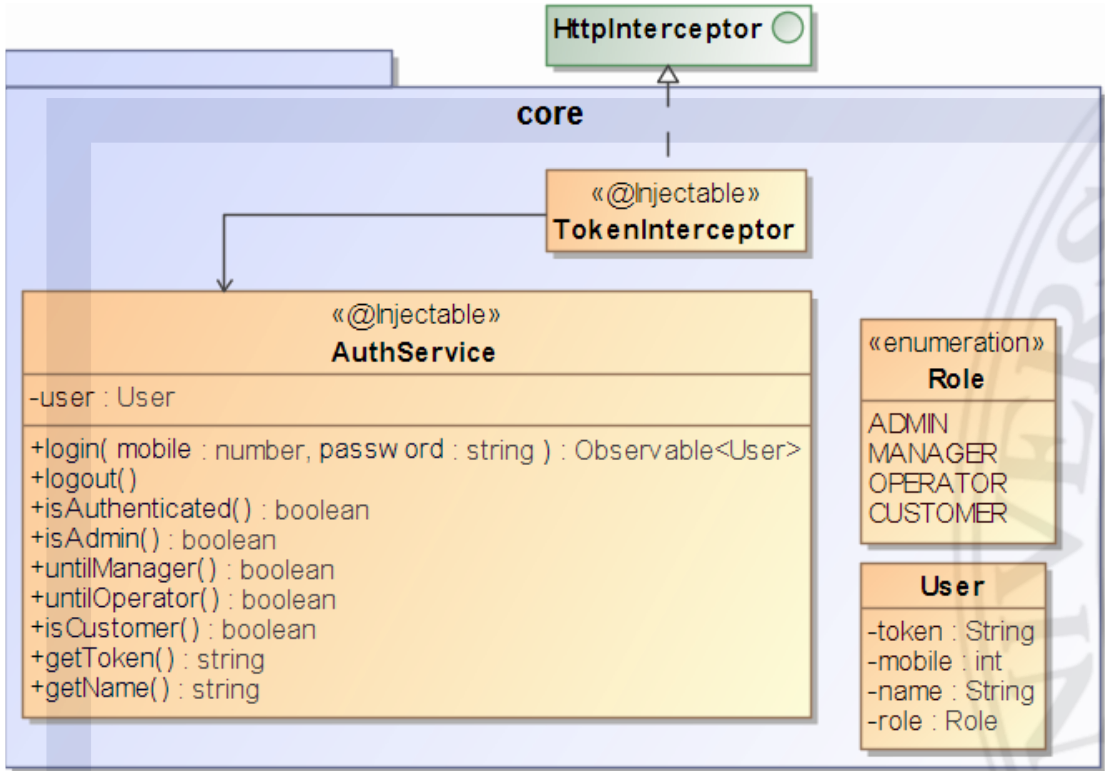
- Datos o carga del token.
  - Proveedor
  - Usuario
  - Expiración
  - Validez
  - Privilegios: rol, nombre…

Signature$_{code64}$

- Se firma con *clave secreta* la carga del *token* para saber si ha sido alterado.

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
eyJuYmYiOjE2MDc0NDk2NzYsInJvbGUiOiJBRE1JTiIsImlzcyI6Im1pdy5qd3QuaXNzdW
VyIiwibmFtZSI6ImFkbWluIiwiZXhwIjoxNjA3NDUzMjc2LCJpYXQiOjE2MDc0NDk2NzYsIn
VzZXIiOiI2In0.
V9QW9gMeYH2cKFxErfhh51FROkUZzLKqlvYhBm2YNLc

Angular
Seguridad



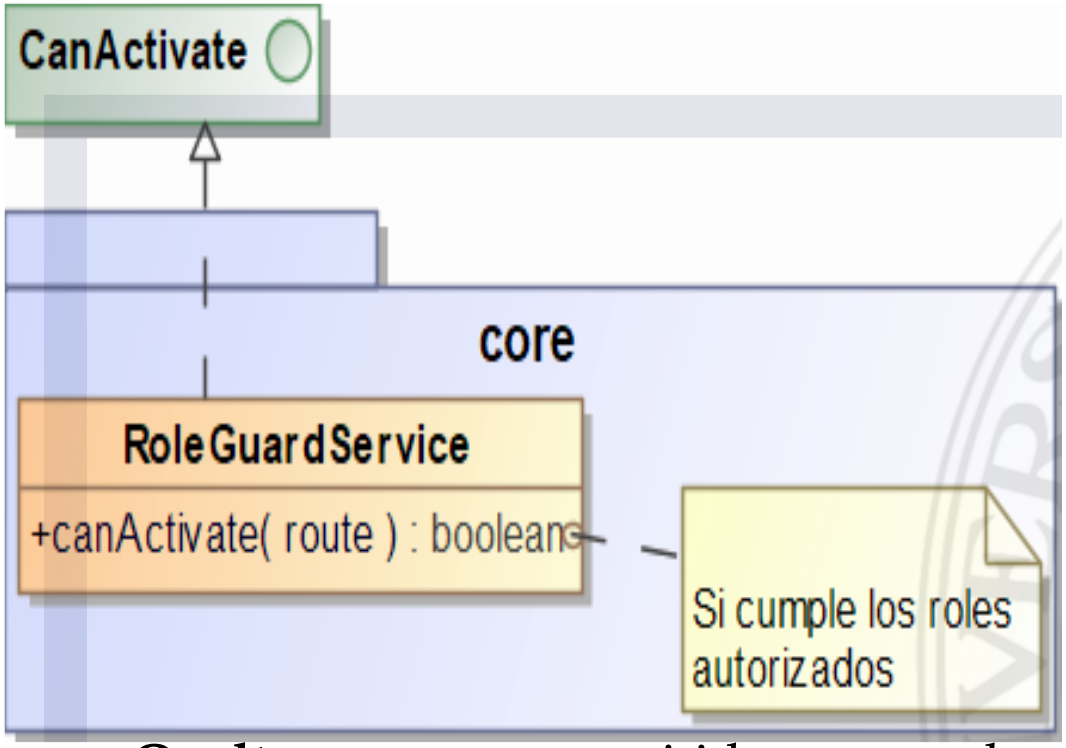
- **login**: crea *user*: ***POST betca-tpv-user//users/token***
- **logout**: elimina el *user*

Auth Basic: Para obtener el token

Bearer JWT: para todos los accesos mediante token

TokenInterceptor. Si existe un **token**, lo incluye automáticamente en la cabecera de todas las peticiones

Angular
Seguridad de enrutamiento



Ocultar rutas no permitidas y en cada ruta, indicar **roles permitidos**

RoleGuardService

Comprueba que el rol actual, el rol del token almacenado, cumple con los roles necesarios para acceder al la ruta

```
{path: 'adviser', component: AdviserComponent}, // public
{path: 'complaints', component: ComplaintsComponent, canActivate: [RoleGuardService], data: {roles: [Role.CUSTOMER]}}
```

Angular
Servicios Mock

Servicios mock

```
export class ArticleService {
  create(article: Article): Observable<Article> {
    return of(article);
  }
  read(barcode: string): Observable<Article> {
    return of({barcode: barcode, description: '...', retailPrice: 10, providerCompany: 'pro1'})
  }
  search(articleSearch: ArticleSearch): Observable<Article[]> {
    return of([
      {barcode: 123, description: '...', retailPrice: 10, providerCompany: 'pro1'},
      {barcode: 345, description: '...', retailPrice: 10, providerCompany: 'pro2'},
      {barcode: 678, description: '...', retailPrice: 10, providerCompany: 'pro1'}
    ])
  }
```

Angular
Pipe *async*

Uso del Pipe `async`

```
<p><button mat-raised-button (click)=requestSync() >Sync </button> {{sync}}</p>
<p><button mat-raised-button (click)=requestAsync()>Async</button> {{asyn | async}}</p>
```

```
export class InputOverviewExample {
  sync: string;
  asyn: Observable<string>;
  requestSync(): void {
    this.serviceMock()
    .subscribe(item => this.sync = item);
  }
  requestAsync(): void {
     this.asyn = this.serviceMock();
  }
  serviceMock(): Observable<string> {
    return of('Result');
  }
}
```

Angular
Componentes

my-comp.component.html

```
<p>Hello {{user}}! <input type="button" value="Ok" (click)="onClick()"></p>
```

```
import { Component, EventEmitter, Input, Output } from '@angular/core';
@Component({ selector: 'app-my-comp', templateUrl: './my-comp.component.html' })
export class MyCompComponent {
   @Input() user = 'By default';
   @Output() pressed = new EventEmitter<string>();
   onClick(){
      this.pressed.emit(this.user)
   }
}
```

Utilización:

```
<app-my-comp></app-my-comp>
<app-my-comp user="Value"></app-my-comp>
<app-my-comp [user]=user></app-my-comp>
<app-my-comp user="With event" (pressed)="onPressed($event)"></app-my-comp>
```

```
export class AppComponent  {
  user = "Variable";
  onPressed(value:string){
    alert(value);
  }
}
```

Angular
Componentes

```
<app-crud (create)="create()" (read)="read($event)" (update)="update($event)" [data]="providers" [title]="title"></app-crud>
```

```
<app-crud (create)="create()" (read)="read($event)" (update)="update($event)"
        [data]="providers" [deleteAction]="true" [printAction]="true" [runAction]="true" [title]="title"></app-crud>
```

```
create(): void {
  this.dialog
    .open(ProviderCreationUpdatingDialogComponent)
    .afterClosed()
    .subscribe(() => this.search());
}

read(provider: Provider): void {
  this.dialog.open(ReadDetailDialogComponent, {
    data: {
      title: 'Provider Details',
      object: this.providerService.read(provider.company)
    }
  });
}

update(provider: Provider): void {
  this.providerService
    .read(provider.company)
    .subscribe(fullProvider => this.dialog.open(ProviderCreationUpdatingDialogComponent, {data: fullProvider})
      .afterClosed()
      .subscribe(() => this.search())
    );
}
```

Async

Sync

Angular {→}

https://github.com/miw-upm/betca-tpv

- Perfiles: environments
- Componentes
  - core: *CRUD, Footer, Search, Date*
  - shop/shared: *SearchByBarcode (Search), SearchByCompany*
- *package.json*
- Rutas: *tsconfig.json*

Angular

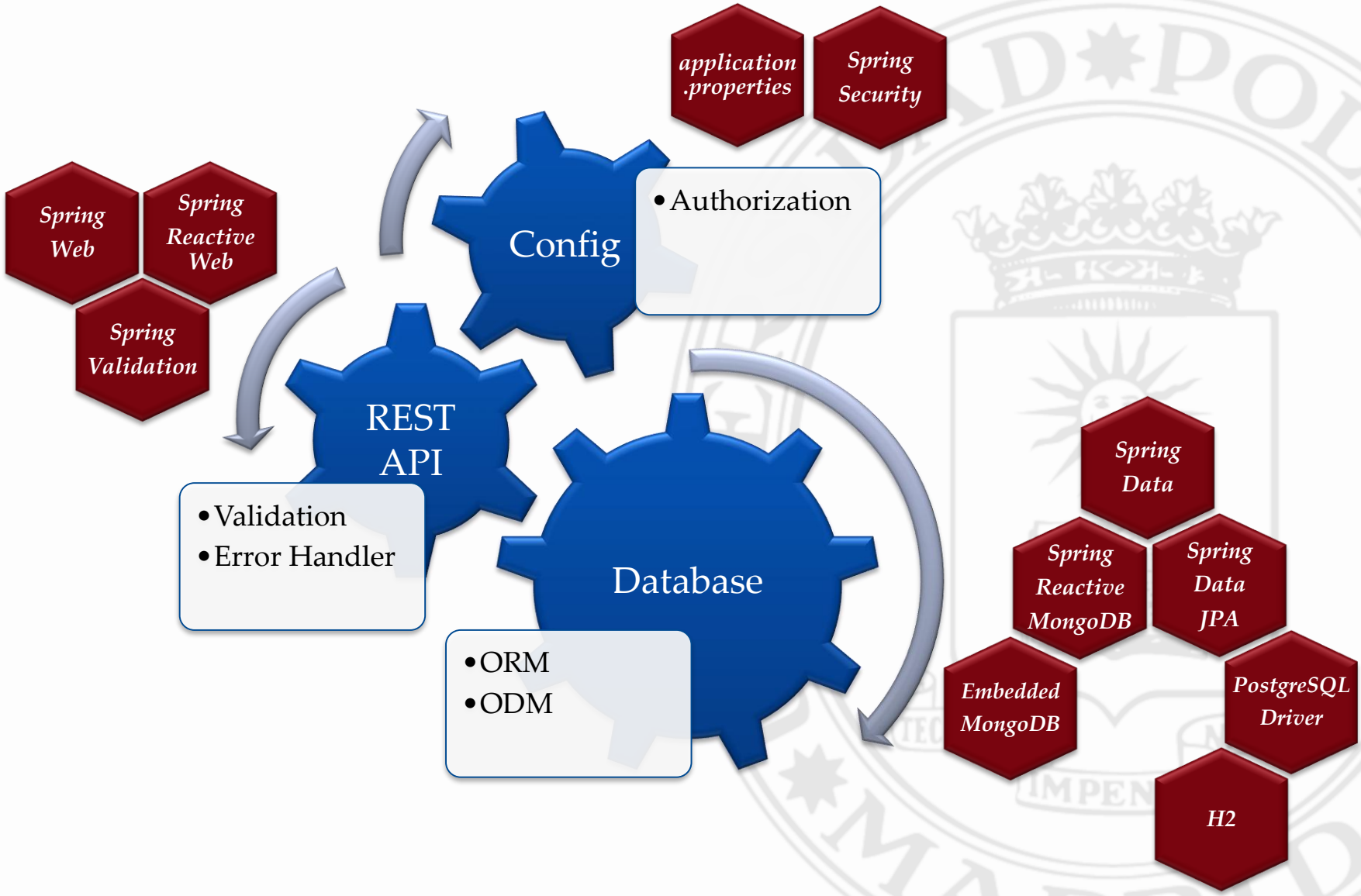
- Comprender el código

Spring

https://spring.io/

Spring es un framework ligero de código abierto, creado por Rod Jhonson, para facilitar el desarrollo de Aplicaciones Empresariales modernas (***Java*** o ***kotlin***)

Spring
Web


application .properties
Spring Security
Spring Web
Spring Reactive Web
Spring Validation
Config
• Authorization
REST API
• Validation
• Error Handler
Database
• ORM
• ODM
Spring Data
Spring Reactive MongoDB
Spring Data JPA
Embedded MongoDB
PostgreSQL Driver
H2

Spring

https://spring.io/

Spring Boot	Spring Framework	Spring Data	Spring Security	Spring Cloud
• Construye y arranca rápidamente la aplicación con una configuración inicial mínima de Spring.	• Proporciona un modelo completo de programación y configuración para aplicaciones empresariales modernas.	• Proporciona acceso a diferentes tipos de bases de datos.	• Protege la aplicación con diferentes sistemas de autenticación y autorización.	• Proporciona herramientas que implementas los patrones comunes de sistemas distribuidos.

Spring Cloud Data Flow | Spring for GraphQL | Spring AI | Spring REST Docs | Spring HATEOAS | Spring CLI | …

Spring Framework

https://spring.io/projects/spring-framework

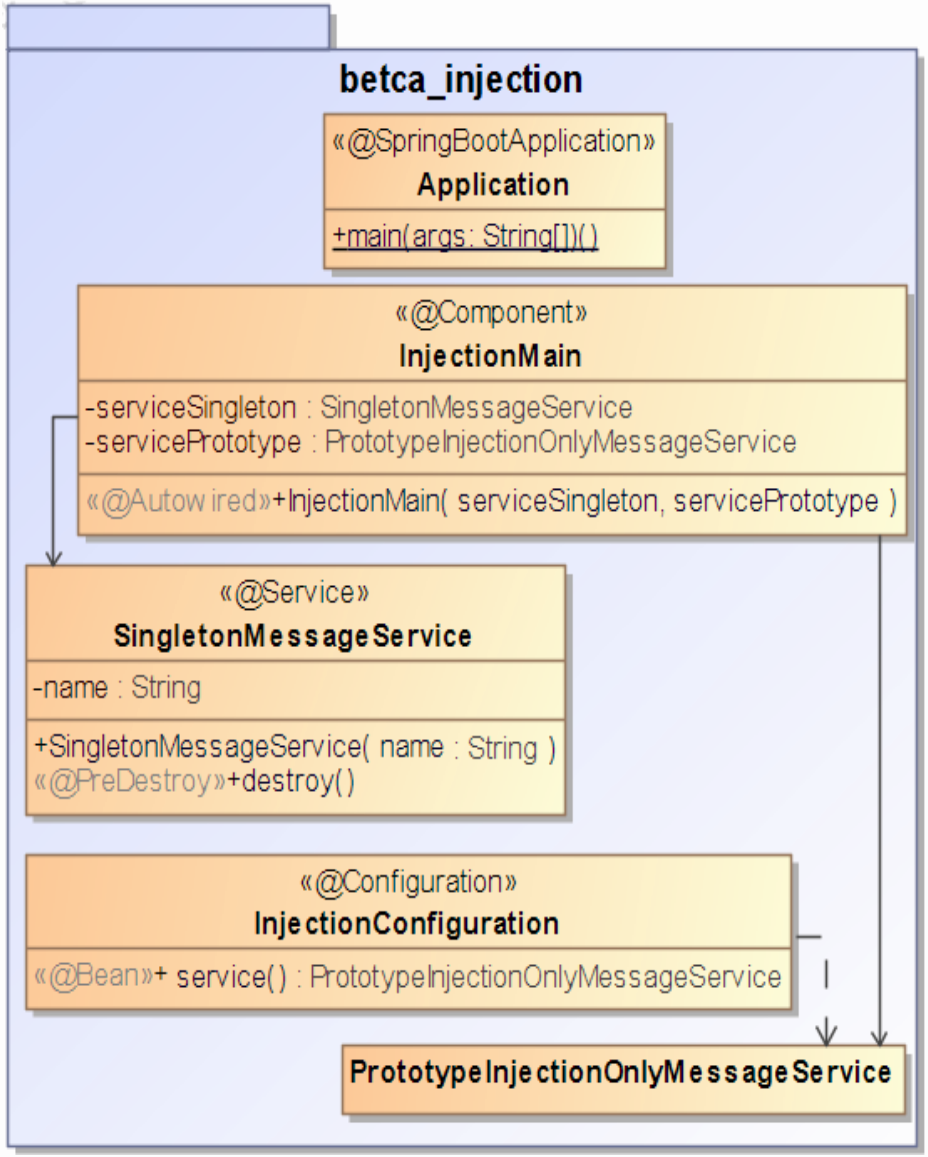
Spring Reactive

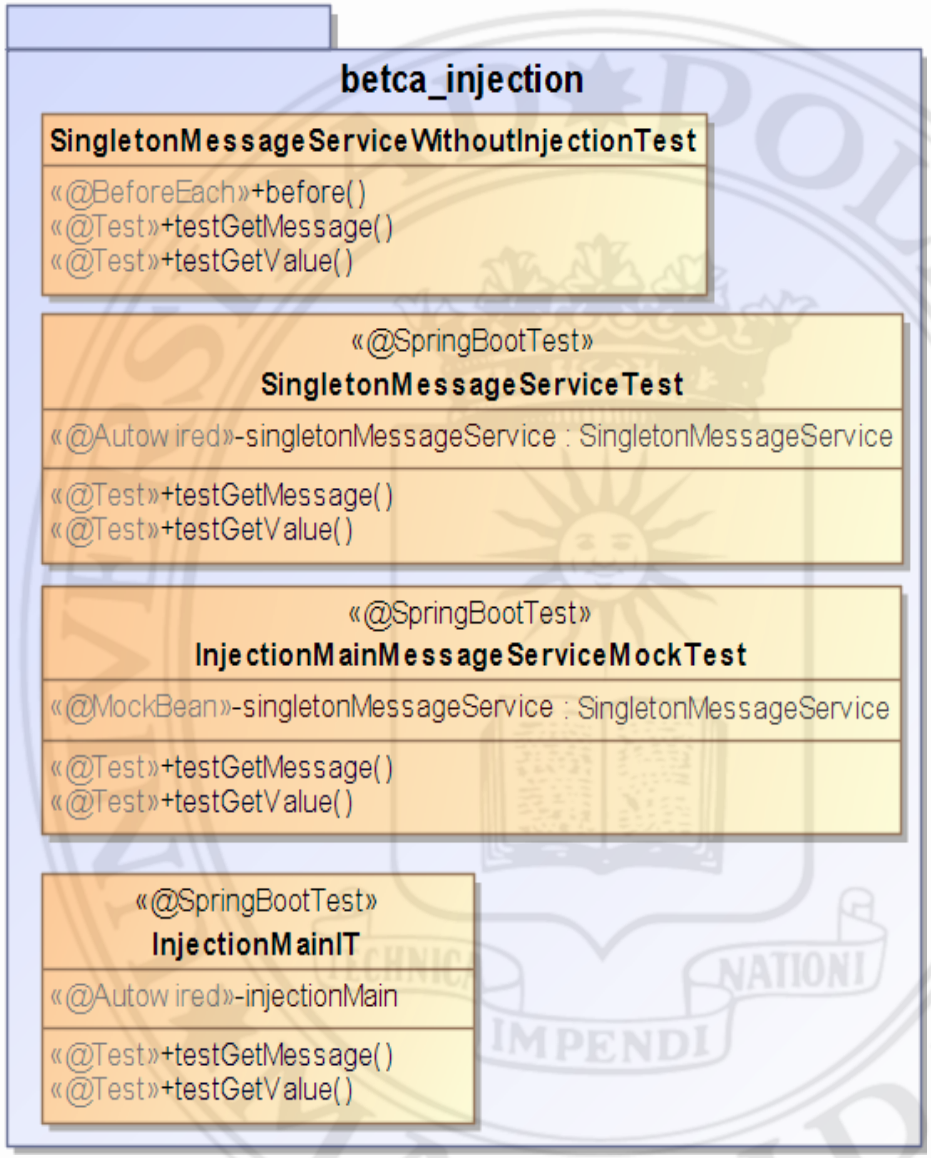
- Web asíncrona, no bloqueante.
- Pocos hilos (procesadores multi-core) pero masivo de conexiones.
- Contenedor Servlets 3.1+: *Netty*.
- Repositorios: *Reactive MongoDB*, *Cassandra*, *Redis*, *Couchbase*, *R2DBC* (*Postgres*, *MySQL*, *SQL Server*)

Spring MVC

- Web síncrona, bloqueante.
- Un hilo por petición.
- Contenedor Servlets: *Tomcat*, *Jetty*…
- Repositorios: *JDBC* (*Postgres*, *MySQL*, *SQL Server*), *JPA*, *MongoDB*.

Spring Framework
Inyección de dependencias





Inyección de dependencias

https://github.com/miw-upm/betca-spring

- **betca-injection**
- **https://start.spring.io/**

Inyección de Dependencias en práctica

- Crear un servicio con un método que devuelve un *String* con el día de la semana, crear un test para probarlo (*LocalDate.now().getDayOfWeek().toString()*)
- Crear un componente que utiliza el servicio anterior y tiene un método que indica si es fin de semana (*isWeekEnd*), crear *test unitario* y de *integración*.
- Añadir la propiedad "*miw.author*=***" con el nombre del autor en el fichero *application.properties* y añadir al servicio un segundo método que devuelve un *String* con el nombre del autor, ampliar test de prueba.

