

Universidad de Sevilla

Escuela Técnica Superior de Ingeniería Informática



Grado en Ingeniería Informática del Software

Diseño y Pruebas II

Curso 2021/2022

ARQUITECTURE REPORT WIS D05

Repositorio: <https://github.com/Ginpasfer/Acme-Recipes>

Grupo de Prácticas	S07
Estudiantes	Rol
Pastor Fernández, Ginés	Project Manager Developer Operator Tester
Giráldez Álvarez, Pablo	Developer Analyst Tester
Rijo Hernández, Badayco	Developer Tester
Solís Miranda, Antonio Manuel	Developer Tester
Paradas Borrego, Álvaro	Developer Tester

Índice

1. Resumen ejecutivo	3
2. Tabla de revisiones	3
3. Introducción.....	3
4. Contenido	3
5. Conclusión.....	9
6. Bibliografía	9

1. Resumen ejecutivo

En este reporte se va a documentar lo que ha aprendido el grupo sobre la arquitectura implementada en el proyecto. Se tendrá en cuenta el documento realizado en la primera entrega de la asignatura en el que se preguntaba acerca de que se sabía sobre la arquitectura a implementar. Por lo que se verá si las ideas que el equipo puso en común y plasmó en ese documento iban encaminadas a lo que deparaba la arquitectura del proyecto a realizar.

2. Tabla de revisiones






















Versión	Fecha	Autor	Descripción de cambios
1.0	04/09/2022	Pablo Giráldez Álvarez	- Creación del documento

3. Introducción

























En la primera entrega, como se comentó antes, el equipo realizó y entregó un documento que trataba sobre la información que se conocía sobre un Sistema de Información Web. Ahora, una vez el equipo se encuentra en la última entrega, comentará lo que ha aprendido y qué sabe sobre la arquitectura del proyecto basándose y mencionando lo que ya se introdujo en ese documento. Se verá también como se ha estructurado el proyecto de la asignatura, adjuntando las fotos necesarias para mostrar lo que se cuenta. Se hará referencia en gran medida a la arquitectura REST y el patrón Modelo Vista Controlador (MVC). Por último, el equipo expondrá y pondrá en común sus conclusiones sobre lo visto acerca de la arquitectura de este proyecto y se incluirán referencias bibliográficas si procede.

4. Contenido

Con respecto a la arquitectura REST, esta es una arquitectura entre cliente y servidor. El cliente se encarga de enviar peticiones para crear, modificar o recuperar datos, a lo que el servidor responde mostrando el resultado. En esta arquitectura, la petición está formada por un verbo HTTP el cual define la acción a realizar. Este verbo puede ser un GET (por ejemplo, cuando obteníamos una lista), un POST (por ejemplo, cuando creábamos un cierto objeto rellenando el formulario pertinente), un PUT (para actualizar un recurso) o un DELETE (para borrar un recurso). El navegador en este caso era el encargado de enviar las peticiones HTTP al servidor y mostraba por pantalla el resultado, valiéndose de los recursos de la base de datos. Como servidor de la base de datos, se ha usado Tomcat, que se encargaba de recibir las peticiones que enviaba el navegador, procesar dichas peticiones y devolver la respuesta a la petición. Como base de datos para almacenar los recursos de la aplicación se ha usado DBever, donde se guardaba la información y entradas en sus respectivas tablas sobre cada objeto que se utilizaba en la aplicación.

▼  acme-recipes-22.8	
▼  Tables	
>  administrator	32K
>  anonymous	32K
>  any	32K
>  authenticated	32K
>  bulletin	16K
>  chef	32K
>  consumer	32K
>  currency	16K
>  dish	64K
>  epicure	32K
>  hibernate_sequence	16K
>  item	48K
>  item_quantity	48K
>  memorandum	32K
>  peep	16K
>  provider	32K
>  recipe	48K
>  system_configuration	16K
>  user_account	32K

También, en las carpetas “src/main/webapp/WEB-INF/resources/initial-data” y “src/main/webapp/WEB-INF/resources/sample-data” se almacenaban ficheros “.csv” con los datos sobre las distintas entidades creadas en las clases Java.

