

EAPLI

# Sample Project eCafeteria

# eCafeteria

It is intended to develop a system that comprises a set of services related to the operation of a canteen in an educational establishment.

The system supports different types of users and services, in particular:

- **User:** consultation of menus, management of meal reservations, consultation of the current account and balance;
- **Cashier:** delivery of meals and loading of cards;
- **Menu Manager:** definition of the types of dishes, definition of dishes (recipes), and allergens, consultation and elaboration of menus, analysis of users' preferences. Position typically performed by nutritionists.

The eCafeteria project that we describe in this lesson is a first release of a system designed for this domain.

The eCafeteria system is not integrated with payment systems.

Meals can be booked until the previous day as long as the user's account has the required amount available. Payments are made in person at the cashiers at a specific time outside meal times. The account can be credited / loaded in one of the canteen boxes.

The main functional areas of the application are:

1. **Meal booking** - canteen users must register in advance on the electronic platform, which allows the booking and cancellation of meals. They can make consumption inquiries and also have an alert service available depending on the account balance;
2. **Cashier** - delivery of meals;
3. **Menus** - allows you to prepare and make weekly menus available. The analysis of previous sales and reservations is essential to determine which menus meet users' preferences.



# eCafeteria

- Cafeteria management
- Users, Kitchen and Menu managers, Cashiers
- User app
- Backoffice app
  - Kitchen management
  - Menu management
- POS
  - Delivery station

pag\_isep / ecafeteria-base / docu x +

← → ↻ bitbucket.org/pag\_isep/ecafeteria-base/src/master/documentation/

+

+

+

Java

ecafeteria-base

<>

Source

🔗

Commits

🔗

Branches

🔗

Pull requests

🔄

Pipelines

☁

Deployments

📧

Issues

🔗

Jira Software

BETA

📖

Wiki

📄

Downloads

⚙

Settings

Paulo Sousa / ecafeteria-base

documentation

Check out ⋮

Sample repository for a DDD course unit (EAPLI) at the School of Engineering at Polytechnic Institute Of Oporto