Persistencia
Spring Framework

JTA

- Java Transaction API
- Soporta el manejo de transacciones tanto de forma declarativa como programado.

JDBC

- Java Database Connectivity.
- Nos proporciona una capa de abstracción para acceder mediante SQL.
- ***String name = jdbcTemplate.queryForObject("select name from actor where id = ?", 1212L);***

R2DBC

- Reactive Relational Database Connectivity.
- Nos da acceso a BD SQL utilizando patrones reactivos.
- ***Flux<String> names = client.sql("SELECT name FROM person") .all();***

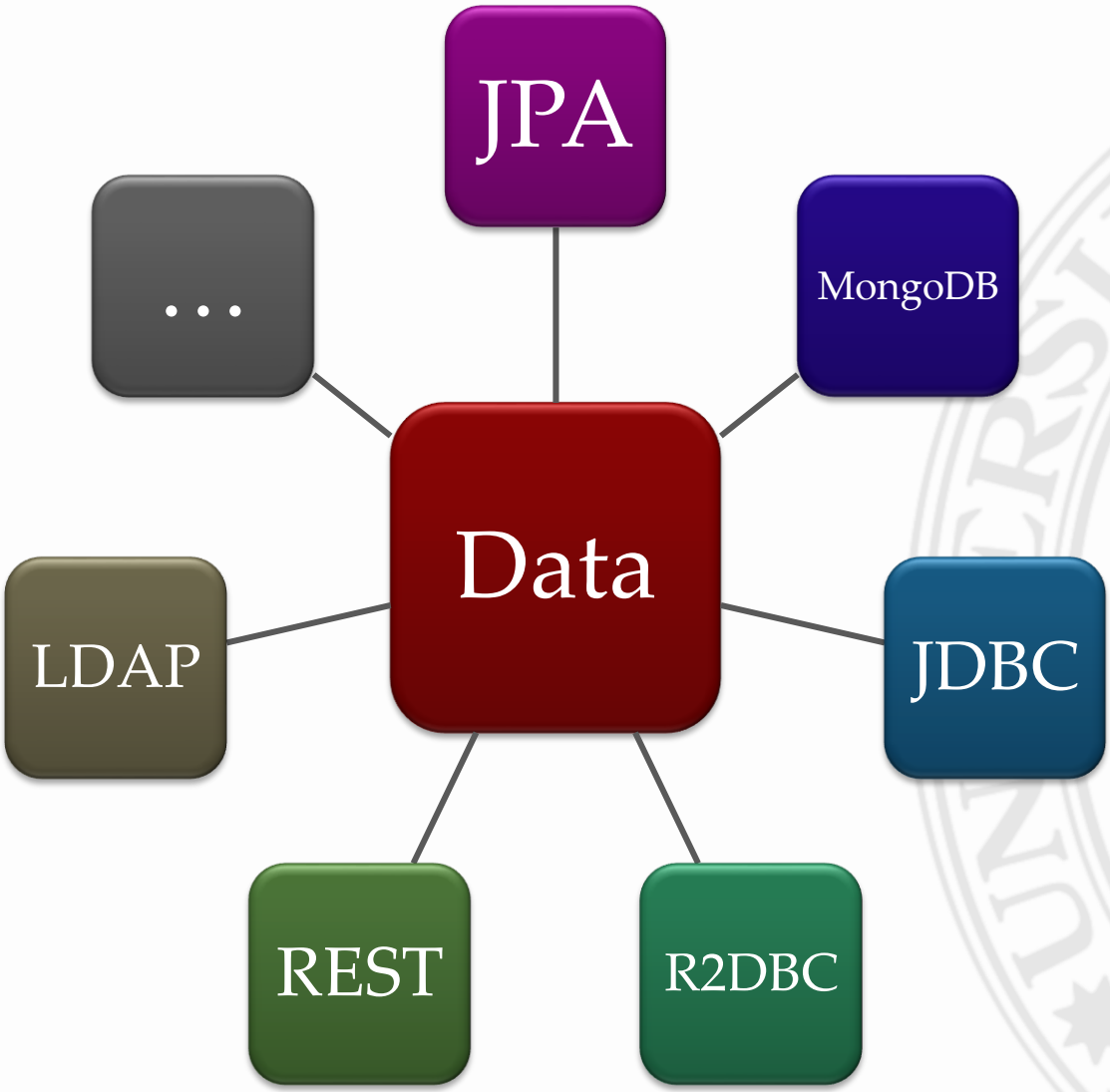
ORM

- Object Relational Mapping
- Soporta Java Persistence API (JPA) y Hibernate nativo

OXML

- Object-XML Mapping
- Nos proporciona la serialización XML (Marshaller & Unmarshaller)

Persistencia
Spring Data



Spring Data

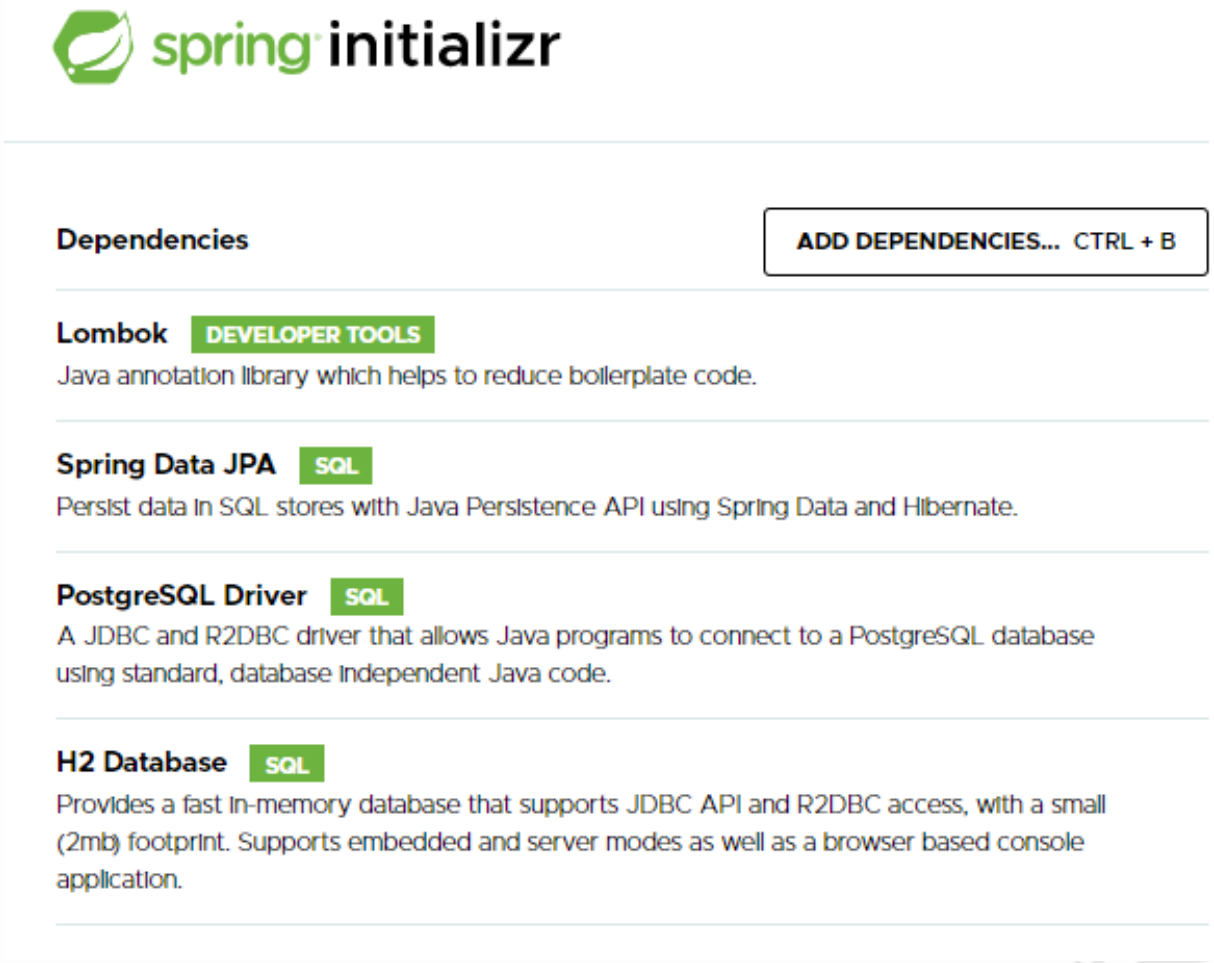
Proporciona un modelo de acceso a datos de alto nivel

DAO - Data Access Object
- Ofrece implementación completa

JPA - Java Persistence API
- Interfaces: javax.persistence.*
- Implementaciones: Hibernate, TopLink, OpenJPA, ObjectDB

MongoDB
- Hu**mongo**us: NoSQL, esquema libre
- BSON (Binary)

Spring Data
JPA. Dependencias



src/*/resources

- application-dev.properties
- test.properties
  - *@TestPropertySource*

Library: JAR

- *@SpringBootApplication)*
- *@TestConfig*

Package

- *Jakarta (a partir Spring-boot 3)*

Spring Data
Database. Configuración

H2 (test.properties)

- **Database H2 Embedded**

```
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.urlj=dbc:h2:mem:testdb
#spring.datasource.username=sa
#spring.datasource.password=
```

Postgresql (application-dev.properties)

- **Database Postgresql external**

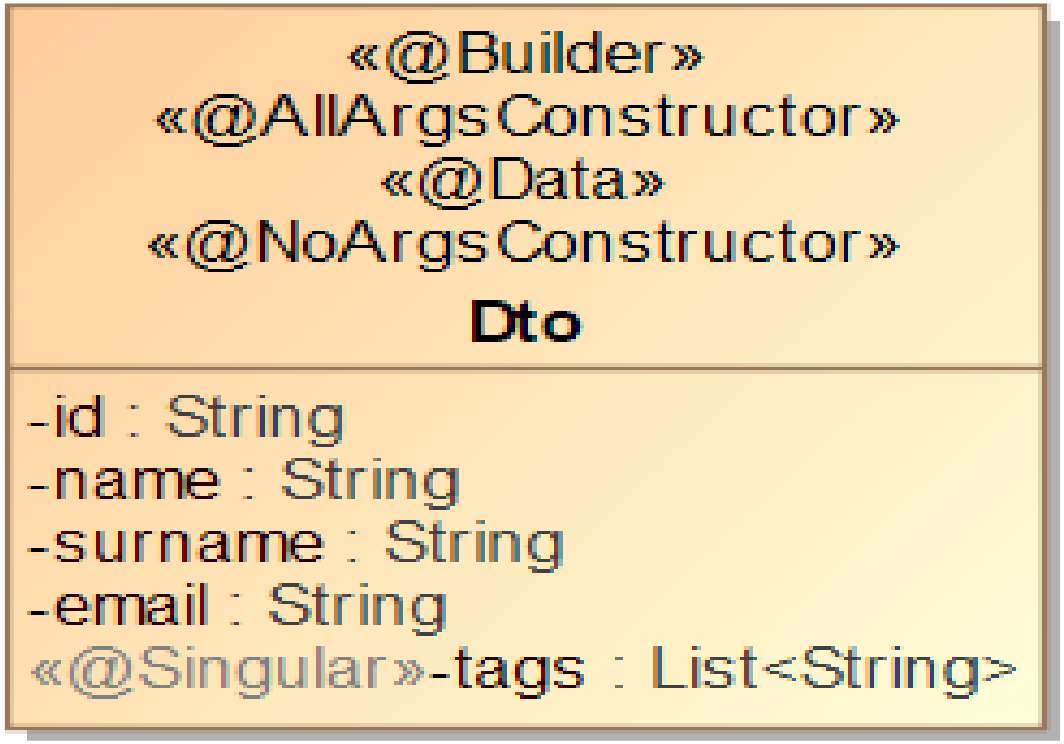
```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:postgresql://localhost:5432/betca
spring.datasource.username=postgres
spring.datasource.password=
```

Postgresql (application-prod.properties)

- **Database Postgresql in server**

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=${DATABASE_URL}
```

Spring Data
Proyecto Lombok



Dto.builder().id("1").tag("tag1").tag("tag2").build()

Proyecto Lombok

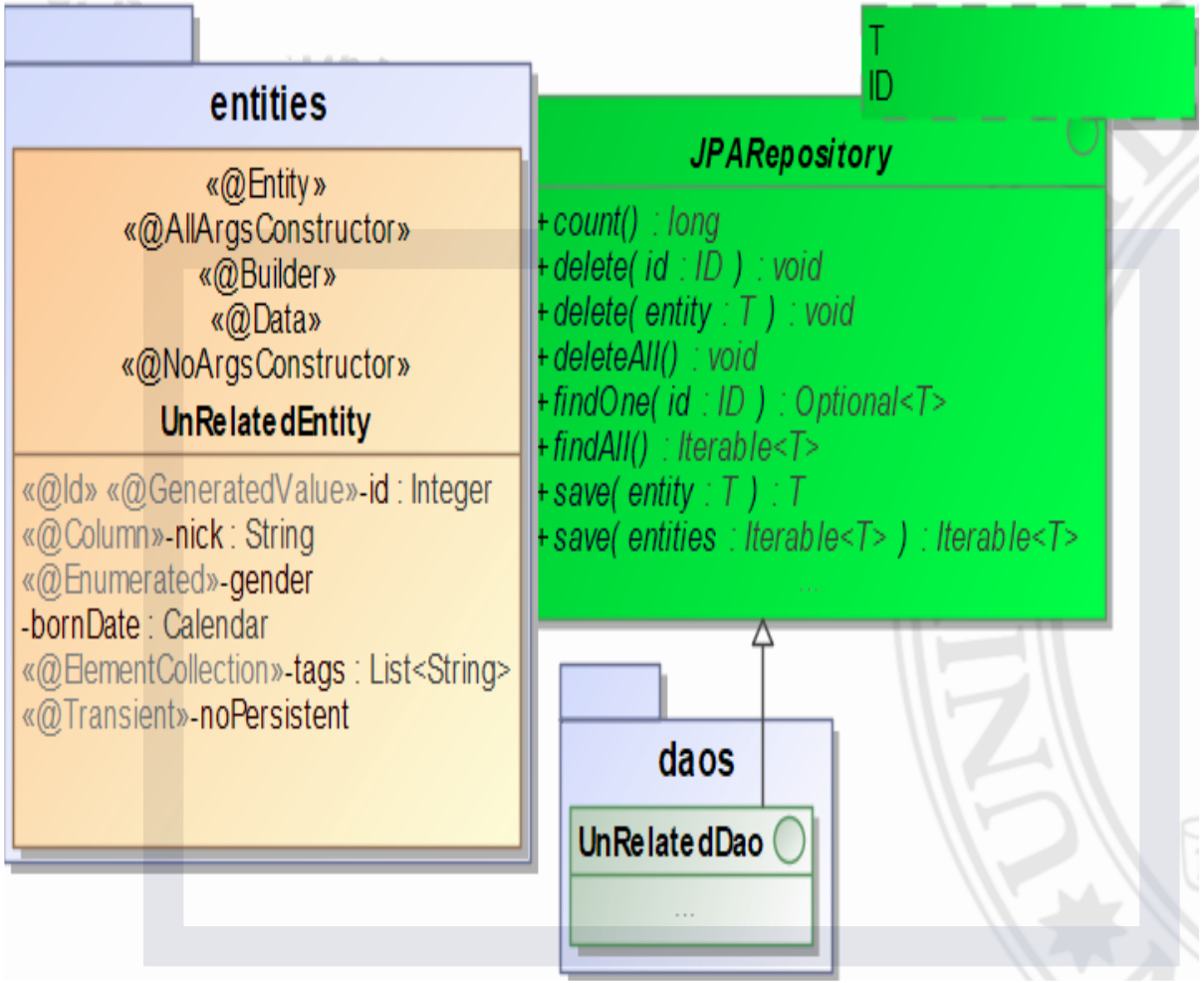
https://projectlombok.org/

- **@Data**
  - *@Getter*
  - *@Setter*
  - *@ToString*
  - *@EqualsAndHashCode*
  - *@RequiredArgsConstructor*
- @NoArgsConstructor
- @AllArgsConstructor
- @Builder
  - @Singular
- @Value

Spring Data
Proyecto Lombok