▼  > Acme-Recipes-22.8 [Acme-Recipes-22.8 D04]
>  src/main/java
▼  src/main/webapp
>  META-INF
▼  WEB-INF
▼  resources
▼  initial-data
 administrator.csv
 anonymous.csv
 authenticated.csv
 system-configuration.csv
 user-account.csv
▼  sample-data
 bulletin.csv
 chef.csv
 dish.csv
 epicure.csv
 item.csv
 itemQuantity.csv
 memorandum.csv
 peep.csv
 recipe.csv
 user-account.csv
>  views

Como servidor de la base de datos, en esta asignatura se ha usado MariaDB.

```
start-mariadb.cmd - Acceso directo
2022-09-04 21:12:06 0 [Note] .\bin\mysqld (server 10.6.5-MariaDB-log) starting as process 2064 ...
2022-09-04 21:12:06 0 [Note] InnoDB: Compressed tables use zlib 1.2.11
2022-09-04 21:12:06 0 [Note] InnoDB: Using transactional memory
2022-09-04 21:12:06 0 [Note] InnoDB: Number of pools: 1
2022-09-04 21:12:06 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
2022-09-04 21:12:06 0 [Note] InnoDB: Initializing buffer pool, total size = 134217728, chunk size = 134217728
2022-09-04 21:12:06 0 [Note] InnoDB: Completed initialization of buffer pool
2022-09-04 21:12:06 0 [Note] InnoDB: Starting crash recovery from checkpoint LSN=192211670,192211670
2022-09-04 21:12:06 0 [Note] InnoDB: Last binlog file '.\mysql-bin.000099', position 11172482
2022-09-04 21:12:06 0 [Note] InnoDB: 128 rollback segments are active.
2022-09-04 21:12:06 0 [Note] InnoDB: Removed temporary tablespace data file: "ibtmp1"
2022-09-04 21:12:06 0 [Note] InnoDB: Creating shared tablespace for temporary tables
2022-09-04 21:12:06 0 [Note] InnoDB: Setting file 'ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...
2022-09-04 21:12:06 0 [Note] InnoDB: File 'ibtmp1' size is now 12 MB.
2022-09-04 21:12:06 0 [Note] InnoDB: 10.6.5 started; log sequence number 192211750; transaction id 315346
2022-09-04 21:12:06 0 [Note] Plugin 'FEEDBACK' is disabled.
2022-09-04 21:12:06 0 [Note] InnoDB: Loading buffer pool(s) from C:\Users\giral\Documents\Archivos universidad\3_TERCERO\DP2\Workspace-22.0\Workspace-22.0\Tools\Servers\mariadb-10.6.5\data\ib_buffer_pool
2022-09-04 21:12:06 0 [Note] Recovering after a crash using mysql-bin
2022-09-04 21:12:06 0 [Note] Starting table crash recovery...
2022-09-04 21:12:06 0 [Note] Crash table recovery finished.
2022-09-04 21:12:06 0 [Note] InnoDB: Buffer pool(s) load completed at 220904 21:12:06
2022-09-04 21:12:06 0 [Note] Server socket created on IP: '::'.
2022-09-04 21:12:06 0 [Note] Server socket created on IP: '0.0.0.0'.
2022-09-04 21:12:06 0 [Note] .\bin\mysqld: ready for connections.
Version: '10.6.5-MariaDB-log' socket: '' port: 3306 mariadb.org binary distribution
```

Con respecto al patrón MVC, como se dijo en el documento de la primera entrega, es un patrón arquitectónico que separa los datos de una aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintas:

- **Modelo:** Es la representación específica de la información con la que se opera. Incluye los datos y la lógica para operar con ellos.
- **Vista:** Es la presentación del modelo de forma adecuada para interactuar con ella, normalmente a través de una interfaz de usuario.
- **Controlador:** Responde a los eventos de la interfaz de usuario e invoca cambios en el modelo y probablemente en la vista. Es el intermediario entre vista y modelo.