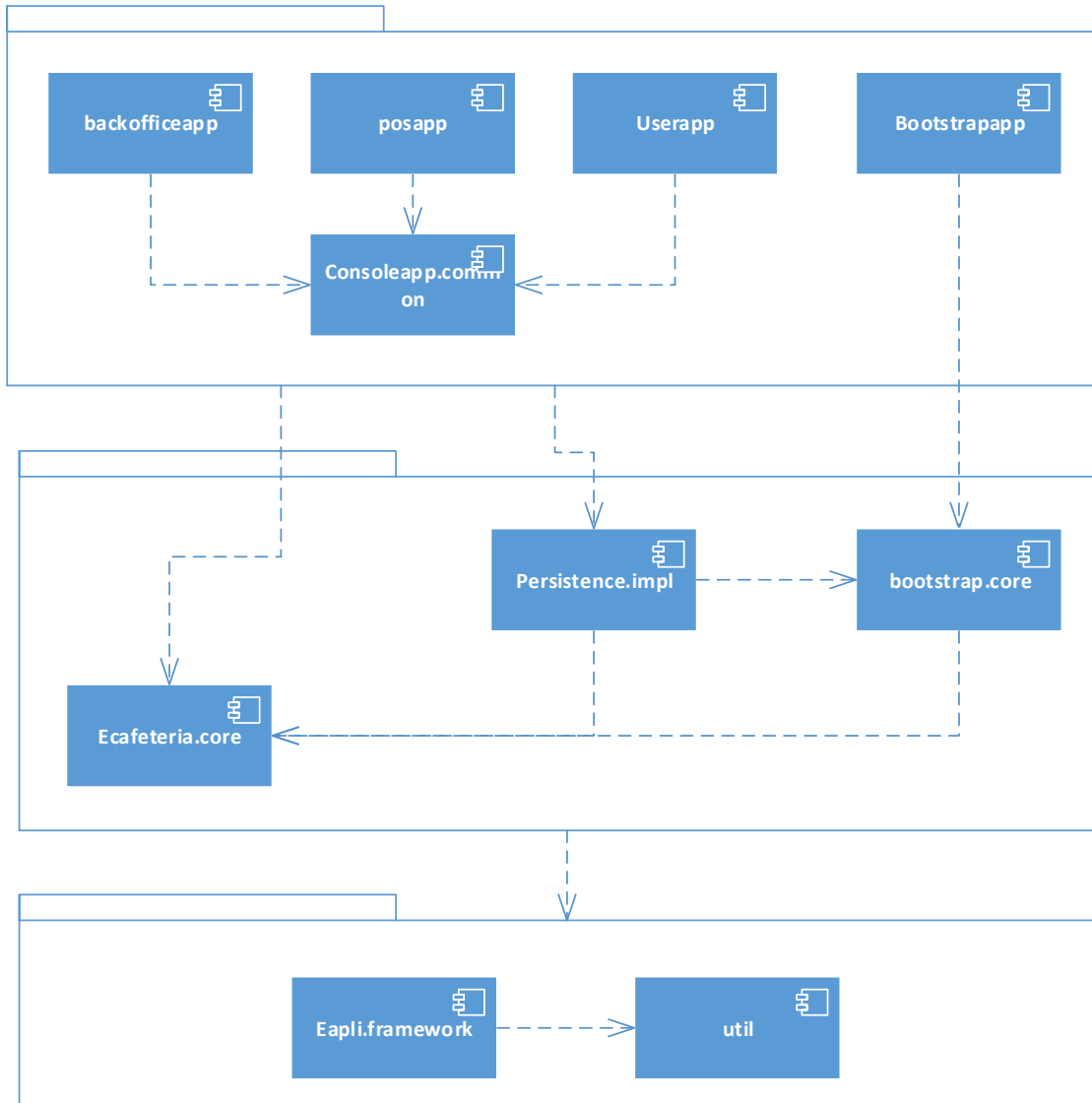
🔗 master ▾

Filter files

ecafeteria-base / documentation

Name	Size	Last commit	Message
⬆ ..			
📁 ActivateDeactivateDish		2017-05-08	initial commit v1.1.0
📁 ChangeDish		2017-05-08	initial commit v1.1.0
📁 ChangeDishType		2017-05-08	initial commit v1.1.0
📁 ListDishTypes		2017-05-08	initial commit v1.1.0
📁 MealBooking		2020-02-17	added missing doc
📁 RegisterDish		2018-02-26	added aggregate and cascade, fetch info to desig...
📁 RegisterDishType		2017-05-08	initial commit v1.1.0
📁 RegisterMaterial		2017-05-08	initial commit v1.1.0
📁 RegisterMeal		2020-02-17	added missing doc
📁 RegisterUser		2017-05-08	initial commit v1.1.0

# Components (a.k.a. projects)



- Backofficeapp, userapp, pos
- Core, console.common
- Persistence.impl
- bootstrap
- Framework
  - Utility classes for DDD applications with JPA in EAPLI context
- Util
  - Generic utility classes

# pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="...">
  <modelVersion>4.0.0</modelVersion>
  <groupId>eapli</groupId>
  <artifactId>ecafeteria</artifactId>
  <version>1.3.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <properties> ... </properties>
  <modules> ... </modules>
  <dependencies>
    <dependency>
      <groupId>eapli</groupId>
      <artifactId>eapli.framework.core</artifactId>
      <version>10.0.0</version>
    </dependency>
    <dependency>
      <groupId>eapli</groupId>
      <artifactId>eapli.framework.infrastructure.authz</artifactId>
      <version>10.0.0</version>
    </dependency>
    <dependency>
      <groupId>eapli</groupId>
      <artifactId>eapli.framework.infrastructure.pubsub</artifactId>
      <version>10.0.0</version>
    </dependency>
    <dependency> ... </dependency>
    <dependency> ... </dependency>
  </dependencies>
  <repositories>
    <repository>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <id>bintray-pagsousa-eapli</id>
      <url>http://dl.bintray.com/pagsousa/eapli</url>
    </repository>
  </repositories>
</project>
```

# eCafeteria design decisions

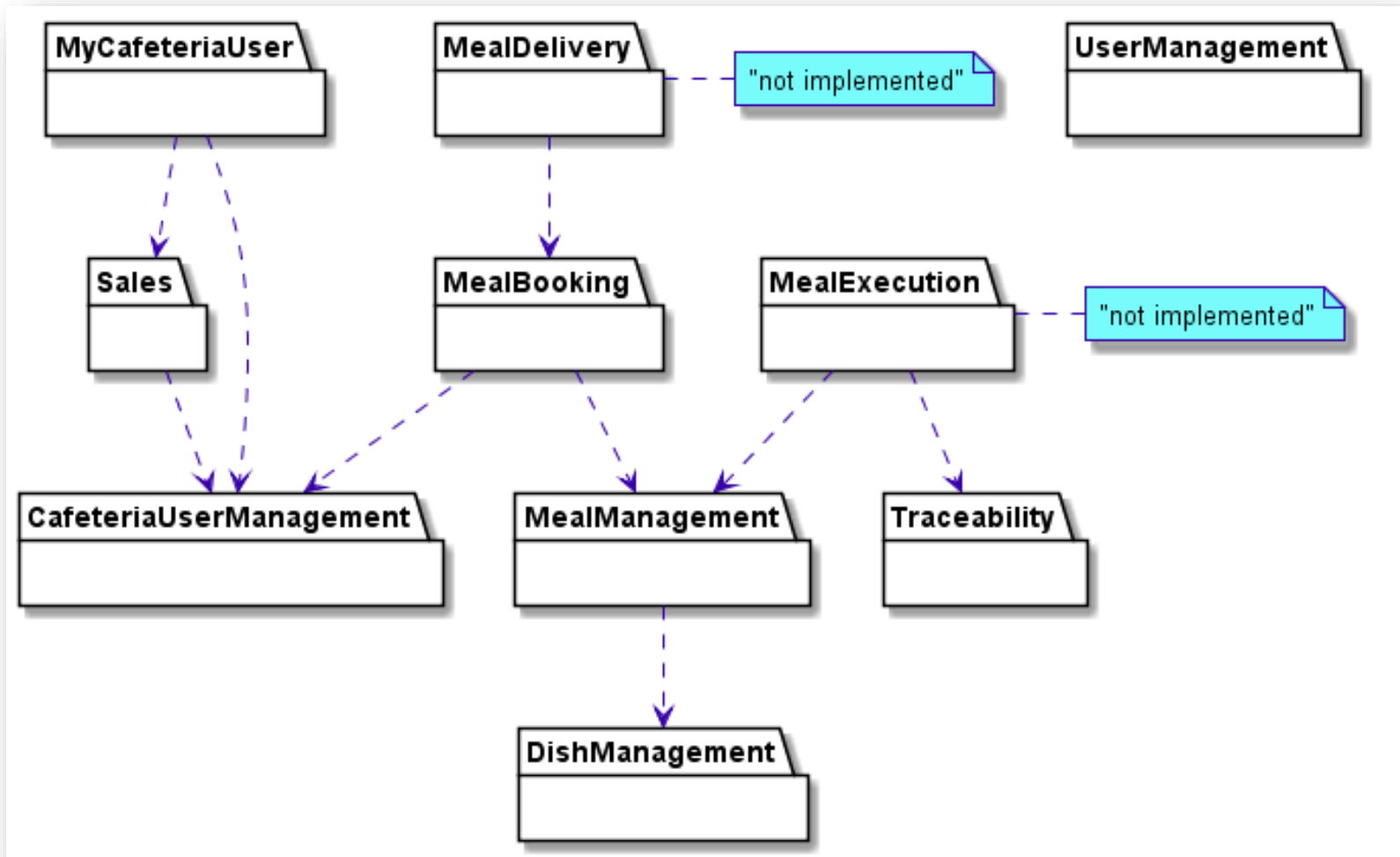
- Layers, Layered Architecture pattern
  - Business oriented
    - Presentation – view layer, user interface