```
@Builder
@Data  // @ToString, @EqualsAndHashCode, @Getter, and
@Setter on all non-final fields, @RequiredArgsConstructor
@EqualsAndHashCode(onlyExplicitlyIncluded = true)
@NoArgsConstructor
@AllArgsConstructor
public class DtoWithLombok {
   @EqualsAndHashCode.Include // only id
   private String id; //@Setter(AccessLevel.NONE) for exceptions
   private String name;
   private String surName;
   private String email;
   @Singular // .tag().tag().tag()
   private List<String> tags;
}
```

Spring Data
JPA

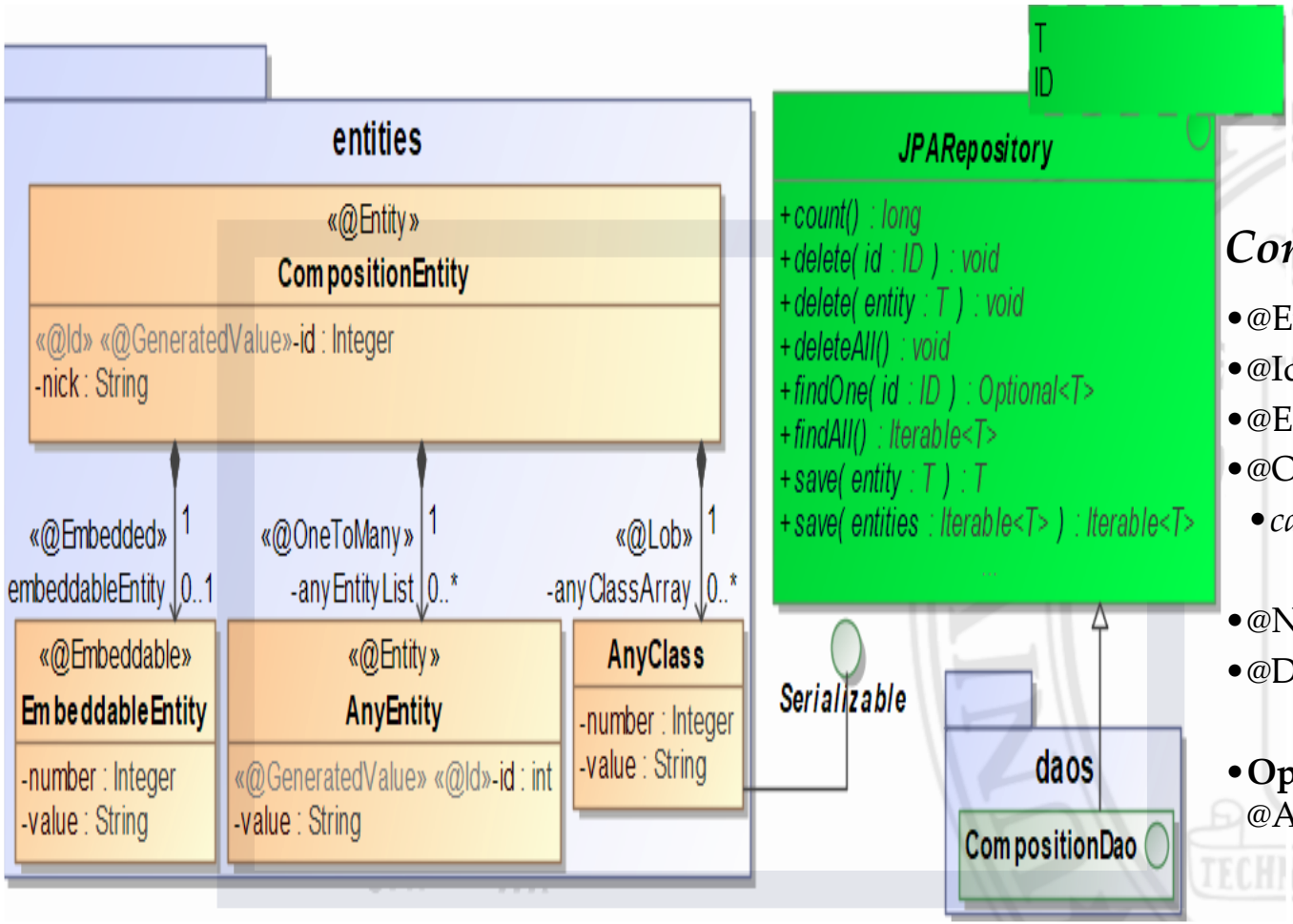


Sin Relación

- @Entity
- @Id

- @NoArgsConstructor
- @Data

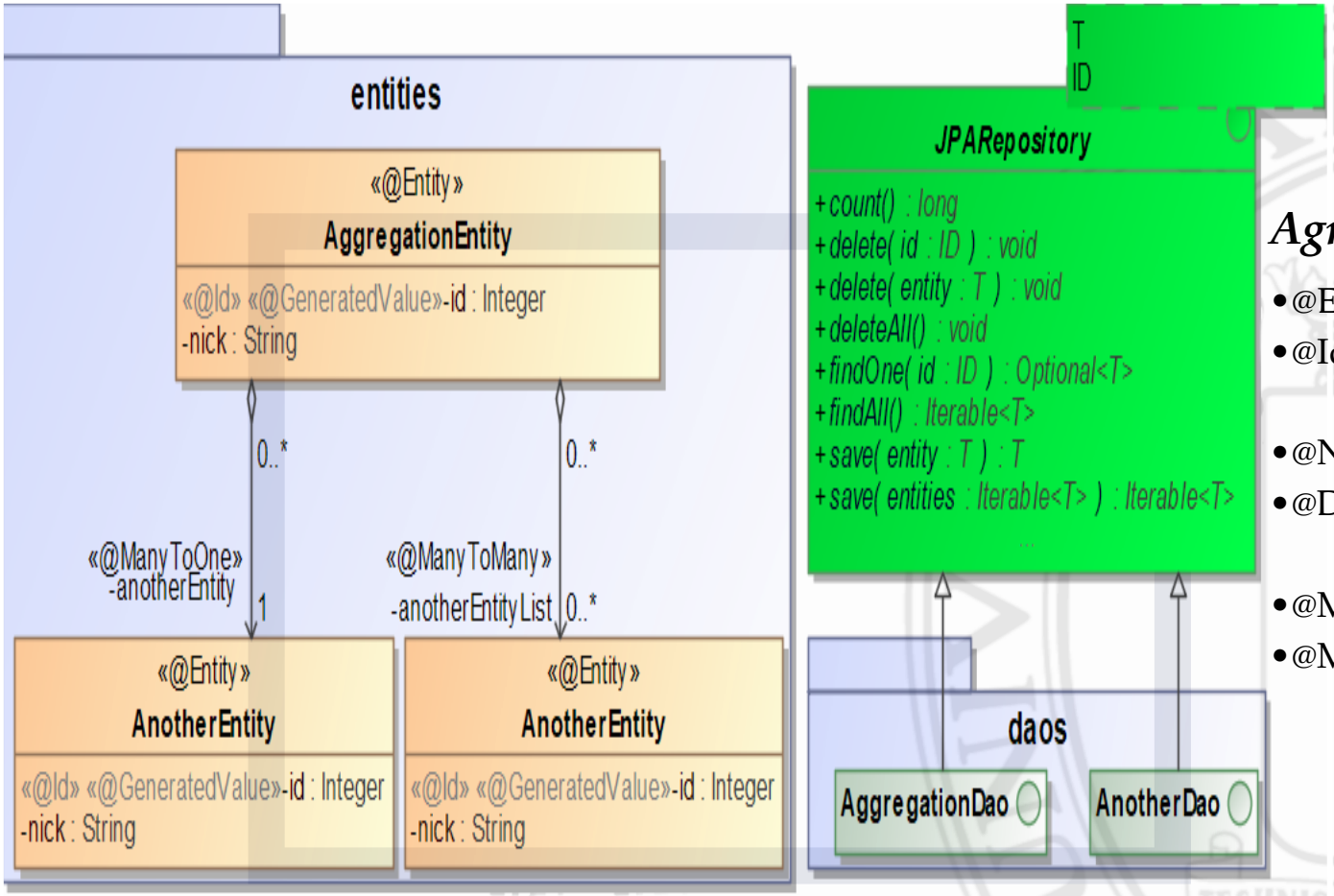
Spring Data
JPA



Composición

- @Entity o @Embeddable
- @Id
- @Embedded o @Lob
- @OneToMany
  - *cascade=cascadeTypeALL*
- @NoArgsConstructor
- @Data
- **Opcional**: @Builder, @AllArgsConstructor

Spring Data
JPA



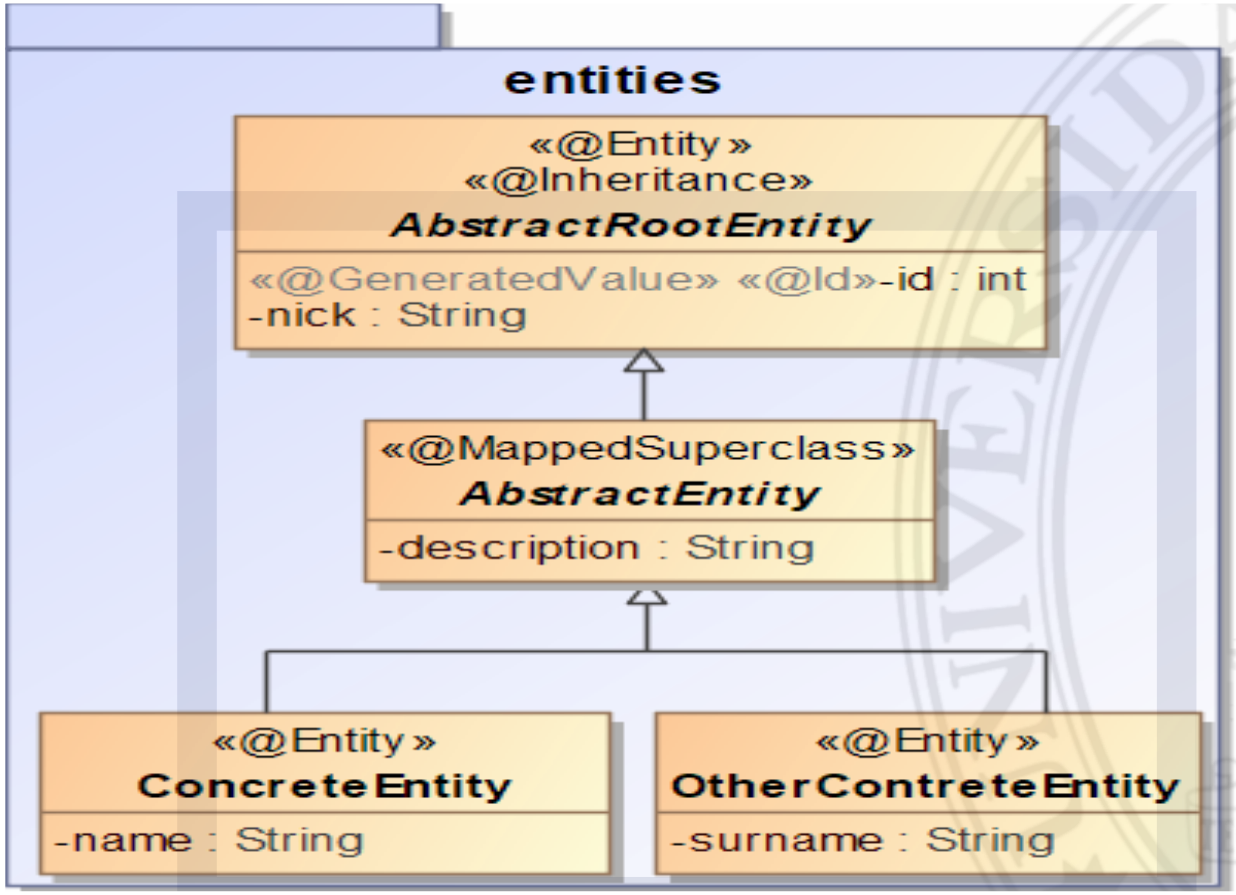
Agregación

- @Entity
- @Id

- @NoArgsConstructor
- @Data

- @ManyToOne
- @ManyToMany

Spring Data
JPA



Herencia

- @Entity
- @Id
- @Inheritance
  - SINGLE_TABLE
  - JOINED
- @MappedSuperclass

Spring Data

Postgresql. *https://www.postgresql.org/*

Docker

Iniciar el contenedor docker con postgres

> docker exec -it postgres-db psql -U postgres

Consola

- \? *Help*
- \l *Database list*
- \connect <db> <usr>
- \dt Table list
- \d <table>; describe
- \q exit
- create database <db>;
- drop database <db>;
- select * from <table>;
- delete from <table>;

- \timing activa

Spring Data

Postgresql. *https://www.postgresql.org/*

Bajarse el fichero *.zip

https://www.enterprisedb.com/download-postgresql-binaries

Descomprimir en el equipo, FUERA del proyecto

Inicializar las Bases de Datos

>bin> .\initdb -D ./data -U postgres -E UTF8

Arrancar – Parar

>bin> .\pg_ctl -D ./data -l logs start

>bin> .\pg_ctl -D ./data -l logs stop

Consola

>bin> chcp 1252

>bin> .\psql -U postgres

Crear Database “betca” & “tpv”

Consola

- \? *Help*
- \l *Database list*
- \connect <db> <usr>
- \dt Table list
- \d <table>; describe
- \q exit
- create database <db>;
- drop database <db>;
- select * from <table>;
- delete from <table>;

- \timing activa

JPA

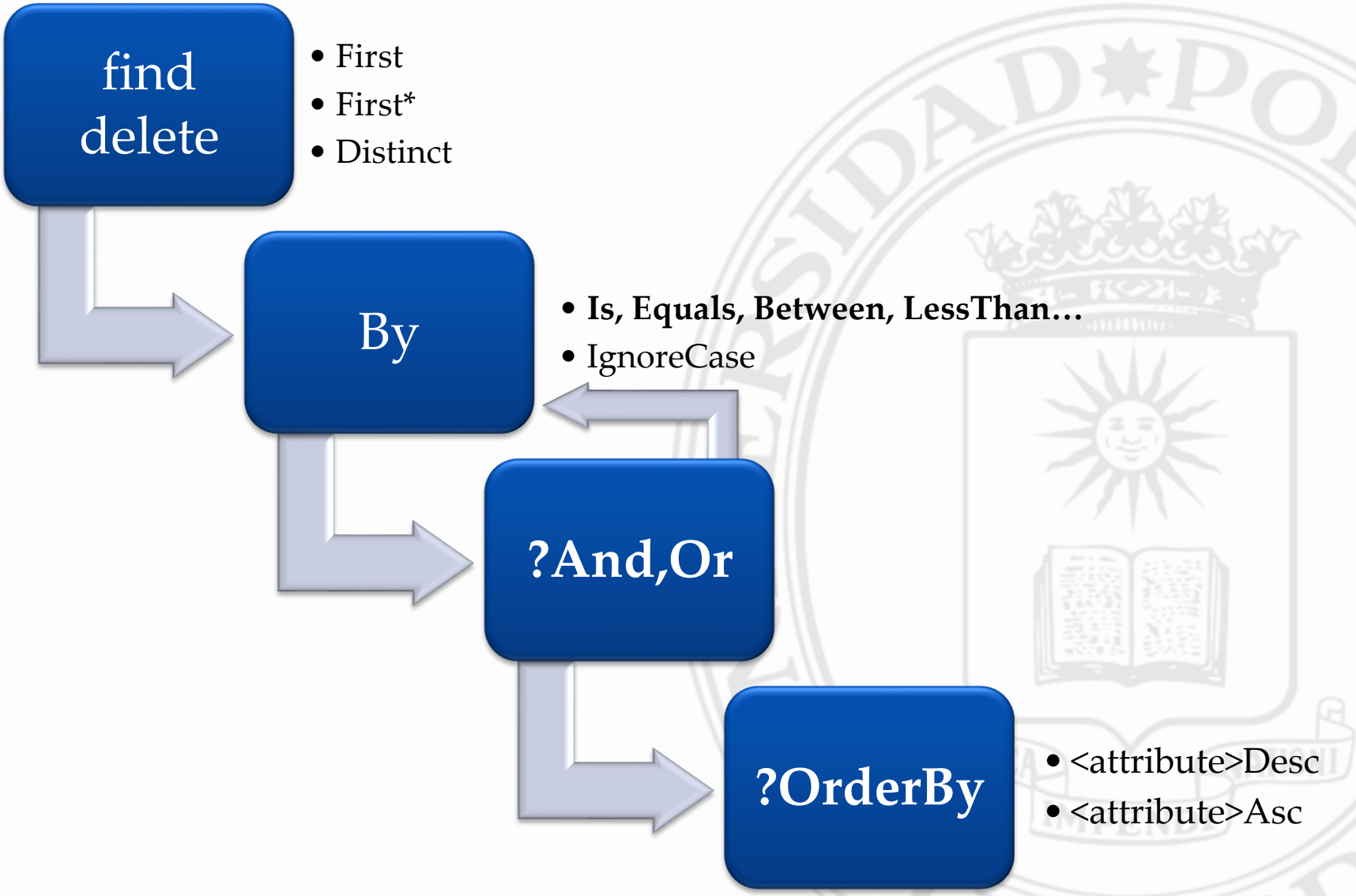
https://github.com/miw-upm/betca-spring

- **{→} Demo de Postgres**
- **Proyecto: betca-jpa**
- **Plugin lombok en IntelliJ (*Settings/plugins*)**

JPA en práctica

- Crear nuevas entidades con tests. **Lanzar los Test con mucha frecuencia!!!**
  - Sin relación
  - Con Composición
  - Con Agregación
- Postgres
  - Instalar Postgres y crear las BD “betca” & “tpv”
  - Comentar la activación de H2 en *test.properties*
  - Arrancar el motor de Postgres y lanzar los test y mirar las tablas creadas

Spring Data
JPA



Spring Data
JPA

Búsquedas por nombre de método

- **findBy**Lastname**And**Firstname, **findBy**Lastname**Or**Firstname
- **findBy**Firstname, **findBy**Firstname**Is**, **findBy**Firstname**Equals**
- **findBy**StartDate**Between**
- **findBy**Age**LessThan**, **findBy**Age**LessThanEqual**, **findBy**Age**GreaterThan**, **findBy**Age**GreaterThanEqual**
- **findBy**StartDate**After**, **findBy**StartDate**Before**
- **findBy**Age**IsNull**, **findBy**Age**NotNull**, **findBy**Age**IsNotNull**
- **findBy**Firstname**Like**, **findBy**Firstname**NotLike**
- **findBy**Firstname**StartingWith**, **findBy**Firstname**EndingWith**
- **findBy**Firstname**Containing**
- **findBy**Age**OrderBy**Lastname**Desc**
- **findBy**Lastname**Not**
- **findBy**Age**In**(Collection<Age> ages), **findBy**Age**NotIn**(Collection<Age> ages)
- **findBy**Active**True**(), **findBy**Active**False**()
- **findBy**Firstname**IgnoreCase**

- List<Entity> **findBy**Atr1(String atr1);
- int **deleteBy**Atr1**GreaterThan**(int value);

Spring Data
JPA

Paginación

- *List<UnRelatedEntity> findByIdGreaterThan(int id, **Pageable pageable**);*
- ***dao**.findByIdGreaterThan(90, **PageRequest.of(1, 5)**);*
- *dao.count();*

JPQL (Java Persistence Query Language)

- *@Query("select u.id from other_name_for_unrelatedentity u where u.id > ?1 and u.id < ?2")*
- List<Integer> findIdByIdBetween(int initial, int end);
- *@Modifying @Query(value="delete from other_name_for_unrelatedentity u where u.nick = ?1")*
- int deleteByNickQuery(String nick);

SQL

- *@Query(value = "SELECT * FROM un_related_entity WHERE NICK = ?1", **nativeQuery = true**)*

Spring Data
JPA

JPQL

- FindByLastnameAndFirstname …where x.lastname = ?1 and x.firstname = ?2
- findByLastnameOrFirstname …where x.lastname = ?1 or x.firstname = ?2
- findByFirstname, findByFirstnameIs, findByFirstnameEquals …where x.firstname = ?1
- findByStartDateBetween …where x.startDate between ?1 and ?2
- findByAgeLessThan …where x.age < ?1
- findByAgeLessThanEqual …where x.age <= ?1
- findByAgeGreaterThan …where x.age > ?1
- findByAgeGreaterThanEqual …where x.age >= ?1
- findByStartDateAfter …where x.startDate > ?1
- findByStartDateBefore …where x.startDate < ?1
- findByAgeIsNull …where x.age is null
- findByAgeNotNull, findByAgeIsNotNull …where x.age not null
- findByFirstnameLike …where x.firstname like ?1
- findByFirstnameNotLike …where x.firstname not like ?1
- findByFirstnameStartingWith …where x.firstname like ?1(parameter bound with appended %)
- findByFirstnameEndingWith …where x.firstname like ?1(parameter bound with prepended %)
- findByFirstnameContaining …where x.firstname like ?1(parameter bound wrapped in %)
- findByAgeOrderByLastnameDesc …where x.age = ?1 order by x.lastname desc
- findByLastnameNot …where x.lastname <> ?1
- findByAgeIn(Collection<Age> ages) …where x.age in ?1
- findByAgeNotIn(Collection<Age> ages) …where x.age not in ?1
- findByActiveTrue() …where x.active = true
- findByActiveFalse() …where x.active = false
- findByFirstnameIgnoreCase …where UPPER(x.firstame) = UPPER(?1)

Spring Data
JPA

JPQL - JOIN

- SELECT agg.nick FROM AggregationEntity agg WHERE agg.anotherEntity.value = ?1
- SELECT anotherList.value FROM AggregationEntity agg JOIN agg.anotherEntityList anotherList
- SELECT another.value FROM AggregationEntity agg JOIN agg.anotherEntity another JOIN agg.anotherEntityList anotherList WHERE anotherList.value = ?1

JPA. Consultas

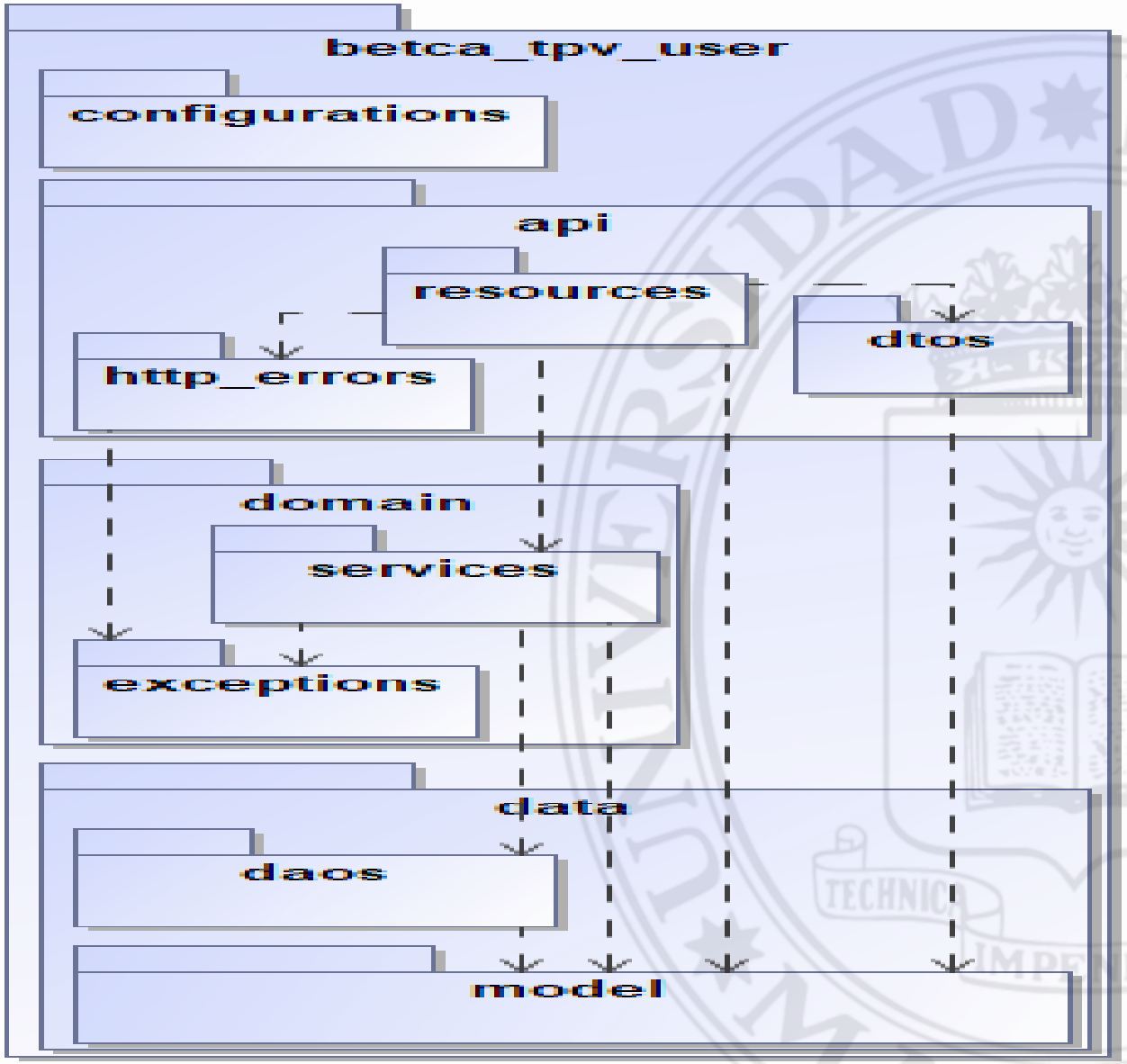
https://github.com/miw-upm/betca-spring

- **betca-jpa**

JPA en práctica

- Definir varias consultas para la entidad: *UnRelatedEntity*
- Definir una consulta con paginación para la entidad: *UnRelatedEntity*
- Definir varias consultas mediante JPQL para la entidad *UnRelatedEntity*
- Definir una consulta en SQL para la entidad *UnRelatedEntity*
- **Realizar los test correspondientes**

Spring. TPV-user
BD



JPA. betca-tpv-user ✎ {→}

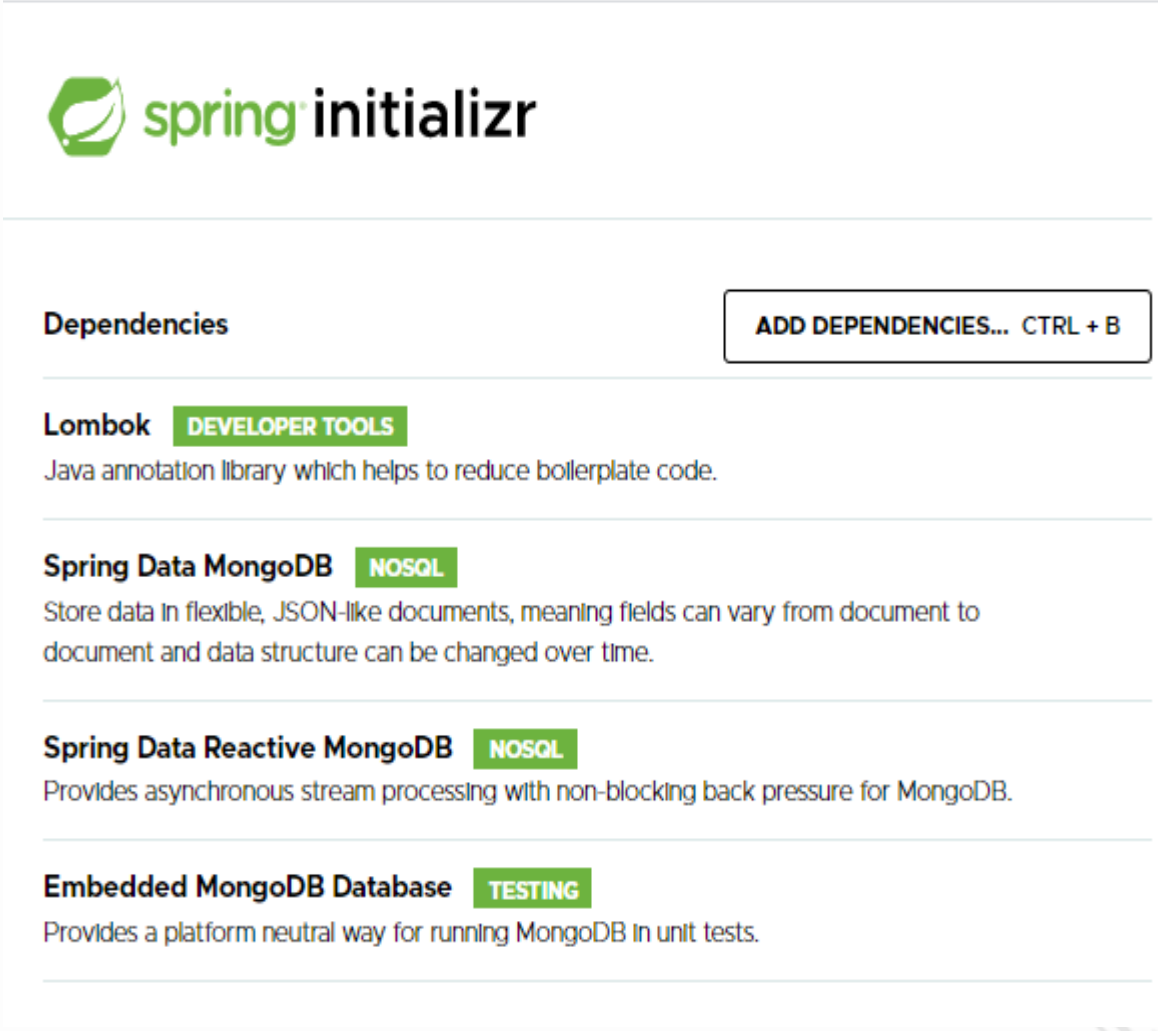
https://github.com/miw-upm/betca-tpv-user

- **betca-tpv-user::data**

✎ JPA en práctica

- Capa **Data** de la práctica **TPV-user**

Spring Data
MongoDB. Dependencias



src/*/resources

- application-dev.properties
- test.properties
  - *@TestPropertySource*

Library: JAR

- *@SpringBootApplication*

Embedded (Spring-boot 3)

- *de.flapdoodle.embed.mongo.spring30x*

Spring Data
Database. Configuración

MongoDB (*.properties)

- MondoDB