El modelo se constituye de tres elementos:

- Entidades: Un objeto de dominios cuyos datos se conservan en la base de datos y tiene su propia identidad.

```
@Entity
@Getter
@Setter
public class Bulletin extends AbstractEntity{
    // Serialisation identifier -----

    protected static final long serialVersionUID = 1L;

    // Attributes -----

    @Temporal(TemporalType.TIMESTAMP)
    @Past
    @NotNull
    protected Date instantiationMoment;

    @NotBlank
    @Length(max=100)
    protected String heading;

    @NotBlank
    @Length(max=255)
    protected String text;

    protected boolean critical;

    @URL
    protected String info;

    // Derived attributes -----
    // Relationships -----
}
```

- Repositorios: Proporcionan métodos para guardar y recuperar objetos de dominio en la base.

```
12 public interface AdministratorBulletinRepository extends AbstractRepository{
13
14     @Query("select b from Bulletin b where b.id = :id")
15     Bulletin findOneBulletinById(int id);
16
17     @Query("select b from Bulletin b")
18     Collection<Bulletin> findAllBulletins();
19
20     @Query("select b from Bulletin b where b.instantiationMoment > :deadline")
21     Collection<Bulletin> findRecentBulletins(Date deadline);
22
23     @Query("select c from SystemConfiguration c")
24     SystemConfiguration getSystemConfiguration();
25
26 }
```

- Servicios: Exponen la funcionalidad del dominio como API. Se suelen organizar en términos de entidades.

```

19 @Service
20 public class AdministratorBulletinCreateService implements AbstractCreateService<Administrator, Bulletin> {
21
22     // Internal state -----
23
24     @Autowired
25     protected AdministratorBulletinRepository repository;
26
27     // AbstractCreateService<Administrator, Bulletin> interface -----
28
29
30     @Override
31     public boolean authorise(final Request<Bulletin> request) {
32         assert request != null;
33
34         return true;
35     }
36
37     @Override
38     public void bind(final Request<Bulletin> request, final Bulletin entity, final Errors errors) {
39         assert request != null;
40         assert entity != null;
41         assert errors != null;
42
43         request.bind(entity, errors, "heading", "text", "critical", "info");
44     }
45

```




























Los controladores responden a eventos en la interfaz de usuario e invoca cambios en el modelo y, probablemente, en la vista. Transforman las entidades en un ModelMap que puede ser procesado por las vistas, realizan la validación de entrada proporcionada por las vistas, transforman, si es necesario, esta entrada en entidades, llaman a los servicios para realizar acciones y devuelven el nombre de la vista que se debe cargar. El controlador debe mandar dos partes de información: la vista que se va a utilizar y la información que se va a utilizar en la vista, es decir, el modelo.

```

1 @Controller
2 public class AdministratorBulletinController extends AbstractController<Administrator, Bulletin>{
3
4     // Internal state -----
5
6
7     @Autowired
8     protected AdministratorBulletinCreateService createService;
9
10    // Constructors -----
11
12
13    @PostConstruct
14    protected void initialise() {
15
16        super.addCommand("create", this.createService);
17    }
18
19 }

```

Y las vistas representan la información proporcionada por el controlador en el ModelMap. La configuración de la vista específica se realiza en “applications.properties”. Las bibliotecas que se usan en la tecnología de la vista deben estar incluidas en el proyecto. Se suele utilizar JSP, cuyas librerías se incluyen mediante dependencias en la configuración del proyecto. Todas las vistas de cada funcionalidad se han realizado en formato “.jsp” y se almacenan en la ruta “src/main/webapp/WEBINF/views”.

- ▼  views
 - ▼  administrator
 - >  administrator-dashboard
 - >  bulletin
 - >  system-configuration
 - ▼  any
 - >  item
 - >  item-quantity
 - >  peep
 - >  recipe
 - >  user-account
 - ▼  authenticated
 - >  bulletin
 - >  chef
 - >  epicure
 - >  system-configuration
 - ▼  chef
 - >  dish
 - >  item
 - >  item-quantity
 - >  memorandum
 - >  recipe
 - ▼  epicure
 - >  dish
 - >  epicure-dashboard
 - >  memorandum
 - >  fragments

5. Conclusión

Como conclusión, el equipo entero considera que, con respecto a la asignatura de DP1, ciertas clases como los controladores y los servicios han sido mucho más fáciles realizar, ya que el contenido para cada uno de estos era casi idéntico para cada entidad y funcionalidad. Al equipo le costó un poco al principio entender la estructura del proyecto y cómo funcionaba todo, pero a partir de esto, la dificultad que entrañaba la asignatura solo incidía en cómo realizar lo que nos pedían, aunque sabiendo siempre como estructurar la idea (creando el controlador, los servicios, el repositorio, las vistas, etc).

6. Bibliografía

No aplica intencionalmente.