    - Application – service layer, controllers
    - Domain – business logic layer, domain objects, entities
    - Persistence – data access layer, logging, repositories

DTO alternatives  
are also present

- Domain objects travel to UI for output



# Core packages



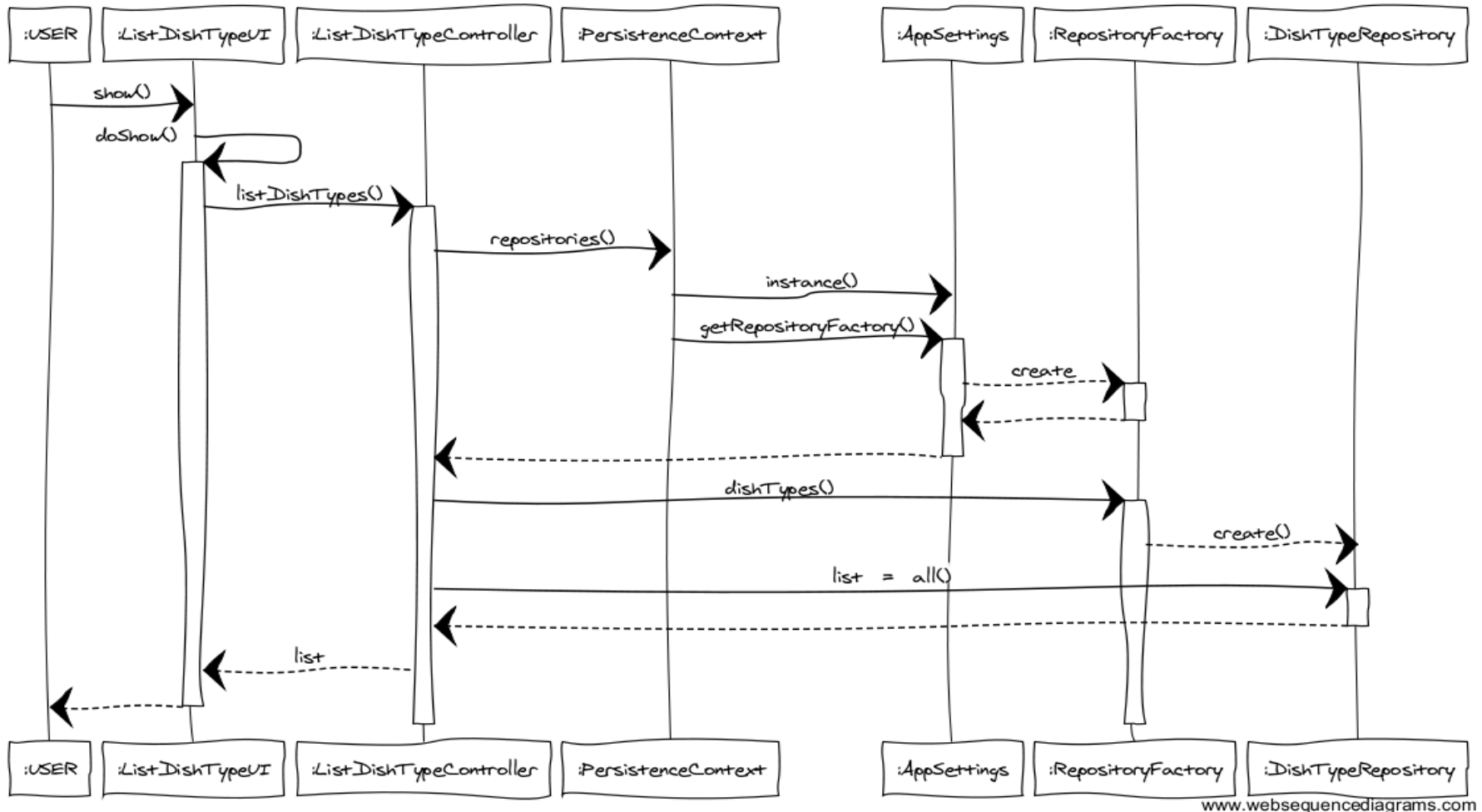


# Some additional design decisions

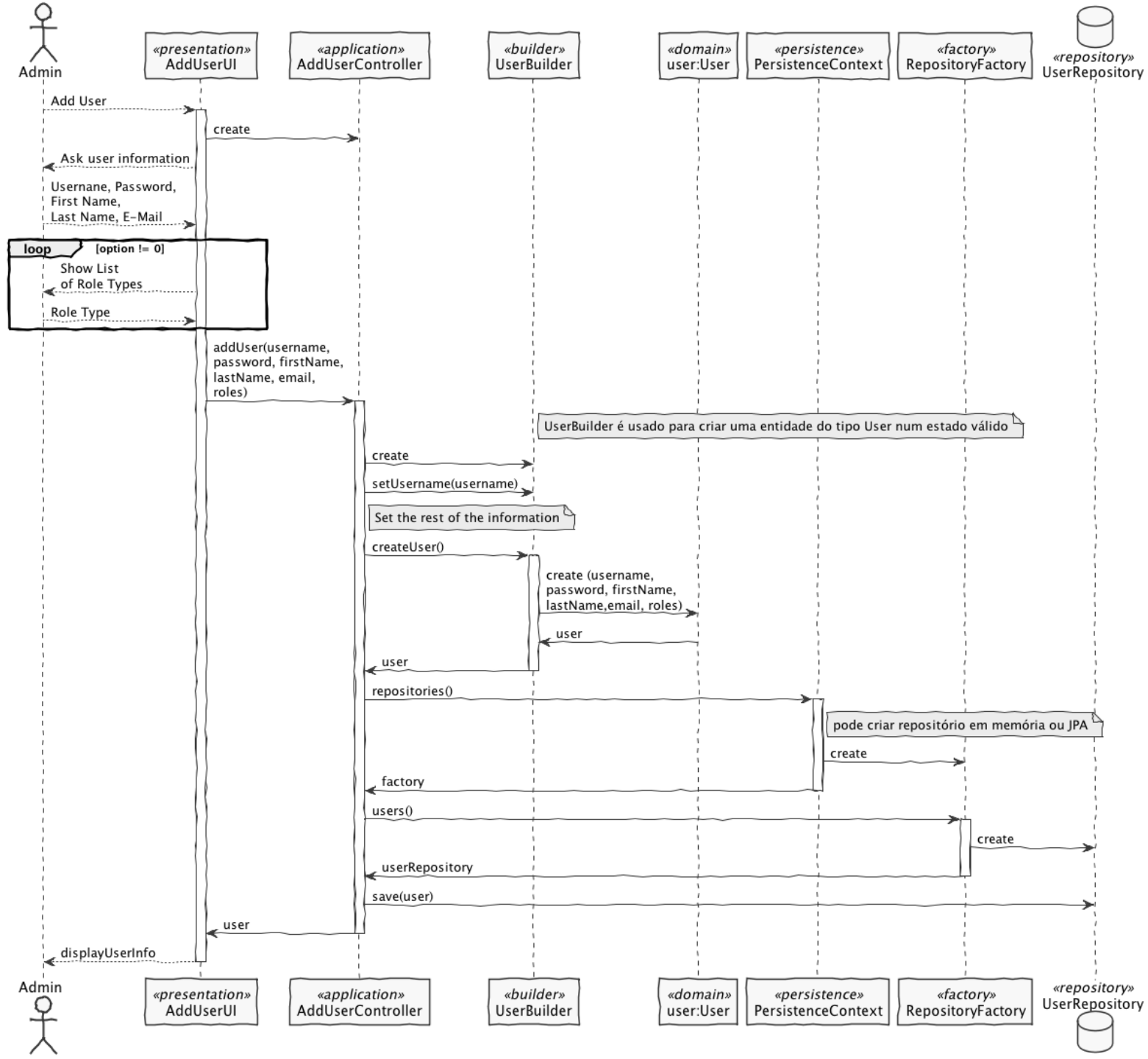
- Support two repositories
  - In memory
  - Relational database
- Decide which repository implementation to use based on property file
- Bootstrap data
- Simple main menu

# List Dish Types

SD - List All Dish Types



# Register New User



# Domain invariants as unit tests