```
# spring.data.mongodb.database=test
# spring.data.mongodb.host=localhost
# spring.data.mongodb.port=27017
# spring.data.mongodb.username=
# spring.data.mongodb.password=
```

Spring Data
Mongodb. *https://www.mongodb.com/*

Docker

Iniciar el contenedor con mongodb

```
> docker exec -it mongo-db mongosh "mongodb://mongo:mongo@localhost:27017/tpv?authSource=admin"
```

Consola

- help
- db.version()
- db.stats()
- show databases
- use <myBD>
- show collections
- db.dropDatabase()
- db.coll1.drop()
- db.coll1.find()
- db.coll1.find().pretty()
- db.coll1.count()

Spring Data
Mongodb. *https://www.mongodb.com/*

Bajarse el fichero *.zip

https://www.mongodb.com/try/download/community

Descomprimir en equipo, FUERA del proyecto

Crear la carpeta de BD

>mongodb>mkdir data\db

Arrancar – Parar

>bin> mongod --dbpath **/data/db | Ctrl^C

Consola

>bin> mongo

Consola

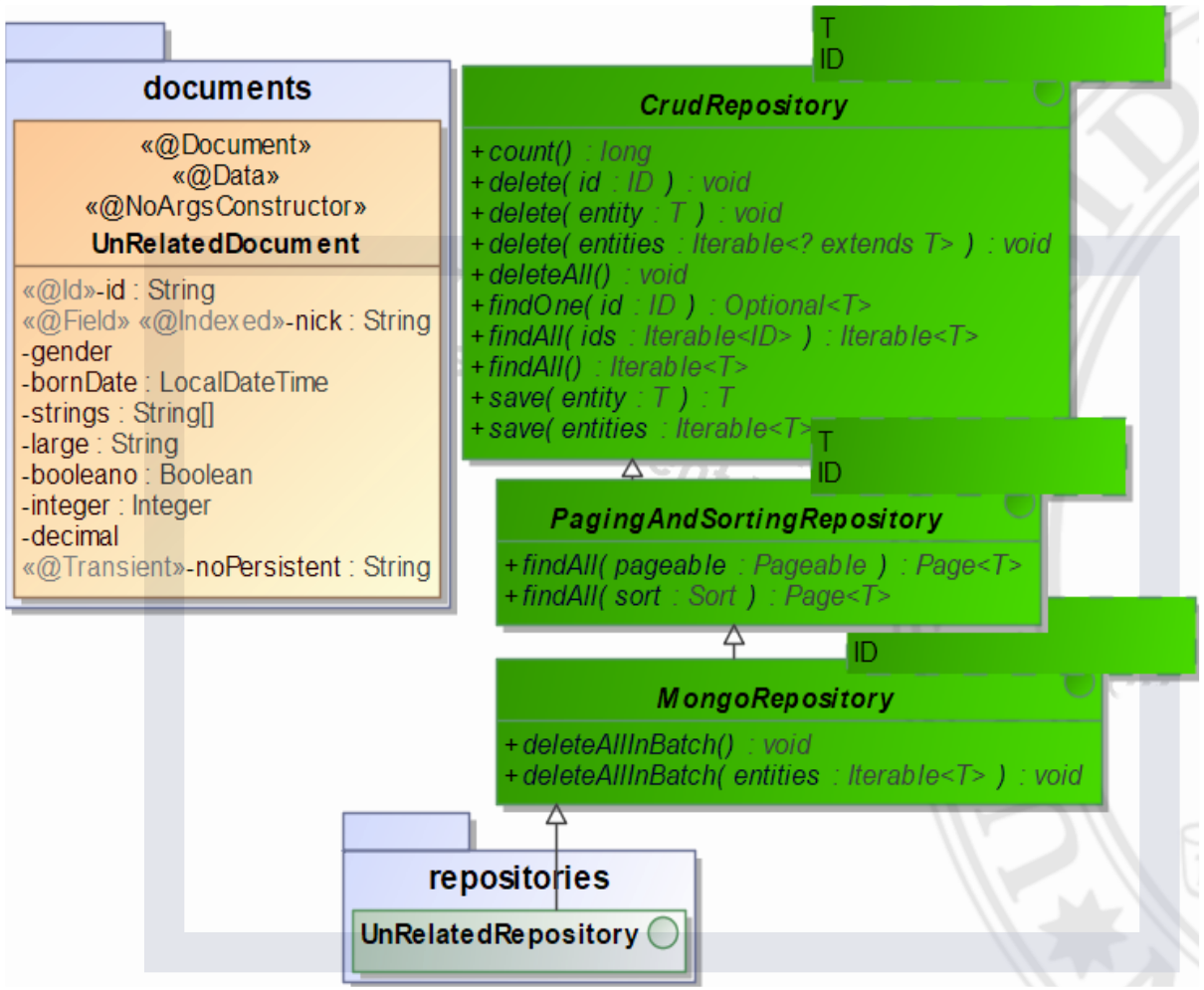
- help
- db.version()
- db.stats()
- show databases
- use <myBD>
- show collections
- db.dropDatabase()
- db.coll1.drop()
- db.coll1.find()
- db.coll1.find().pretty()
- db.coll1.count()

Spring Data
Mongodb

Consola

- db.coll1.insert({ name:"jesus" })
  - *{"_id" : ObjectId("5a86ad2e9d610a0fe33aa1ea"), "name" : "jesus"}*
- db.coll1.insert({ _id:1 , name:"jesus" })
  - *{ "_id" : 1, "name" : "jesus" }*
- db.coll1.insert ({ _id: "2", name:"ny", central: { _id:1, location:"Madrid"} })
  - *{ "_id" : "2", "name" : "ny", "central" : { "_id" : 1, "location" : "Madrid" } }*
- db.coll1.insert({ _id: 3, name:"ny", central: [{ _id:2, location:"Madrid"},{_id:3, location:"Sevilla"}] })
  - *{"_id" : 3, "name" : "ny", "central" : [{ "_id" : 2, "location": "Madrid" }, { "_id" : 3, "location" : "Sevilla" }] }*
- db.other.insert({ _id: 1, name:"nameInOther" })
- db.coll1.insert({ _id: 4, name:"nameInColl1", other: DBRef("other",1) })
  - *{ "_id" : 4, "other" : DBRef("other", 1) }*

Spring Data
Mongodb

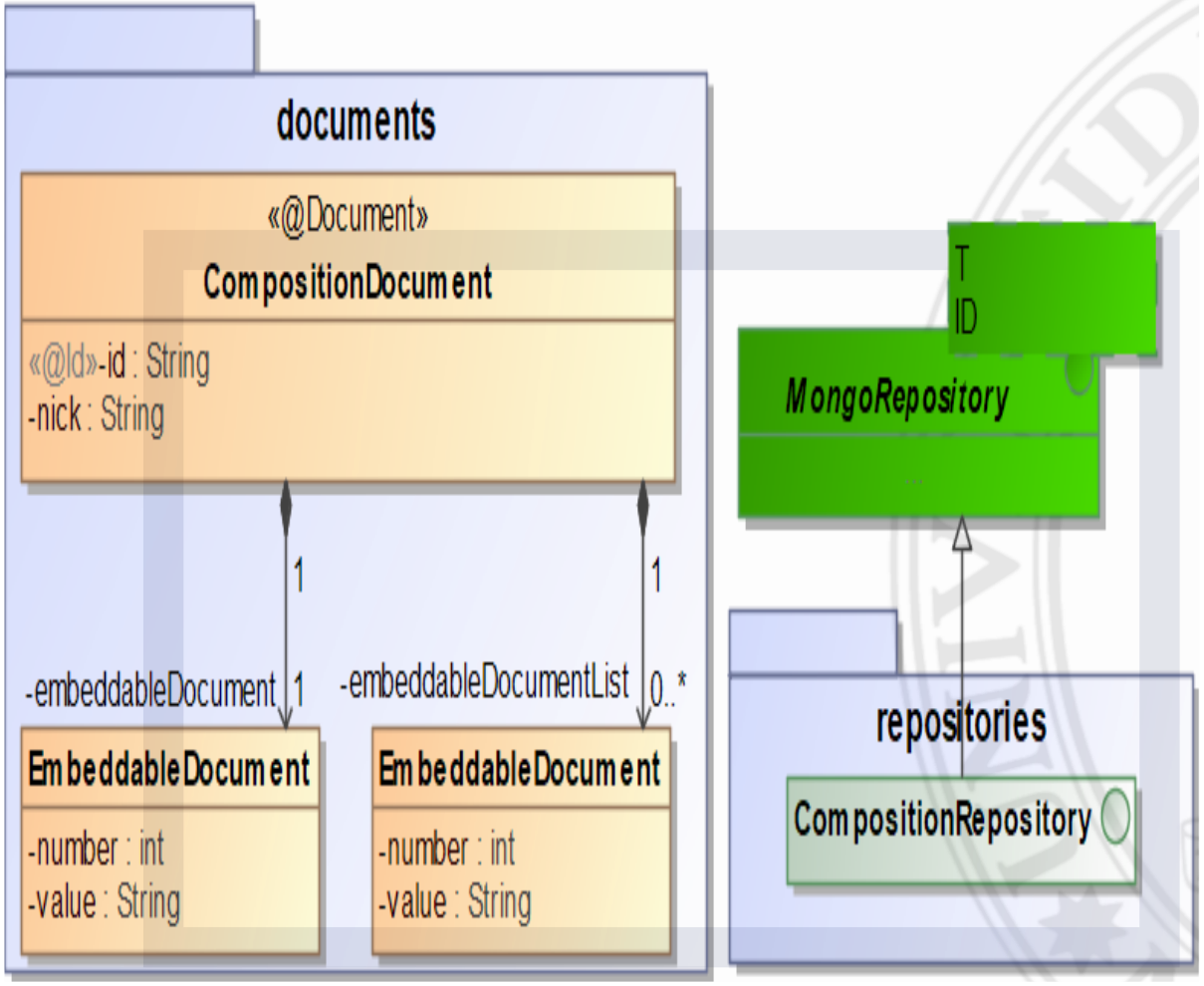


Sin Relación

- @Document
- @Id

- @NoArgsConstructor
- @Data

Spring Data
Mongodb

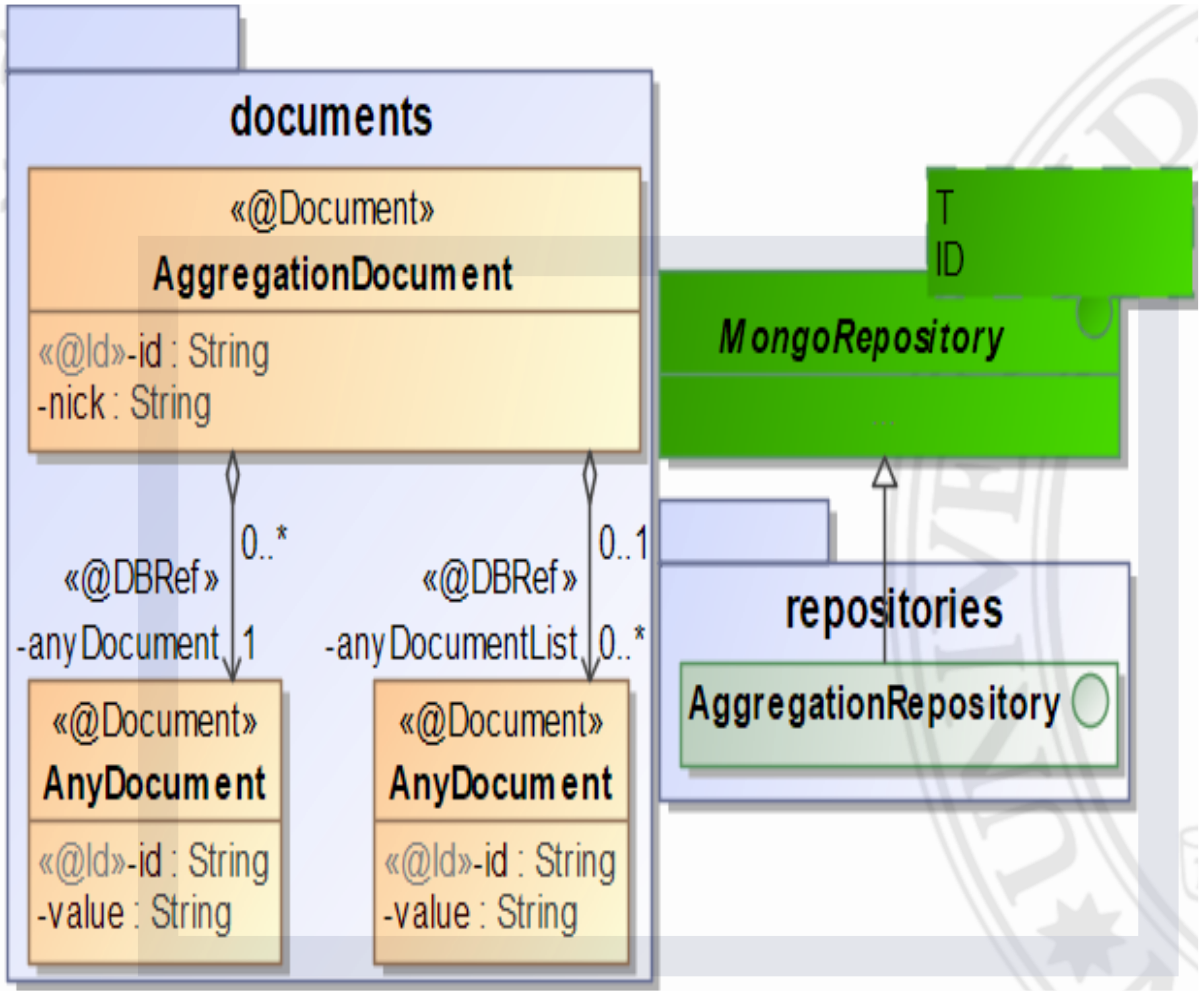


Composición

- @Document
- @Id

- @NoArgsConstructor
- @Data

Spring Data
Mongodb



Agregación

- @Document
- @Id
- @DBRef

- @NoArgsConstructor
- @Data

Spring Data
Mongodb

Búsquedas por nombre de método VS Json Query

- **findByBirthdateAfter(Date date)** {"birthdate" : {"$gt" : date}}
- **findByAgeGreaterThan(int age)** {"age" : {"$gt" : age}}
- **findByAgeGreaterThanEqual(int age)** {"age" : {"$gte" : age}}
- **findByBirthdateBefore(Date date)** {"birthdate" : {"$lt" : date}}
- **findByAgeLessThan(int age)** {"age" : {"$lt" : age}}
- **findByAgeLessThanEqual(int age)** {"age" : {"$lte" : age}}
- **findByAgeBetween(int from, int to)** {"age" : {"$gt" : from, "$lt" : to}}
- **findByAgeIn(Collection ages)** {"age" : {"$in" : [ages…]}}
- **findByAgeNotIn(Collection ages)** {"age" : {"$nin" : [ages…]}}
- **findByFirstnameNotNull()** {"firstname" : {"$ne" : null}}
- **findByFirstnameNull()** {"firstname" : null}
- **findByFirstnameLike(String name)** {"firstname" : name} (name as regex)
- **findByFirstnameNotLike(String name)** {"firstname" : { "$not" : name }} (name as regex)
- **findByFirstnameContaining(String name)** {"firstname" : name} (name as regex)
- **findByFirstnameNotContaining(String name)** {"firstname" : { "$not" : name}} (name as regex)
- **findByAddressesContaining(Address address)** {"addresses" : { "$in" : address}}
- **findByAddressesNotContaining(Address address)** {"addresses" : { "$not" : { "$in" : address}}}
- **findByFirstnameRegex(String firstname)** {"firstname" : {"$regex" : firstname }}
- **findByFirstname(String name)** {"firstname" : name}
- **findByFirstnameNot(String name)** {"firstname" : {"$ne" : name}}
- **findByActiveIsTrue()** {"active" : true}
- **findByActiveIsFalse()** {"active" : false}
- **findByLocationExists(boolean exists)** {"location" : {"$exists" : exists }}

Spring Data
Mongodb

Búsquedas por nombre de método VS Json Query

- *@Query("{nick:{$in:?0} }") // Query NOT necessary*
  ListUnRelatedDocument findByNickInCollection nicks
- *@Query"?#{ [0] == null ? { $where : 'true'} : { nick : {$regex:[0], $options: 'i'} } }"*
  ListUnRelatedDocument findByNickLikeIgnoreCaseNullSafe(String nick)
- *@Query"{$and:[" // allow NULL: all elements*
  "?#{ [0] == null ? {_id : {$ne:null}} : { nick : {$regex:[0], $options: 'i'} } },"
  "?#{ [1] == null ? {_id : {$ne:null}} : { large : {$regex:[1], $options: 'i'} } }"
  "] }"
  ListUnRelatedDocument findByNickLikeAndLargeLikeNullSafe(String nick, String large)

Mongodb. Consultas

https://github.com/miw-upm/betca-spring

- **betca-mongodb**

MongoDB en práctica

- Instalar Mongodb y observar como quedan las colecciones
- Definir varias consultas para la entidad: *UnRelatedEntity*
- Definir una consulta con paginación para la entidad: *UnRelatedEntity*
- Definir varias consultas mediante JSON Query para la entidad *UnRelatedEntity*
- **Realizar los test correspondientes**

MongoDB. betca-tpv-core {→}

https://github.com/miw-upm/betca-tpv-core

- **betca-tpv-core::persistencia**

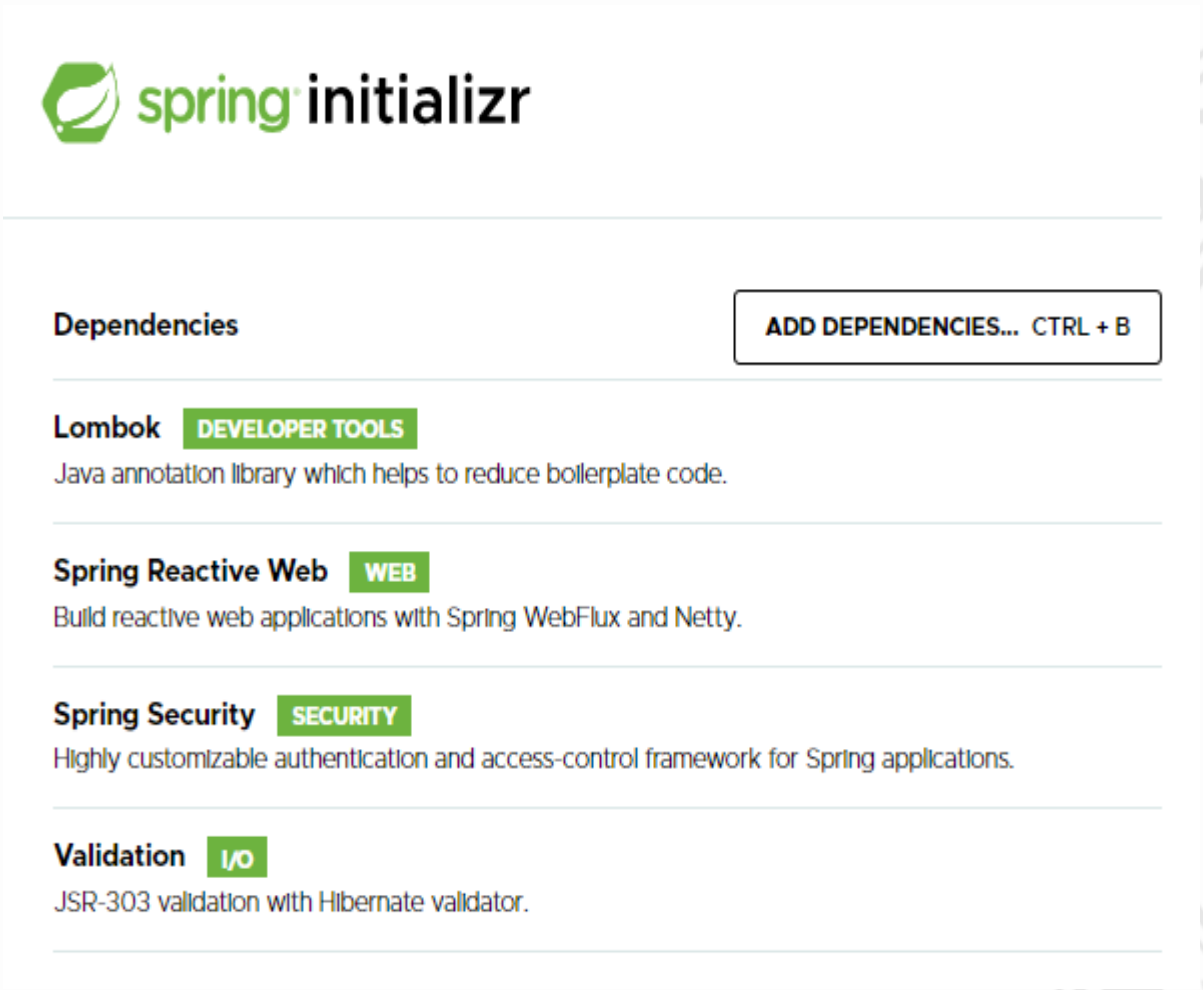
MongoDB en práctica

- Capa **Data** de la práctica **TPV-core**

Spring
Rest

Configuración	Ends Points	Errores	Validación	Seguridad
• Establecer la configuración de Spring y creación de los Beans.	• Establecer las **rutas de los End-points**. • Establecer los parámetros y cuerpos requeridos.	• Manejar los errores que pudieran ocurrir. • **JAMAS** se puede permitir un error no tratado.	• Validación de los datos de entrada.	• Control de acceso.

Spring
Rest



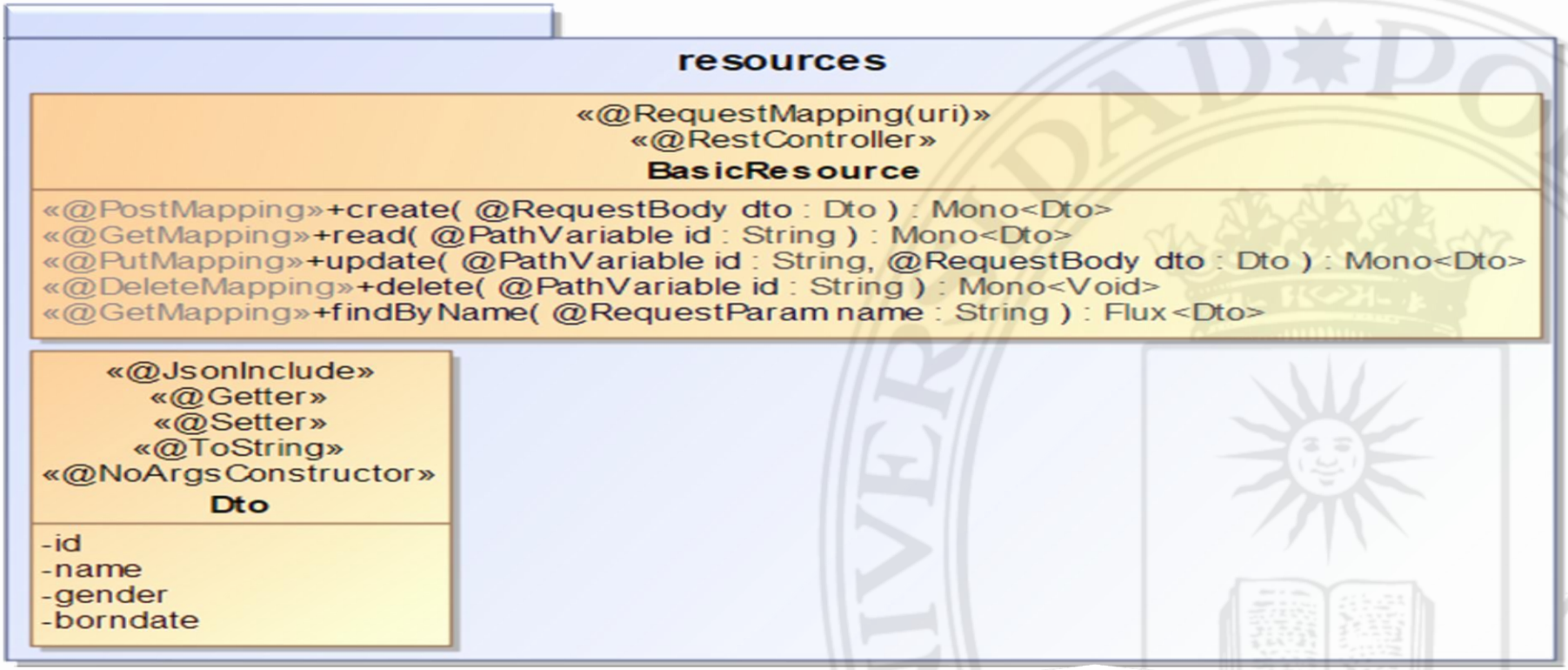
src/*/resources

- application.properties
- test.properties

@RestTestConfig

- @SpringBootTest
- @AutoConfigureWebTestClient
- @TestPropertySource

Rest
Spring



PUT /basic/666 {dto}

GET /basic/search?name=Miw&q=other

Rest. betca-spring
✎ {→}

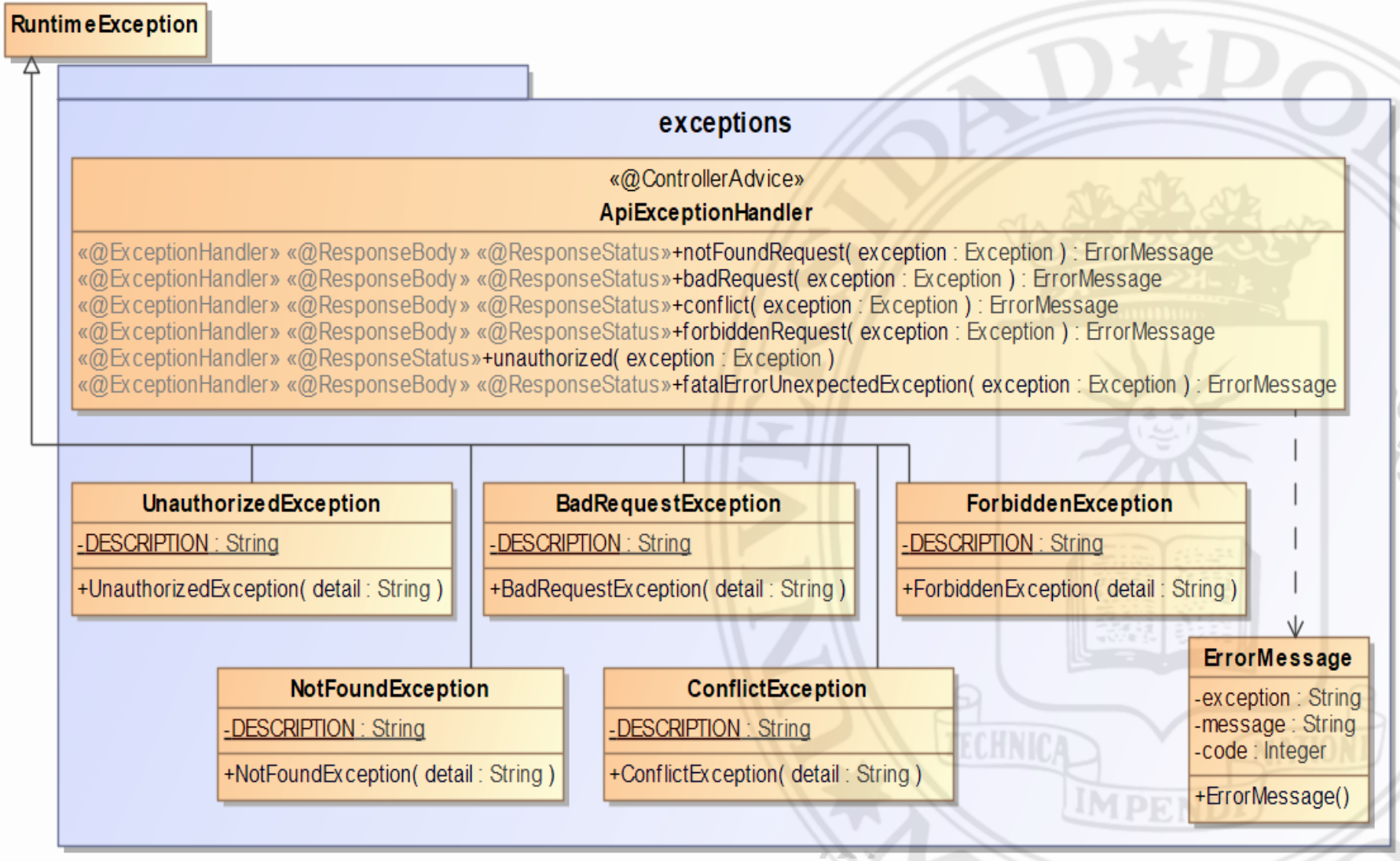
https://github.com/miw-upm/betca-spring

- **betca-rest**

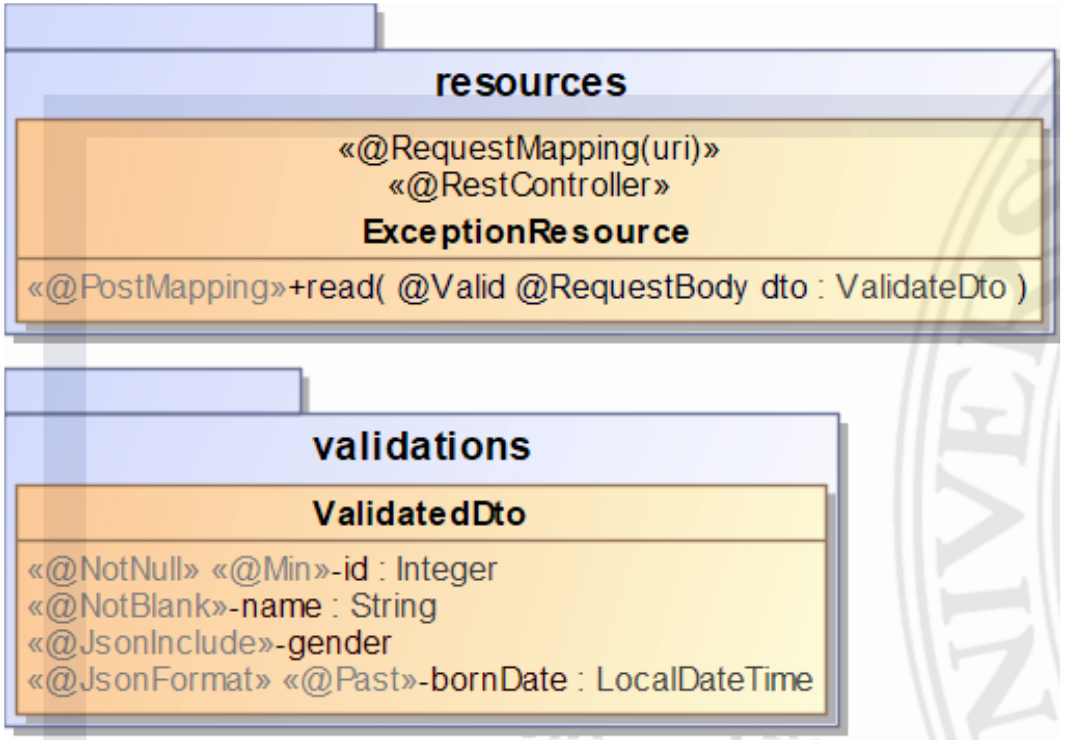
✎ Rest en práctica

- Basic Resource
- Añadir end-point PATCH /basic, recibe en el cuerpo *[{id,name}]* para actualizar

Spring
Excepciones



Spring
Validaciones



Validador propio: *PositiveBigDecimal*

Validadores

- **@DecimalMax, @DecimalMin**
- **@Digits**
- **@Email**
- **@Future, @FutureOrPresent**
- **@Max, @Min**
- **@Negative, @NegativeOrZero**
- **@Null, @NotNull**
- **@NotEmpty, @NotBlank**
- **@Past, @PastOrPresent**
- **@Positive, @PositiveOrZero**
- **@Pattern**
- **@Size**

Exceptions. betca-spring ✎ {→}

https://github.com/miw-upm/betca-spring

- **betca-rest**

✎ Rest en práctica

- *ExceptionResource, ValidatedDto*
- Añadir…

Spring
Seguridad

Rol

- Es un nombre abstracto para configurar el permiso de acceso de un conjunto de recursos de una aplicación.

Usuario

- Es una identidad única reconocida por el servidor. Un usuario puede tener varios roles.

Recursos

- Se controla las acciones de diferentes roles.
- Distribuido: anotaciones en el recurso.
- Centralizado: Uri's.

Autenticación

- Memoria
- Propio
- API Key o Token
- LDAP
- Auth2
- OpenIp
- …

Spring
Seguridad

AuthenticationWebFilter

- Es un filtro de seguridad, que se procesa y se decide si se continua.
- Depende del *AuthenticationManager* para procesar.

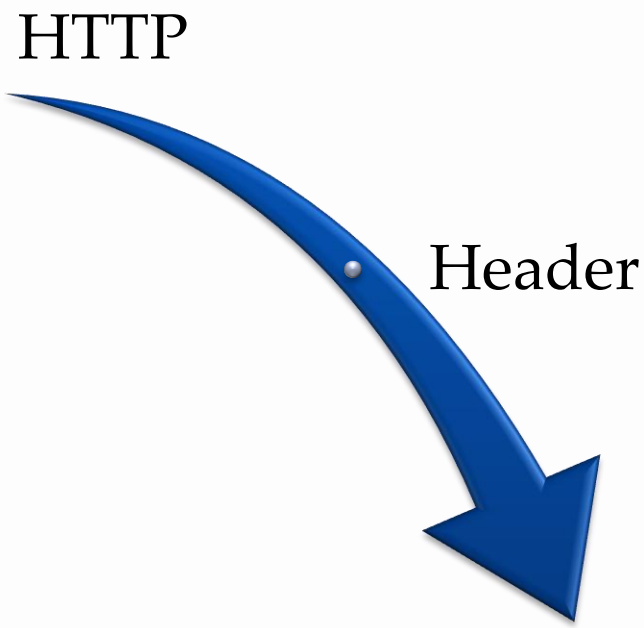
AuthenticationManager

- Aquí se procesa si los datos de autenticación son correctos, dependiendo del tipo de autenticación (HttpBasic, Bearer…)
- Depende del *AuthenticationConverter* para extraer los datos de la cabecera de la petición.

AuthenticationConverter

- Extrae de la cabecera los datos de la autenticación y los prepara para ser procesados.

Seguridad. Spring
Basic Auth

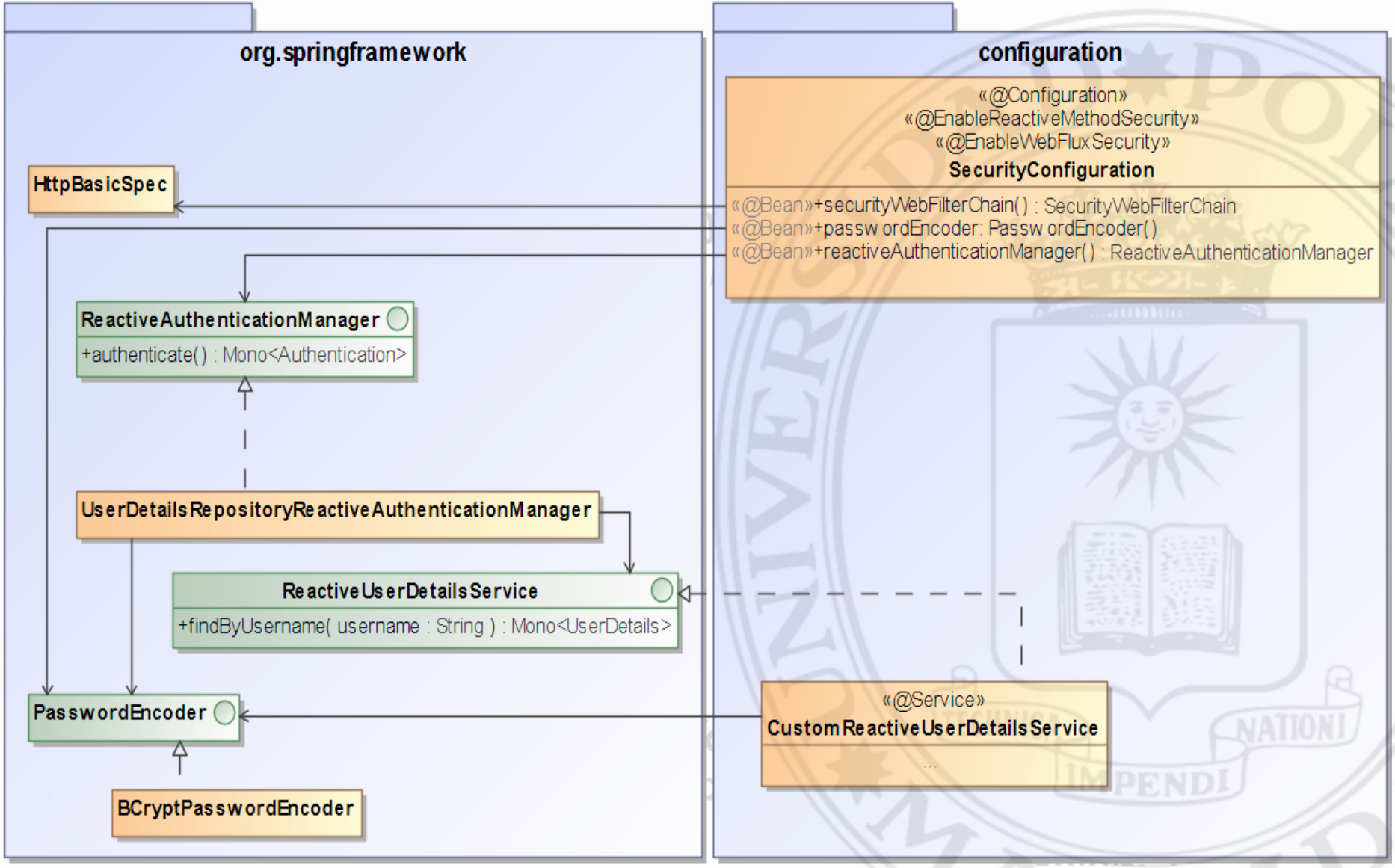


Authorization=Basic (*user:password*)$_{\textbf{code64}}$

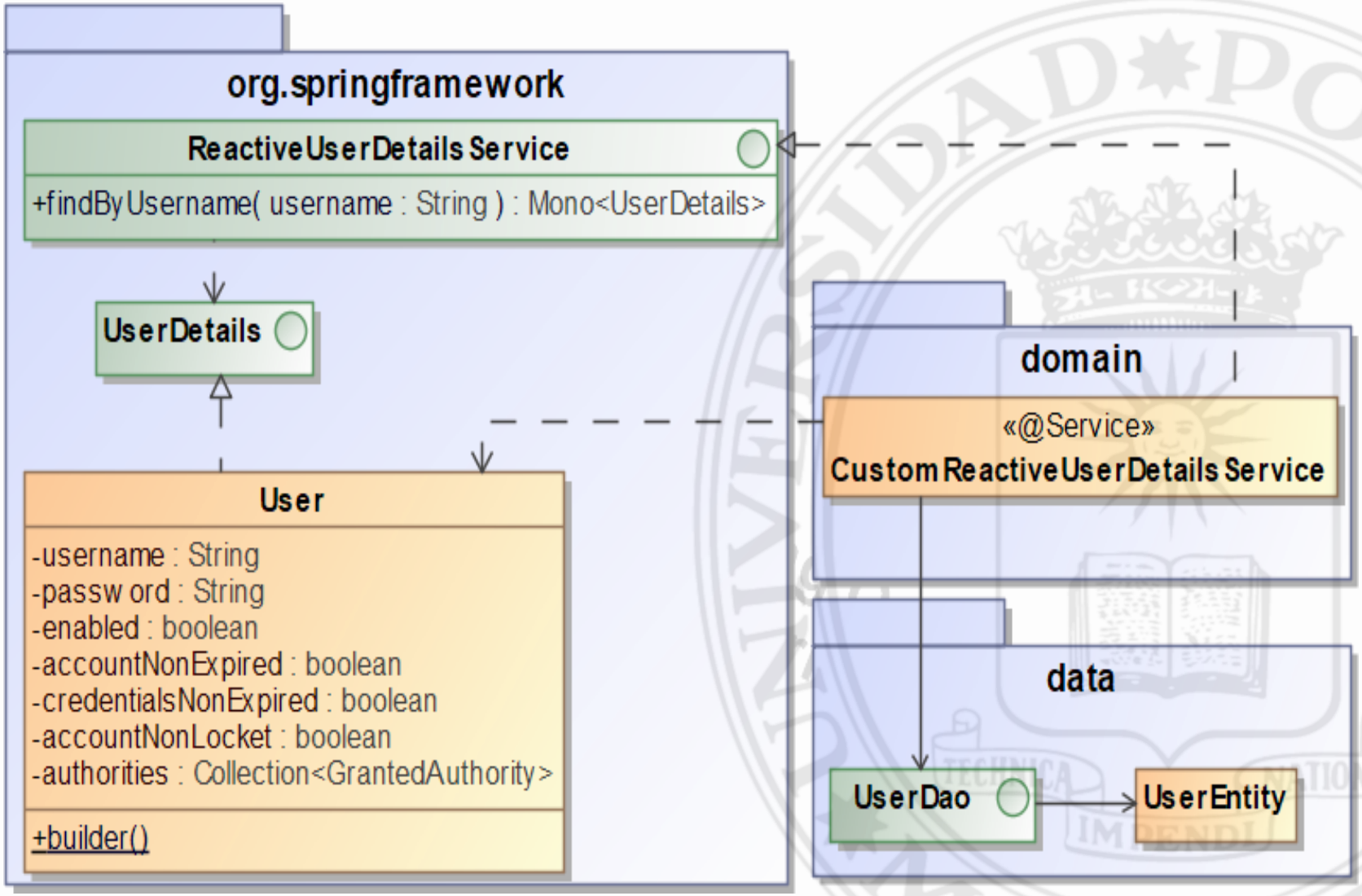
Configuración de Spring

- ***Bean*** de la cadena de filtros: ***SecurityWebFilterChain***
  - Filtro *Basic Auth*: ***.httpBasic()***
- ***Bean*** de encriptación de claves: ***PasswordEncoder***
  - *BCryptPasswordEncoder*
- ***Bean*** del manejador de seguridad: ***ReactiveAuthenticationManager***
  - *CustomReactiveUserDetailsService*

Seguridad. Spring
Basic Auth



Seguridad. Spring
Basic Auth



Security. betca-spring

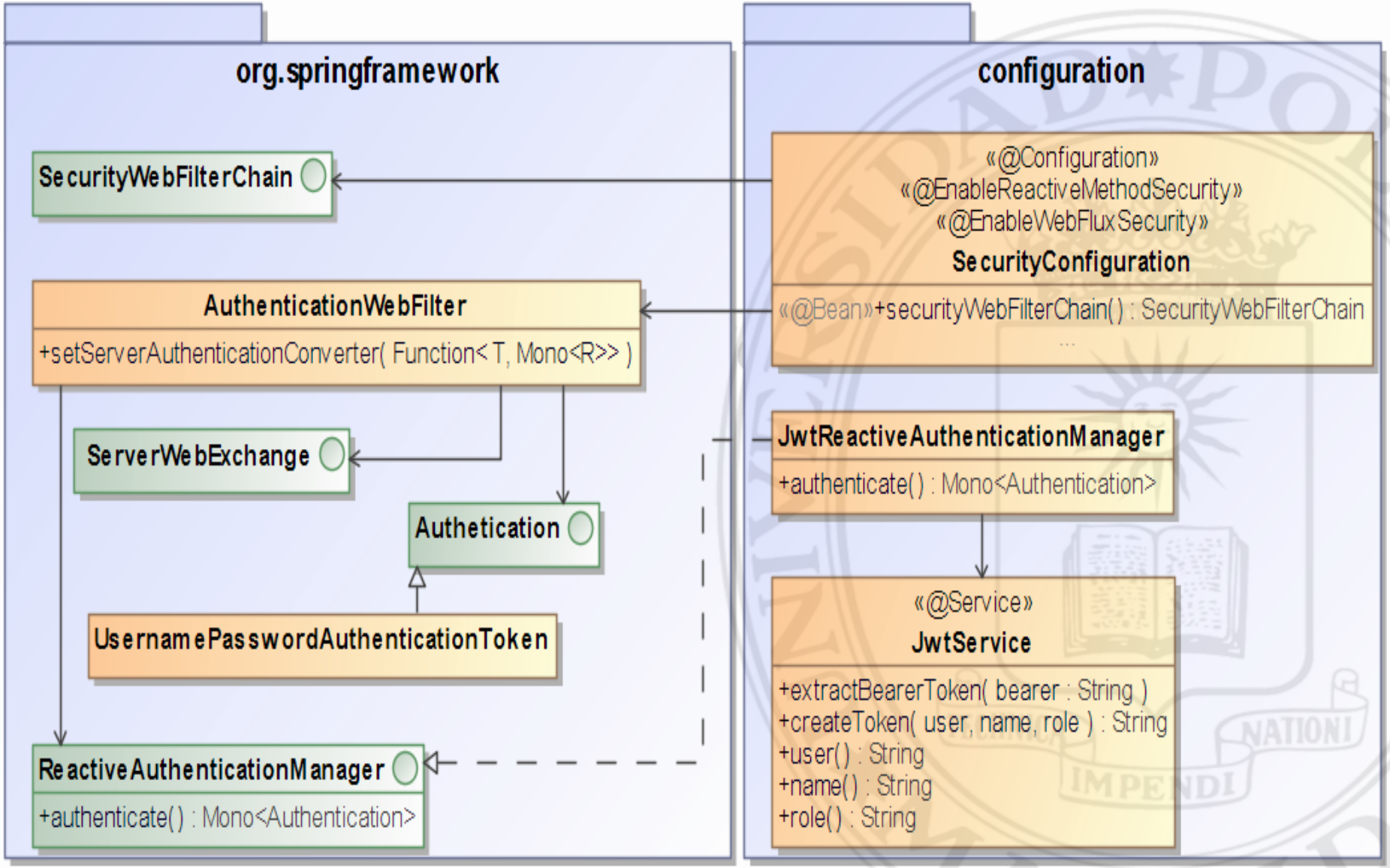
https://github.com/miw-upm/betca-spring

- **betca-rest**

Rest en práctica

- *SecurityResource, CustomReactiveUserDetailsService*
- Crear el end-point DELETE */basic-auth/{ID}*, sólo podrá borrar el usuario "*u1*" con el rol de "*PLAYER*", se configura con anotaciones.

Seguridad. Spring
JWT



Security. betca-spring

✎ {→}

https://github.com/miw-upm/betca-spring

- **betca-rest**

✎ Rest en práctica

- *JwtReactiveAuthenticationManager, JwtService*
- Añadir…

Servicio email

pom.xml

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

application.properties

```
# Email ########################################
# Se debe configurar la cuenta de gmail para soportar la conexion de la aplicacion
# Login to Gmail
# Access the URL as https://www.google.com/settings/security/lesssecureapps
# Select "Turn on"
spring.mail.defaultEncoding=UTF-8
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=${MAIL_USER}
spring.mail.password=${MAIL_PASS}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.test-connection=false
```

test.properties

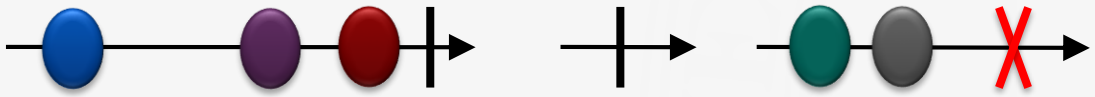
```
# Mail
spring.mail.username=notUser
spring.mail.password=notPass
```

Servicio email

```
@TestConfig
public class MailIT {

    @MockBean
    private MailSender mailSender;

    @Autowired
    private MailService mailService;

    @Test
    public void testSendMessage() {
        final ArgumentCaptor<SimpleMailMessage> captor =
         ArgumentCaptor.forClass(SimpleMailMessage.class);
        this.mailService.send("test@gmail.com", "test message");
        verify(mailSender).send(captor.capture());
        assertEquals(mailService.getFrom(), captor.getValue().getFrom());
        assertEquals("test@gmail.com", captor.getValue().getTo()[0]);
        assertEquals(mailService.getSubject(), captor.getValue().getSubject());
        assertEquals("test message", captor.getValue().getText());
    }
}
```

Reactive Programming

- Flujo de datos asíncronos

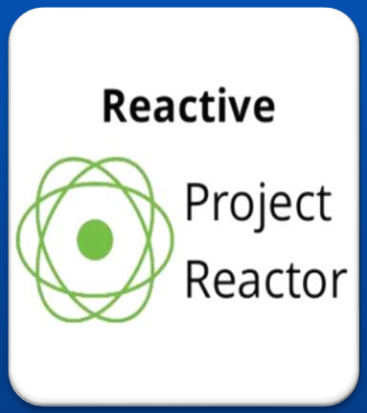


- Programación Funcional
- Programación Asíncrona

Spring. Reactive
Proyectos

RxJS - Javascript

- Angular
- Clase: *Observable*
- https://*rxjs.dev/guide/overview*

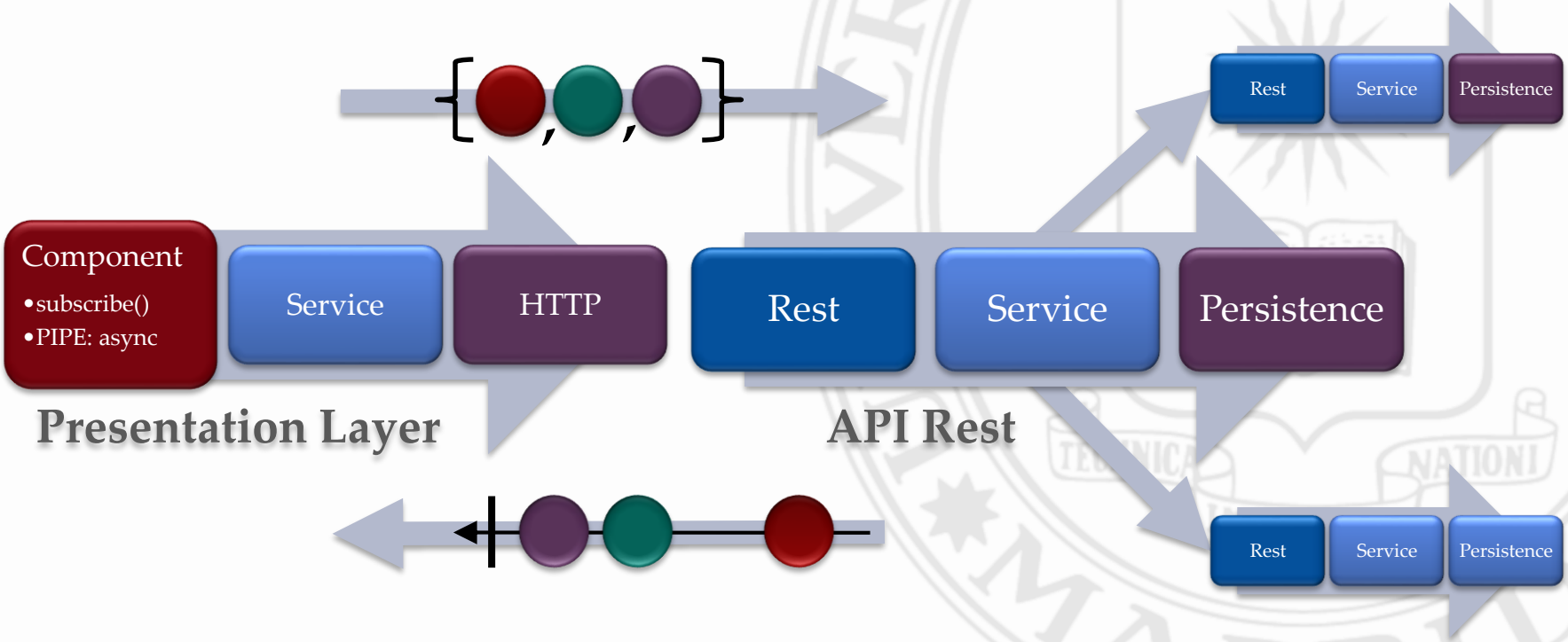


Project-reactor - Java

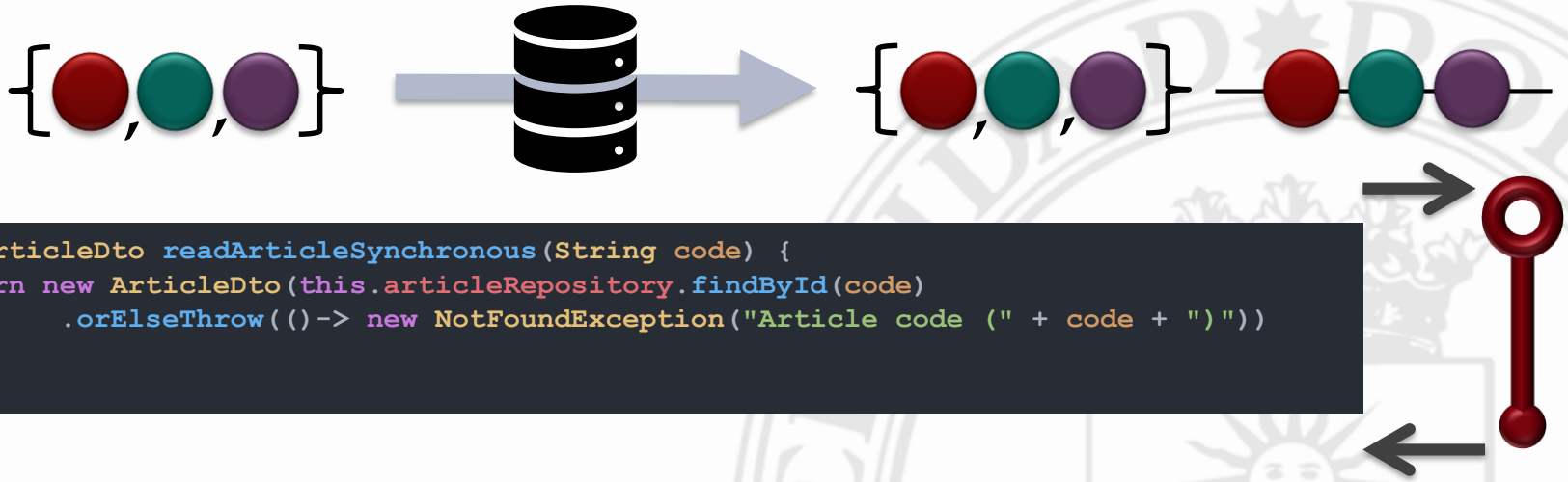
- Java - Spring
- Clases: *Mono & Flux*
- *https://projectreactor.io/docs/core/release/reference*
- *https://projectreactor.io/docs/core/release/api*

Spring Reactive
Flujo

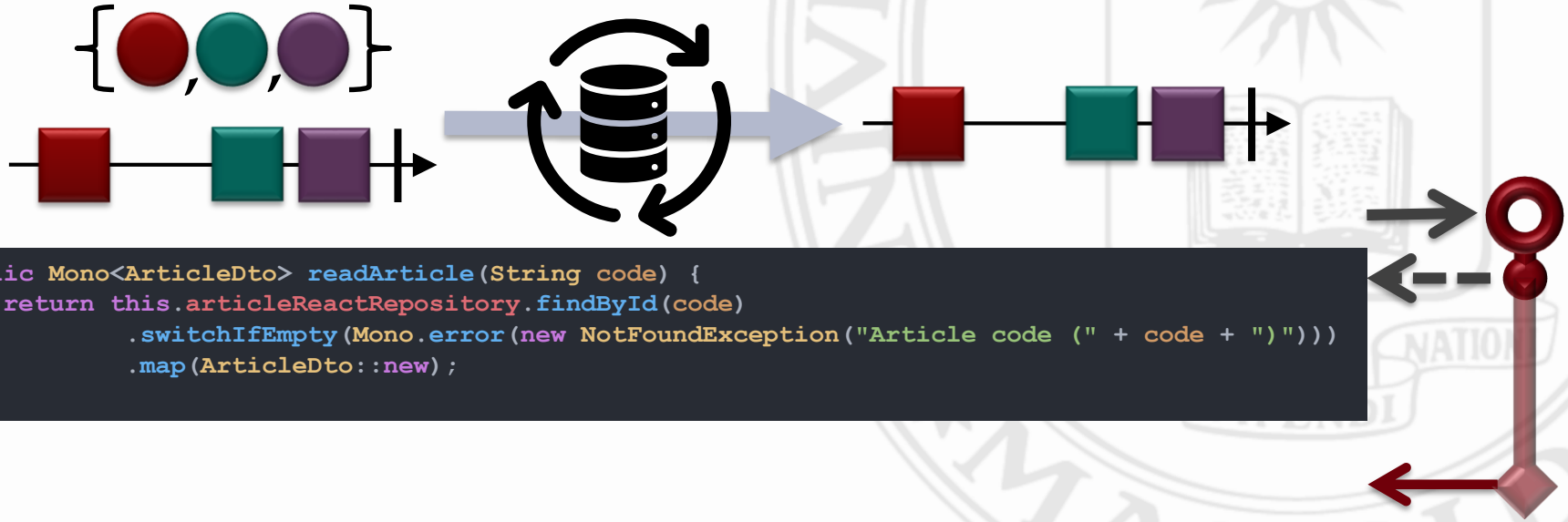
Métodos que devuelven un tipo *Publisher* no *deberían* subscribirse porque ello *podría romper* la cadena del publicador



Spring. Reactive
BD



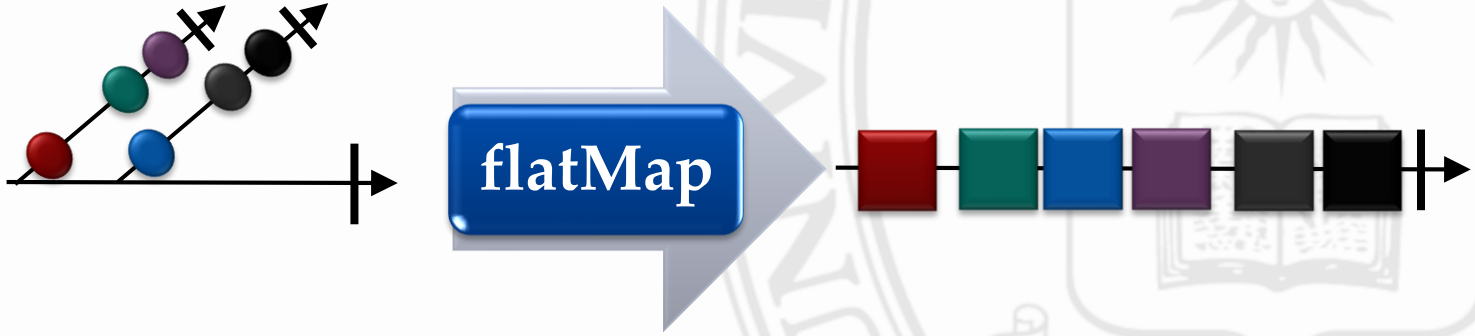
```
public ArticleDto readArticleSynchronous(String code) {
    return new ArticleDto(this.articleRepository.findById(code)
            .orElseThrow(()-> new NotFoundException("Article code (" + code + ")"))
    );
}
```



```
public Mono<ArticleDto> readArticle(String code) {
    return this.articleReactRepository.findById(code)
            .switchIfEmpty(Mono.error(new NotFoundException("Article code (" + code + ")")))
            .map(ArticleDto::new);
}
```

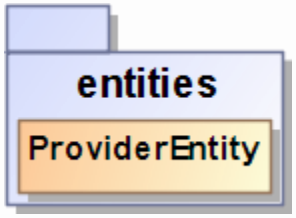
Programación Reactiva
Funciones


```
public Flux<Integer> convertToInteger(Flux<String> flux) {
    return flux.map(Integer::valueOf);
}
```




```
public Flux<Integer> convertToIntegerFlatMap(Flux<String> flux) {
    return flux.flatMap(string -> Mono.just(Integer.valueOf(string)));
}
```

Spring. Reactive
BD



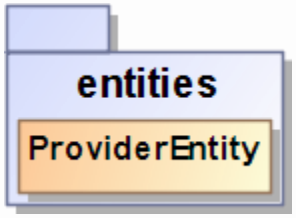
Reactive

- *map:* transforma un objeto en otro
- *flatMap:* aplana un flujo en objetos
- *switchIfEmpty*: cambia un flujo vació por otro flujo

```
@Override
public Mono< Provider > readByCompany(String company) {
    return this.providerReactive.findByCompany(company)
            .switchIfEmpty(Mono.error(new NotFoundException("Non existent company: " + company)))
            .map(ProviderEntity::toProvider);
}
```

```
@Override
public Flux< Provider > findByCompanyAndPhoneAndNoteNullSafe(String company, String phone, String note) {
    return this.providerReactive.findByCompanyAndPhoneAndNoteNullSafe(company, phone, note)
            .map(ProviderEntity::toProvider);
}
```

Spring. Reactive
BD



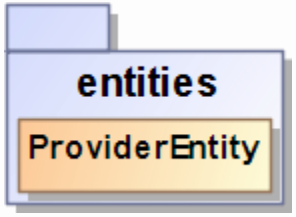
Reactive

- *map:* transforma un objeto en otro
- *flatMap:* aplana un flujo en objetos
- *switchIfEmpty*: cambia un flujo vació por otro flujo
- *mergeWith*: fusiona un flujo con otro
- *then()*: retorna un *Mono<Void>* cuando el flujo se completa
- *then(Mono<V>)*: retorna el *Mono* cuando el flujo se completa

```
private Mono< Void > assertCompanyAndNifNotExist(String company, String nif) {
    return this.providerReactive.findByCompany(company) // Mono<ProviderCompany>
            .mergeWith(this.providerReactive.findByNif(nif)) // Flux<ProviderCompany>
            .flatMap(provider -> Mono.error(new ConflictException
                    ("Existing already company or nif : " + company + "," + nif)
            )) // Flux<Object>
            .then(); // Mono<Void>
}
```

```
@Override
public Mono< Provider > create(Provider provider) {
    return this.assertCompanyAndNifNotExist(provider.getCompany(), provider.getNif()) // Mono<Void>
            .then(this.providerReactive.save(new ProviderEntity(provider))) // Mono<ProviderEntity>
            .map(ProviderEntity::toProvider); // Mono<Provider>
}
```

Spring. Reactive
BD

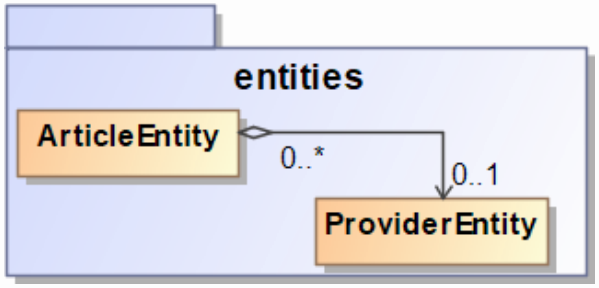


Reactive

- *map:* transforma un objeto en otro
- *flatMap:* aplana un flujo en objetos
- *switchIfEmpty*: cambia un flujo vació por otro flujo
- *mergeWith*: fusiona un flujo con otro
- *then()*: retorna un *Mono<Void>* cuando el flujo se completa
- *then(Mono<V>)*: retorna el *Mono* cuando el flujo se completa

```
@Override
public Mono< Provider > update(String company, Provider provider) {
    return this.providerReactive.findByCompany(company)
            .switchIfEmpty(Mono.error(new NotFoundException("Non existent company: " + company)))
            .flatMap(providerEntity -> {
                BeanUtils.copyProperties(provider, providerEntity);
                return this.providerReactive.save(providerEntity);
            })
            .map(ProviderEntity::toProvider);
}
```

Spring. Reactive
BD



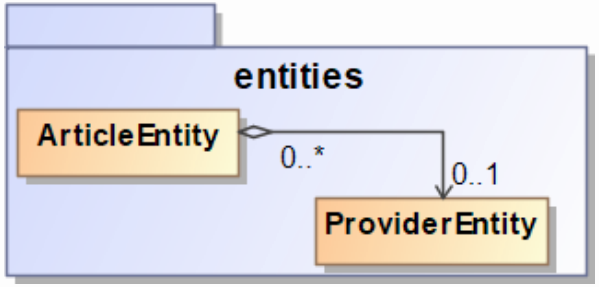
Reactive

- *map:* transforma un objeto en otro
- *flatMap:* aplana un flujo en objetos
- *switchIfEmpty*: cambia un flujo vació por otro flujo
- *mergeWith*: fusiona un flujo con otro
- *then()*: retorna un *Mono<Void>* cuando el flujo se completa
- *then(Mono<V>)*: retorna el *Mono* cuando el flujo se completa

```
@Override
public Mono< Article > readByBarcode(String barcode) {
    return this.articleReactive.findByBarcode(barcode)
            .switchIfEmpty(Mono.error(new NotFoundException("Non existent article barcode: " + barcode)))
            .map(ArticleEntity::toArticle);
}
```

```
@Override
public Flux< String > findBarcodeByBarcodeLikeAndNotDiscontinuedNullField(String barcode) {
    return this.articleReactive.findByBarcodeLikeAndNotDiscontinuedNullSafe(barcode)
            .map(ArticleEntity::getBarcode);
}
```

Spring. Reactive
BD



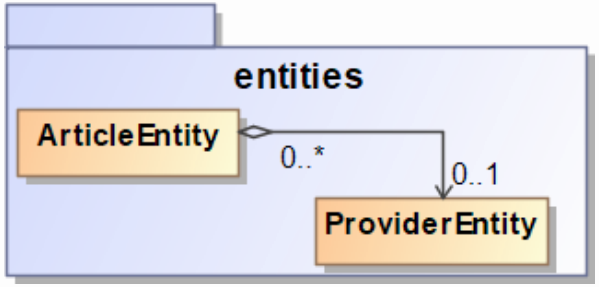
Create Article

- *Barcode* debe ser único
- *Provider* puede ser null, pero si no es null, debe existir
- *justOrEmpty*: crea un *Mono<V>*, si es null, crea un *Mono<Void>*

```
private Mono< Void > assertBarcodeNotExist(String barcode) {
    return this.articleReactive.findByBarcode(barcode)
            .flatMap(articleEntity -> Mono.error(
                    new ConflictException("Article Barcode already exists : " + barcode)
            ));
}
```

```
@Override
public Mono< Article > create(Article article) {
    return this.assertBarcodeNotExist(article.getBarcode())
            .then(Mono.justOrEmpty(article.getProviderCompany()))
            .flatMap(providerCompany -> this.providerReactive.findByCompany(article.getProviderCompany())
                    .switchIfEmpty(Mono.error(
                            new NotFoundException("Non existent company: " + article.getProviderCompany())
                    ))
            )
            .map(providerEntity -> new ArticleEntity(article, providerEntity))
            .switchIfEmpty(Mono.just(new ArticleEntity(article, null)))
            .flatMap(this.articleReactive::save)
            .map(ArticleEntity::toArticle);
}
```

Spring. Reactive
BD

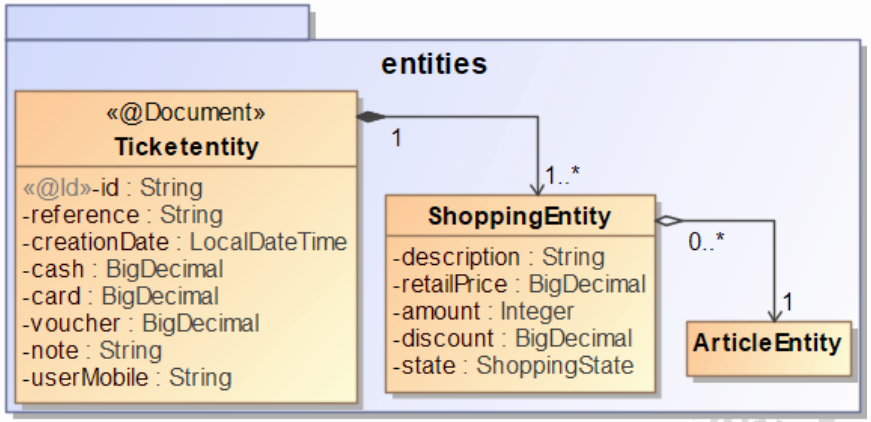


Reactive

- *map:* transforma un objeto en otro
- *flatMap:* aplana un flujo en objetos
- *switchIfEmpty*: cambia un flujo vació por otro flujo
- *mergeWith*: fusiona un flujo con otro
- *then()*: retorna un *Mono<Void>* cuando el flujo se completa
- *then(Mono<V>)*: retorna el *Mono* cuando el flujo se completa
- *justOrEmpty*: crea un *Mono<V>*, si es null, crea un *Mono<Void>*

```
@Override
public Mono< Article > update(String barcode, Article article) {
    Mono< ArticleEntity > articleEntityMono;
    if (!barcode.equals(article.getBarcode())) {
        articleEntityMono = this.assertBarcodeNotExist(article.getBarcode())
                .then(this.articleReactive.findByBarcode(barcode));
    } else {
        articleEntityMono = this.articleReactive.findByBarcode(barcode);
    }
    return articleEntityMono
            .switchIfEmpty(Mono.error(new NotFoundException("Non existent article barcode: " + barcode)))
            .flatMap(articleEntity -> {
                BeanUtils.copyProperties(article, articleEntity);
                return this.providerReactive.findByCompany(article.getProviderCompany())
                        .switchIfEmpty(Mono.error(
                                new NotFoundException("Non existent company: " + article.getProviderCompany()))
                        )
                        .map(providerEntity -> {
                            articleEntity.setProviderEntity(providerEntity);
                            return articleEntity;
                        });
            })
            .flatMap(this.articleReactive::save)
            .map(ArticleEntity::toArticle);
}
```

Spring. Reactive
BD



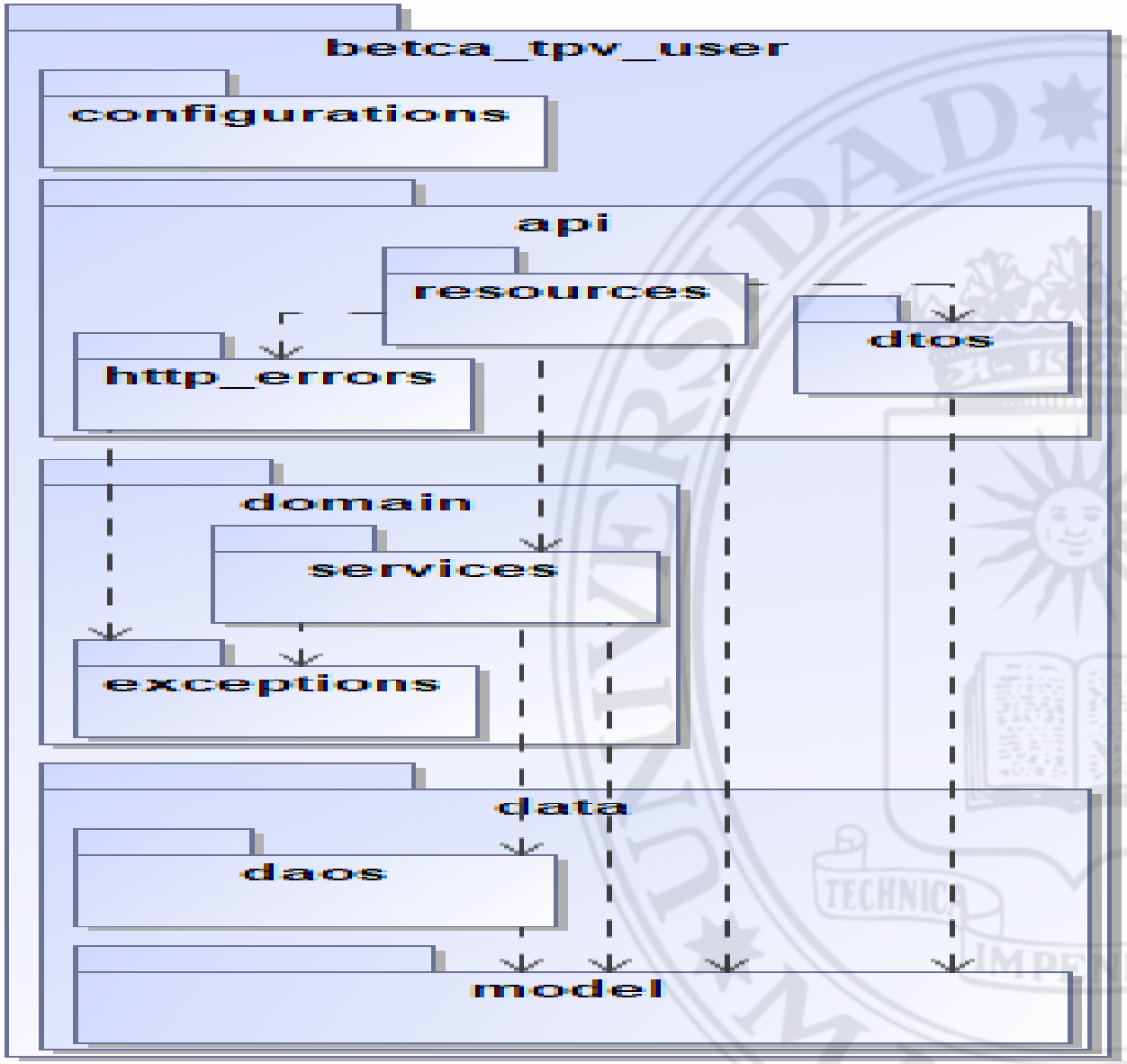
Create Ticket

- Por cada *shopping* se debe mirar que exista el artículo, y obtener el *precio* y *descripción* del artículo
- *doOnNext*: añade comportamiento por cada item

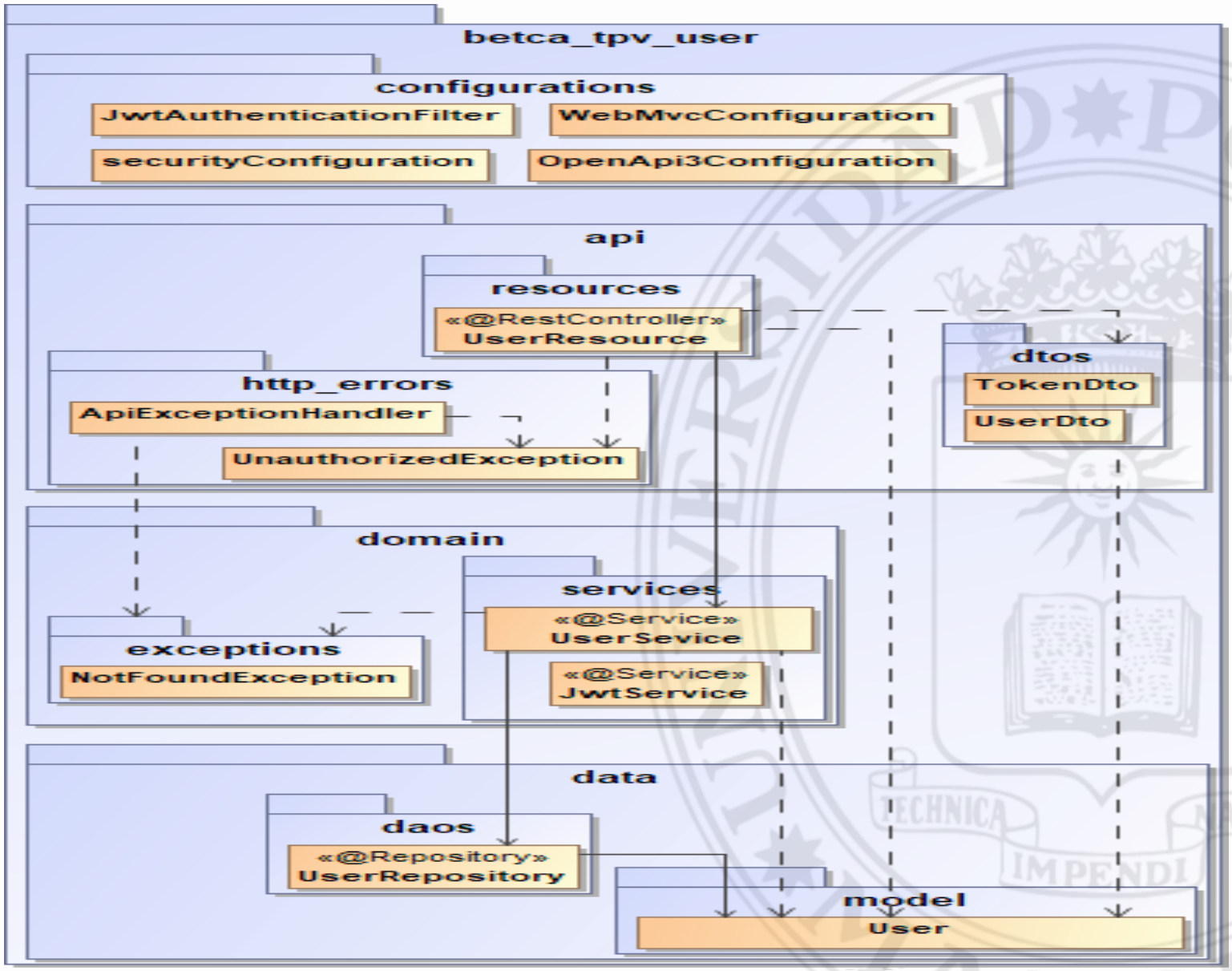
```
@Override
public Mono< Ticket > readById(String id) {
    return this.ticketReactive.findById(id)
            .map(TicketEntity::toTicket);
}
```

```
@Override
public Mono< Ticket > create(Ticket ticket) {
    TicketEntity ticketEntity = new TicketEntity(ticket);
    return Flux.fromStream(ticket.getShoppingList().stream())
            .flatMap(shopping -> {
                ShoppingEntity shoppingEntity = new ShoppingEntity(shopping);
                return this.articleReactive.findByBarcode(shopping.getBarcode())
                        .switchIfEmpty(Mono.error(new NotFoundException("Article: " + shopping.getBarcode())))
                        .map(articleEntity -> {
                            shoppingEntity.setArticleEntity(articleEntity);
                            shoppingEntity.setDescription(articleEntity.getDescription());
                            return shoppingEntity;
                        });
            }).doOnNext(ticketEntity::add)
            .then(this.ticketReactive.save(ticketEntity))
            .map(TicketEntity::toTicket);
}
```

Spring. Reactive
BD

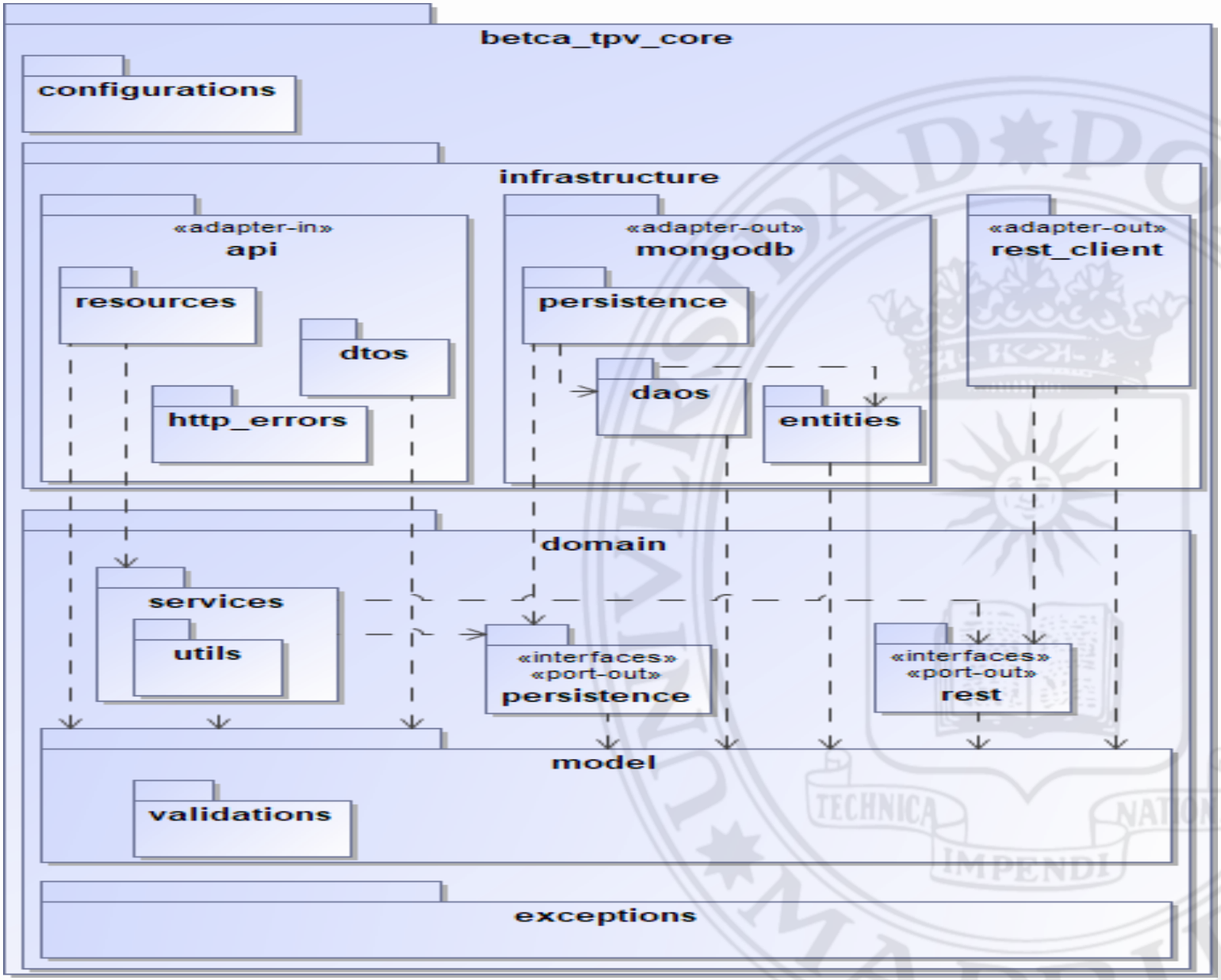


Spring. Reactive
BD

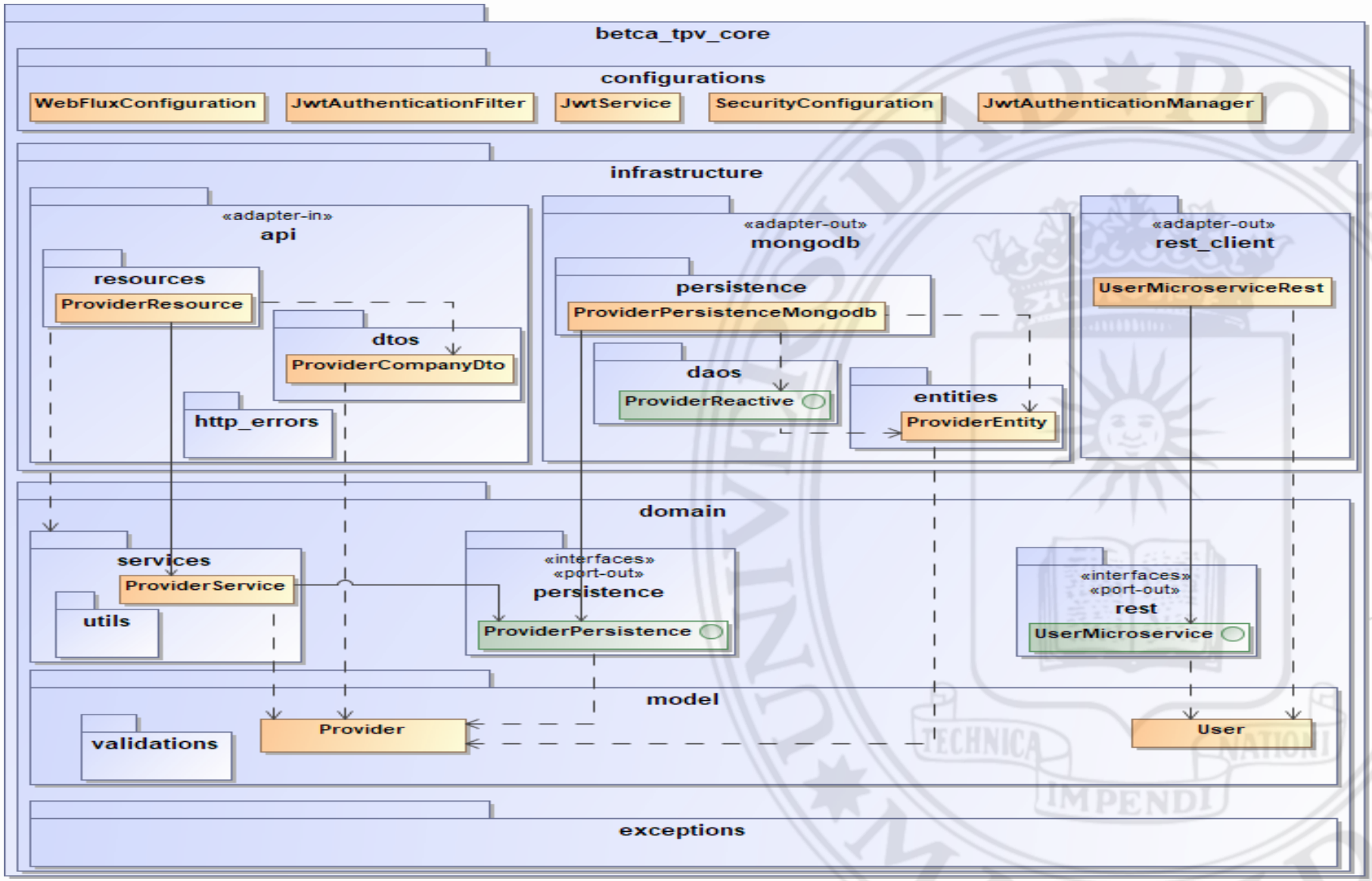


Spring. Reactive

BD



Spring. Reactive
BD



Python

https://www.python.org/

Python es un lenguaje de programación de propósito general.

Esta desarrollado bajo *open-source*.

Fue creado por *Guido van Rossum* a final de los 80

Python
Características

Fuente

- Alto nivel: muy legible
- Propósito general
- Interpretado
- Multiplataforma
- Tipado
  - Estático
  - Dinámico

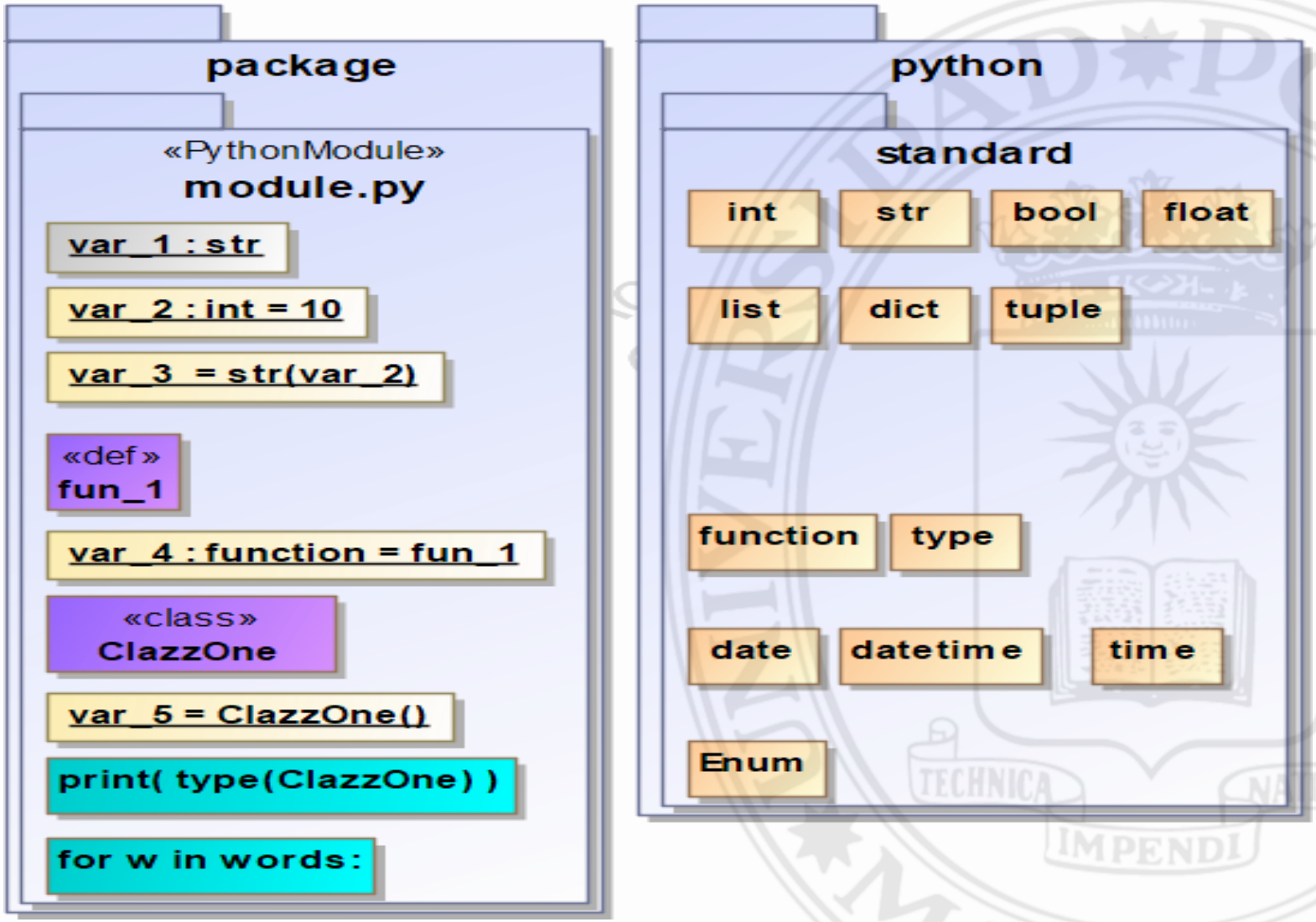
Multiparadigma

- Programación Funcional
- Programación Imperativa
- Programación Orientada a Objetos
- Programación Reflexiva
- Programación Asíncrona

Dependencias

- PIP: gestor de paquetes de Python
- Fichero *requirements.txt:* fichero con las dependencias
- PyPI
  - *https://pypi.org*
  - Repositorio de código

Python
Conceptos



Python
Instalación

Instalar

- Python: *https://www.python.org/downloads/*
- PyCharm: *https://www.jetbrains.com/pycharm/*
- Crear un proyecto con entorno virtual:
  - *py -m venv <project-name\venv>*
    - venv\Scripts\activate.bat
    - venv\Scripts\deactivate.bat
- Abrir con PyCharm
  - Crear fichero: hello_world.py
  - Añadir: print('Hello World')
  - Ejecutar

Consola

- Python --version
- py -V # py==python
- py -h # Help
- py -0 # available pythons
- pip list
- pip --version
- py -m pip --version # run pip library module as a script
- pip show pip # show pip version
- pip install -U <module> # update
- pip install -U pip # -U == --upgrade
- py -m pip unistall <module>
- pip install -r requirements.txt

Python. Lenguaje

https://github.com/miw-upm/betca-Python

- **1_hello_world.py**
- **2_variables.py**
- **3_operators.py**
- **4_flows.py**
- **5_dicts.py**
- **6_functions.py**
- **7_clases.py**

Clonar el proyecto

- *Ejecutar, modificar, aprender…*

Python
Tests

unittest

- *Python standar library*
- Heredar de *TestCase*
- Similar a *JUnit*
- Consola: >**py -m unittest discover *tests***

Ciclo de vida

- *@classmethod def setUpClass: antes de todos los tests*
- def setUpself: antes de cada test
- **def test_1: un test**
- def tearDownself: después de cada test
- *@classmethod def tearDownClass: después de todos los tests*

Asserts

- self.assertEqual, self.assertNotEqual, self.assertTrue, self.assertFalse, self.assertIs (same object), self.assertIsNot, self.assertIsNone, self.assertIsNotNone, self.assertIn, self.assertNotIn
- with self.assertRaises(ZeroDivisionError):
- self.assertAlmostEqual, self.assertNotAlmostEqual
- self.assertGreater, self.assertGreaterEqual, self.assertLess, self.assertLessEqual
- self.assertRegex, self.assertNotRegex

Otros …

- pytest (★7k): *https://docs.pytest.org*

Python
Tests. Mocks

```
def helper_method():
    print('--- helper:', method(666))
```

```
def method(arg):
    print('>>> method... return:', arg)
    return arg
```

```
class TestDemo(TestCase):
    @mock.patch('tests.helper.method', return_value=0)
    def test_two_a(self, mocked_method):
        helper_method()
        mocked_method.assert_called
```

```
class TestDemo(TestCase):
    @mock.patch('tests.helper.method', side_effect=fake_method)
    def test_two_b(self, mocked_method):
        helper_method()
        mocked_method.assert_called_with(666)
```

```
def fake_method(arg):
    print('*** mock... arg:', arg)
    return -111
```

Python
Tests. Mocks

```python
class HelperClass:
    def helper(self):
        print('helper:', Clazz(111).class_method(666))
```

```python
class Clazz:
    attribute: int
    def __init__(self, attribute):
        self.attribute = attribute
    def class_method(self, arg: int):
        print('>>> Clazz::method... arg:', arg)
        return self.attribute + arg
```

```python
class TestDemo(TestCase):
    @mock.patch('tests.actual.Clazz.class_method', return_value=0)
    def test_five_a(self, mocked_method):
        HelperClass().helper()
        mocked_method.assert_called_once()
```

```python
class TestDemo(TestCase):
    @mock.patch('tests.actual.Clazz.class_method', side_effect=fake_method)
    def test_five_b(self, method):
        HelperClass().helper()
        method.assert_called_with(666)
```

```python
def fake_method(arg):
    print('*** mock... arg:', arg)
    return -111
```

Python. Tests

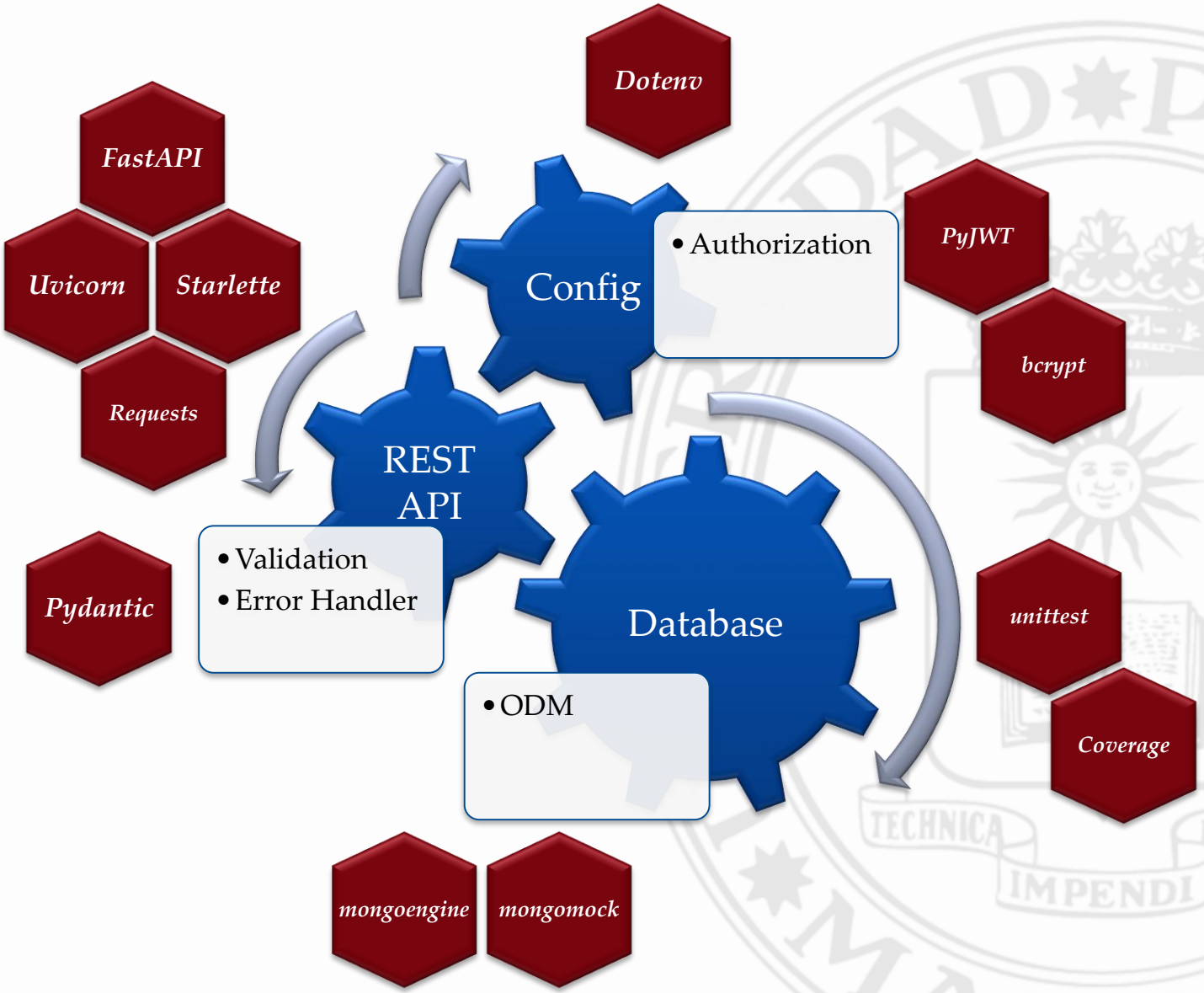
https://github.com/miw-upm/betca-Python

- **test_unittest.py**
- **actual.py**
- **helper.py**
- **test_mock.py**

tests

- *Ejecutar, modificar, aprender…*

Python
Web


Dotenv
FastAPI
Uvicorn
Starlette
Requests
Config
• Authorization
PyJWT
bcrypt
REST API
• Validation
• Error Handler
Pydantic
Database
• ODM
unittest
Coverage
mongoengine
mongomock

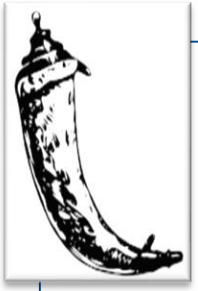
Python
REST

Django

- https://www.djangoproject.com/
- GitHub: ★55k
- Framework Web creado en 2003 y basado en MVC. Soporta ASGI (Asynchronous Server Gateway Interface)

Django REST Framework

- https://www.django-rest-framework.org/
- GitHub: ★20k



Flask

- https://flask.palletsprojects.com
- GitHub: ★54k
- Microframework basado en WSGI (Web Server Gateway Inteface) y Jinja

Flask-RESTful

- https://flask-restful.readthedocs.io
- GitHub: ★6k

FastAPI

- *https://fastapi.tiangolo.com/*
- GitHub: ★26k
- Microframework, moderno y asíncrono.
- Basado en Starlette (uvicorn: ASGI) & Pydantic

Otros

- Tornado: ★20k
- Bottle: ★7k
- Pyramid: ★4k
- …

Python
REST: FastAPI

FastAPI. *https://fastapi.tiangolo.com*

- Instalación:
  - pip install fastapi
  - pip install uvicorn # ASGI server
- Ejecución:
  - uvicorn rest.main:app --reload --port 8085
  - *http://localhost:8000/docs* (openAPI)
- Dependencias:
  - ***Starlette*** (Web). Es un framework/librería muy ligera ideal para construir servicios asíncronos: ASGI (Asynchronous Server Gateway Interface). *https://www.starlette.io/*
  - ***Pydantic*** (data). Validación de datos y gestion de configuración. *https://pydantic-docs.helpmanual.io/*
  - ***Uvicorn*** – Servidor Web ligero y rápido que implementa ASGI. *https://www.uvicorn.org/*

Python
REST. FastAPI

```
app = FastAPI(title='BETCA-Python')
app.include_router(basic)
```

```
from pydantic import BaseModel

class Dto(BaseModel):
    id: str
    name: str
    description: str
    value: Optional[int]
```

```
> uvicorn rest.main:app –reload –port 8085
```

```
basic = APIRouter(
    prefix="/basic",
    tags=["basic"],  # OpenAPI
)

@basic.get("/search")
def find(name: str) -> [Dto]:
    return [Dto(id='1', name=name, description='desc'),
            Dto(id='2', name=name, description='desc2', value=2)]

@basic.post("")
def create(dto: Dto) -> Dto:
    return dto

@basic.get("/{ide}")
def read(ide: str) -> Dto:
    return Dto(id=ide, name="dto", description='desc')

@basic.put("/{ide}")
def update(ide: str, dto: Dto) -> Dto:
    dto.id = ide
    return dto

@basic.delete("/{ide}")
def delete(ide: str):
    print('Dto(', ide, ') deleted')
```

PUT /basic/666 {dto}

GET /basic/search?name=Miw

Python
REST. FastAPI. Tests

```
client = TestClient(app)


class TestService(TestCase):
    def test_read_basic(self):
        response = client.get("/basic/666")
        self.assertEqual(HTTPStatus.OK, response.status_code)
        self.assertEqual('666', response.json()['id'])
        self.assertEqual({'id': '666', 'name': 'dto', 'description': 'desc', 'value': None}, response.json())
        print('response: ', response.json())

    def test_create(self):
        response = client.post("/basic", json=Dto(id='1', name="dto", description='desc').dict())
        self.assertEqual(HTTPStatus.OK, response.status_code)
        print(response.json())

    def test_update(self):
        response = client.put("/basic/666", json={'id': '666', 'name': 'dto', 'description': 'desc'})
        self.assertEqual(HTTPStatus.OK, response.status_code)
        print(response.json())

    def test_delete(self):
        response = client.delete("/basic/666")
        self.assertEqual(HTTPStatus.OK, response.status_code)
```

Python
Pydantic: Validaciones

```python
class FruitEnum(str, Enum):
    pear = 'pear'
    banana = 'banana'

class ValidationDto(BaseModel):
    id: constr(min_length=1, max_length=10)
    name: constr(min_length=4, strip_whitespace=True)
    description:  constr(regex=r'^miw-(betca|spring|python)$')
    value: Optional[conint(multiple_of=5)]
    adult: StrictBool
    color: Color
    fruit: FruitEnum
```

https://pydantic-docs.helpmanual.io

Constrained Types:

- NegativeFloat, NegativeInt, PositiveFloat, PositiveInt, conbytes, condecimal, confloat, conint, conlist, conset, constr

Strict Types

- StrictStr, StrictInt, StrictFloat, and StrictBool

Color Type, Datetime Types…

- 'black', '7fffd4', (255, 255, 255, 0.5), 'rgb(255, 255, 255)', 'hsl(270, 60%, 70%)'

Rest. betca-python

https://github.com/miw-upm/betca-python

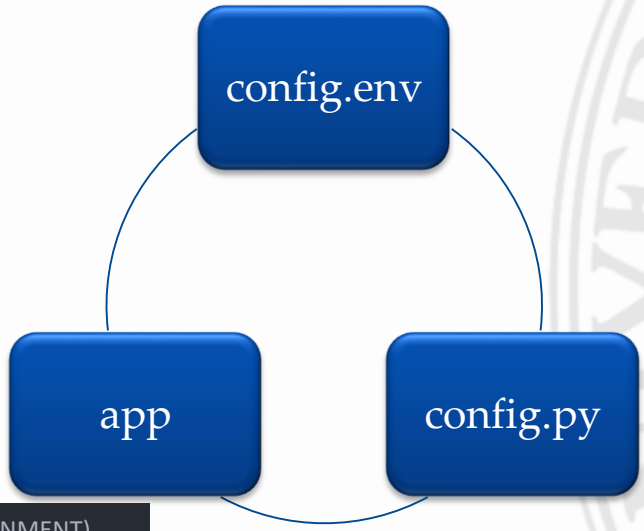
- **betca-rest**

Rest en práctica

- *basic_resource.py*
- Añadir end-point POST /basic/calc, recibe en el cuerpo *{name,value}* y hace un eco
- Añadir end-point PUT /basic/calc, recibe en el cuerpo *{id,name,value}* y hace un eco
- Añadir end-point PATCH /basic, recibe en el cuerpo *[{id,name}]* para actualizar
- *validation_resource.py*

Python
Pydantic: Configuración

```
ENVIRONMENT=prod

DEV_VAR=value-on-dev
DEV_JWT_SECRET=secret-to-test
DEV_*

PROD_VAR=value-on-production
PROD_JWT_SECRET=secret-${USERNAME}
DEV_*
```




```
print(config.ENVIRONMENT)
print(config.VAR)
print(config.JWT_SECRET)
```

```
class Settings(BaseSettings):
    ENVIRONMENT: str = Field(None, env="ENVIRONMENT")
    VAR: Optional[str]
    JWT_SECRET: Optional[str]

    class Config:
        env_file: str = "../config.env"


class DevSettings(Settings):
    class Config:
        env_prefix: str = "DEV_"


class ProdSettings(Settings):
    class Config:
        env_prefix: str = "PROD_"


def get_settings():
    if "prod" == Settings().ENVIRONMENT:
        return ProdSettings()
    else:
        return DevSettings()


config = get_settings()
```

Python
Seguridad. Auth Basic

```python
def test_read_auth_ok(self):
    response = client.get("/auth/basic", auth=('jes', 'pass'))
    self.assertEqual(HTTPStatus.OK, response.status_code)
```

pyjwt: ★4k

bcrypt: ★1k

cryptography: ★4k
pycrypto: ★2k

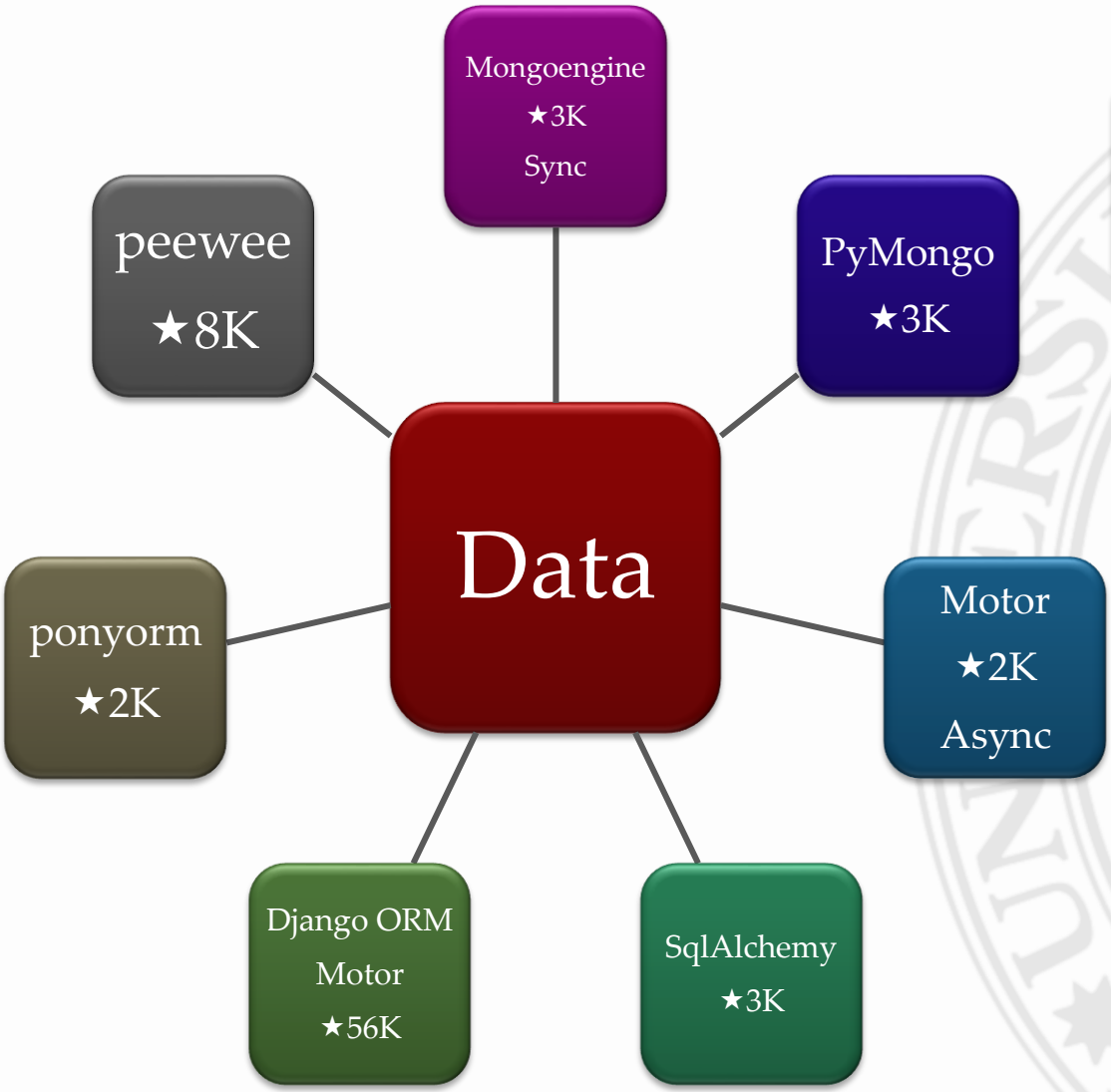
```python
def basic_authentication(credentials: HTTPBasicCredentials = Depends(HTTPBasic())) -> str:
    correct_username = secrets.compare_digest("jes", credentials.username)  # avoid timing attacks
    bd_hashed_password = bcrypt.hashpw(b"pass", bcrypt.gensalt())
    correct_password = bcrypt.checkpw(credentials.password.encode('utf-8'), bd_hashed_password)
    if not (correct_username and correct_password):
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Incorrect user or password",
        )
    return credentials.username


@auth.get("/basic")
def read_with_basic(username: str = Depends(basic_authentication)):
    return {"username": username}
```

Python
Seguridad. JWT

```python
def test_create_token_and_get(self):
    response = client.post("/auth/bearer", auth=('jes', 'pass'))
    self.assertEqual(HTTPStatus.OK, response.status_code)
    token = response.json()['token']
    response = client.get("/auth/bearer", headers={"Authorization": "Bearer " + token})
    self.assertEqual(HTTPStatus.OK, response.status_code)
```

```python
@auth.post("/bearer")
def create_jwt_token(username: str = Depends(basic_authentication)):
    expire = datetime.utcnow() + timedelta(minutes=60)
    encoded_jwt = jwt.encode({'sub': username, 'exp': expire}, 'secret_key-Y01oEAl3iz', algorithm='HS256')
    return {"token": encoded_jwt}


def bearer_authentication(credentials: HTTPAuthorizationCredentials = Depends(HTTPBearer())) -> str:
    try:
        payload = jwt.decode(credentials.credentials, 'secret_key-Y01oEAl3iz', algorithms=["HS512", "HS256"])
        username: str = payload.get("sub")
        if username is None:
            raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED, detail="Non username")
        if username not in ["jes"]:
            raise HTTPException(
                status_code=status.HTTP_403_FORBIDDEN, detail="Not enough permissions")
        return username
    except jwt.DecodeError:
        raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED, detail="Invalid token")
    except jwt.ExpiredSignatureError:
        raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED, detail="expired token")


@auth.get("/bearer")
def read_with_jwt_token(username: str = Depends(bearer_authentication)):
    return {"username": username}
```

Python
Persistencia



Mongoengine

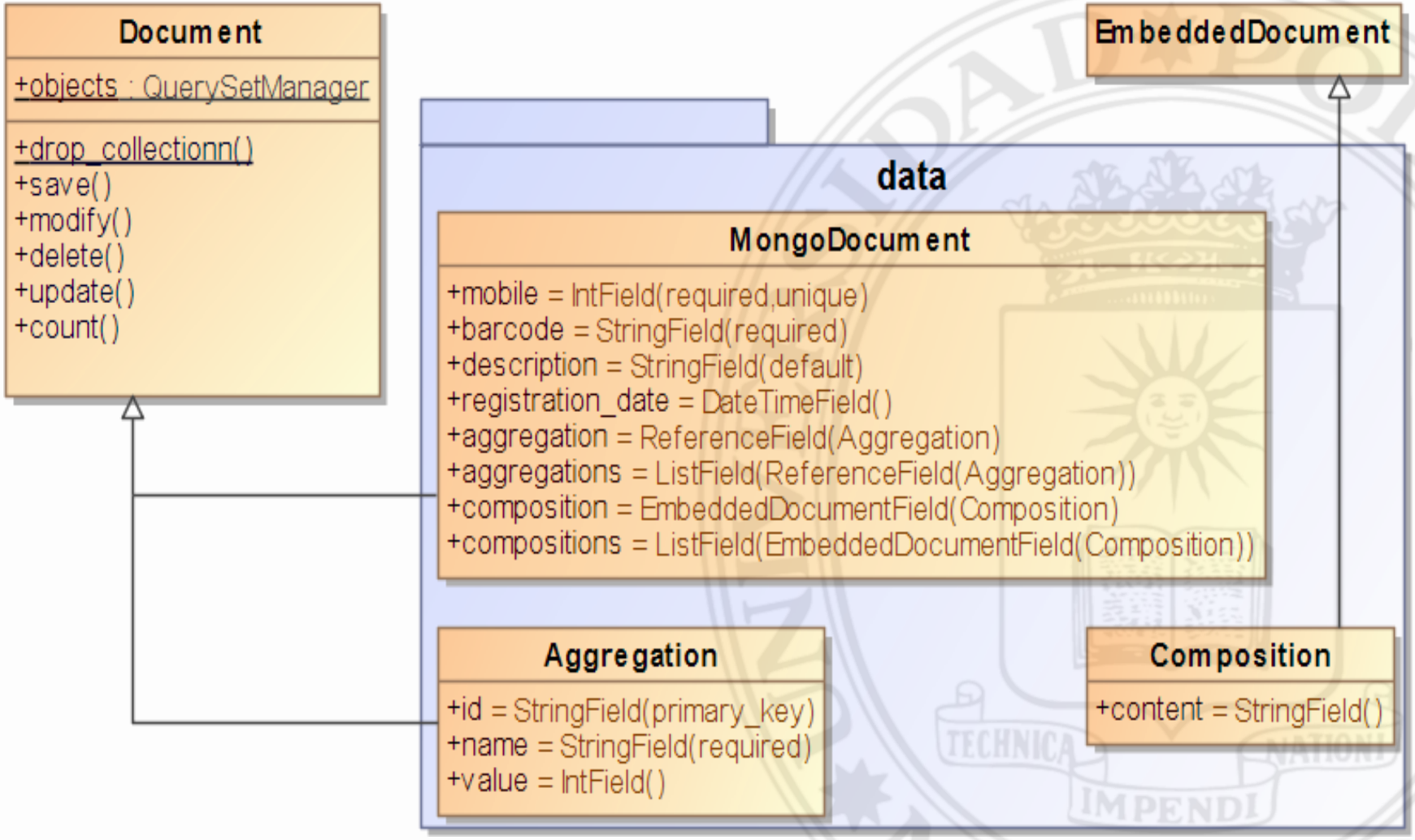
Proporciona un modelo de acceso a datos de alto nivel. Basado en *PyMongo*

http://mongoengine.org/
http://docs.mongoengine.org/

Instalación

- python -m pip install mongoengine

Python
Persistencia. Monoengine



Persistencia. betca-python

✎ {→}

https://github.com/miw-upm/betca-python

- **betca-rest**

✎ Rest en práctica

- *documents.py*

Python
TPV - Arquitectura