```
@Test
public void ensurePasswordHasAtLeastOneDigitAnd6CharactersLong()
{
    new Password("abcdefgh1");
}

@Test(expected = IllegalArgumentException.class)
public void ensurePasswordsSmallerThan6CharactersAreNotAllowed()
{
    new Password("abc1c");
}

@Test(expected = IllegalArgumentException.class)
public void ensurePasswordsWithoutDigitsAreNotAllowed() {
    new Password("abcdefgh");
}
```

# Implemented Uses cases

- Backoffice > Kitchen
  - Add Material
  - List Material
- Backoffice > Chef
  - Add dish type
  - Edit dish type
  - Deactivate dish type
  - List dish types
  - Add dish
  - List dish
  - Add dish (DTO)
  - List dish (DTO)
  - Edit dish > nutritional info
  - Edit dish > price
  - Deactivate dish
  - Register meal
  - List meal
- Backoffice > Reporting
  - Dishes per type
  - High calories dishes
  - Dishes per caloric category
- Backoffice > Admin
  - Add user
  - List users
  - Deactivate user
  - Approve new user
- User
  - Signup
  - List movements
  - Book a meal
  - List my bookings
- POS
  - Recharge user account

# Interesting Uses cases

- Changing an attribute (value object) of an entity
  - Edit dish > nutritional info
  - Edit dish > price
- Reporting
  - Dishes per type
  - High calories dishes
  - Dishes per caloric category
- The current cafeteria user for UserApp
  - MyCafeteriaUserService
  - MyBookingsController
- Aggregated query
  - CafeteriaUserService.balanceOf

# Interesting Uses cases

## SEPARATE LESSONS

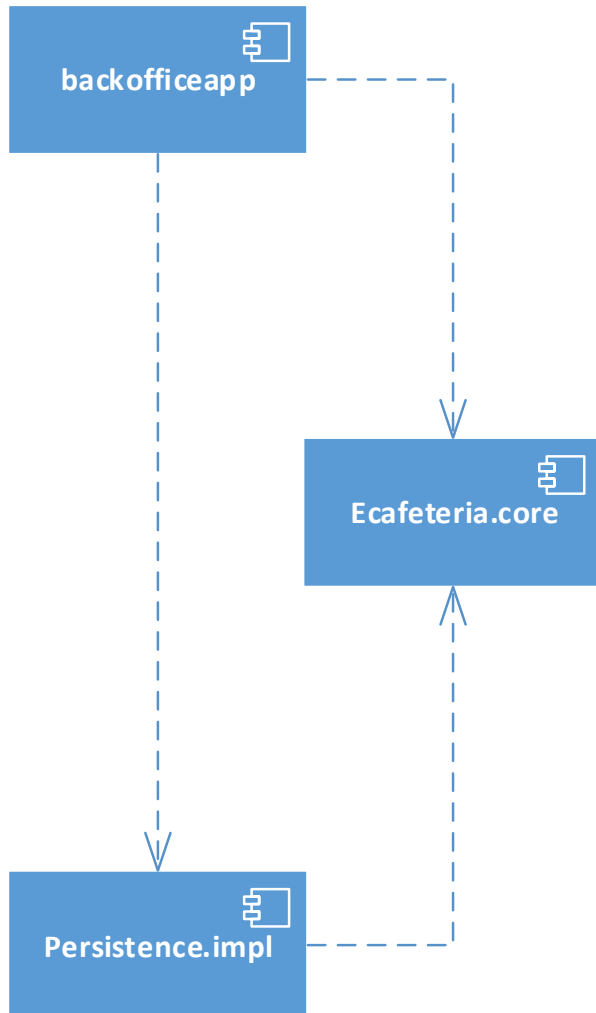
- Transactional control
  - Approve new user
  - Recharge user Card
- Domain objects or DTOs
  - Add dish vs. Add dish (DTO)
  - List dish vs List dish (DTO)
- Pub/Sub
  - Approve new user

# Persistence

PersistenceContext



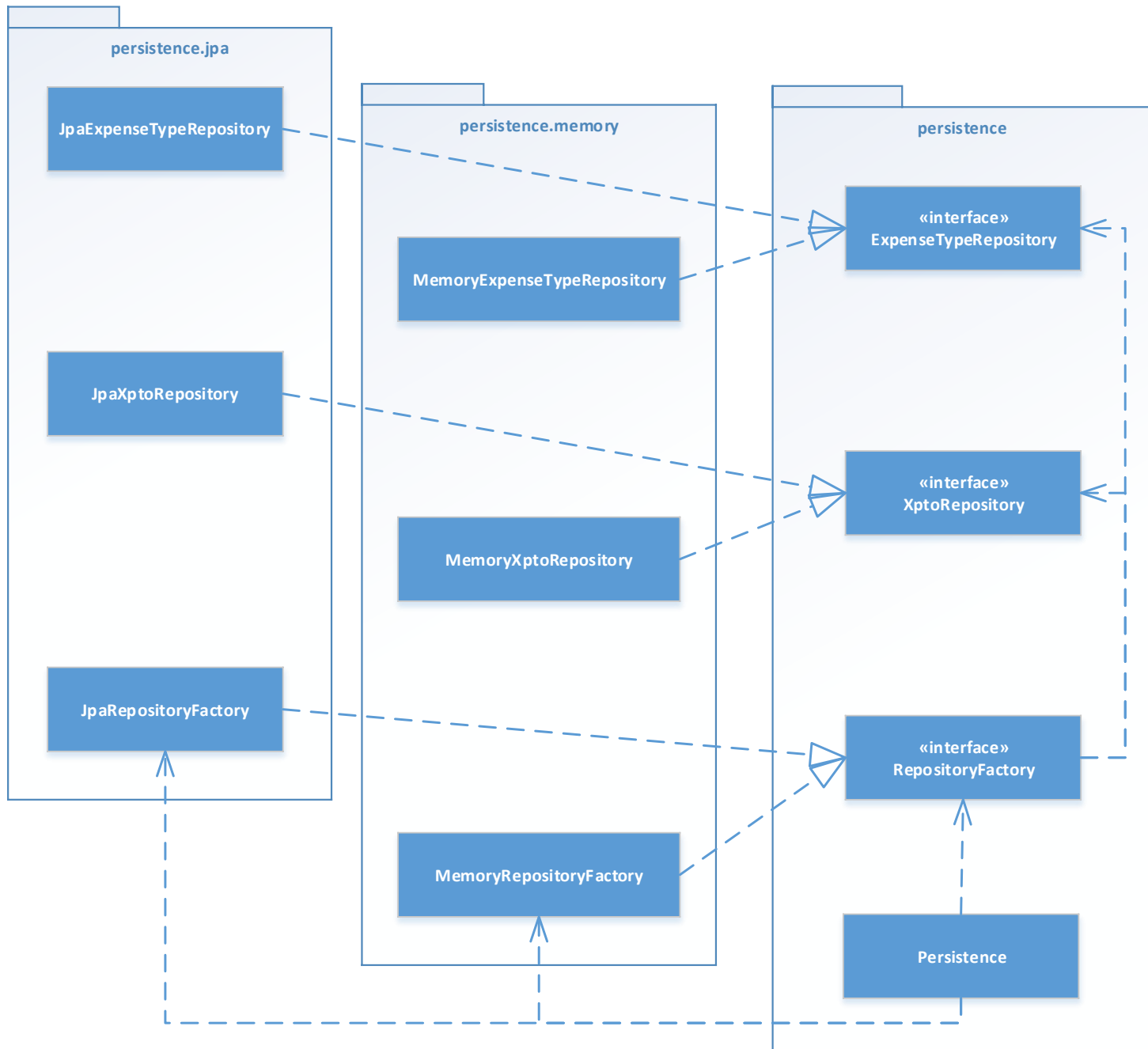
# Persistence

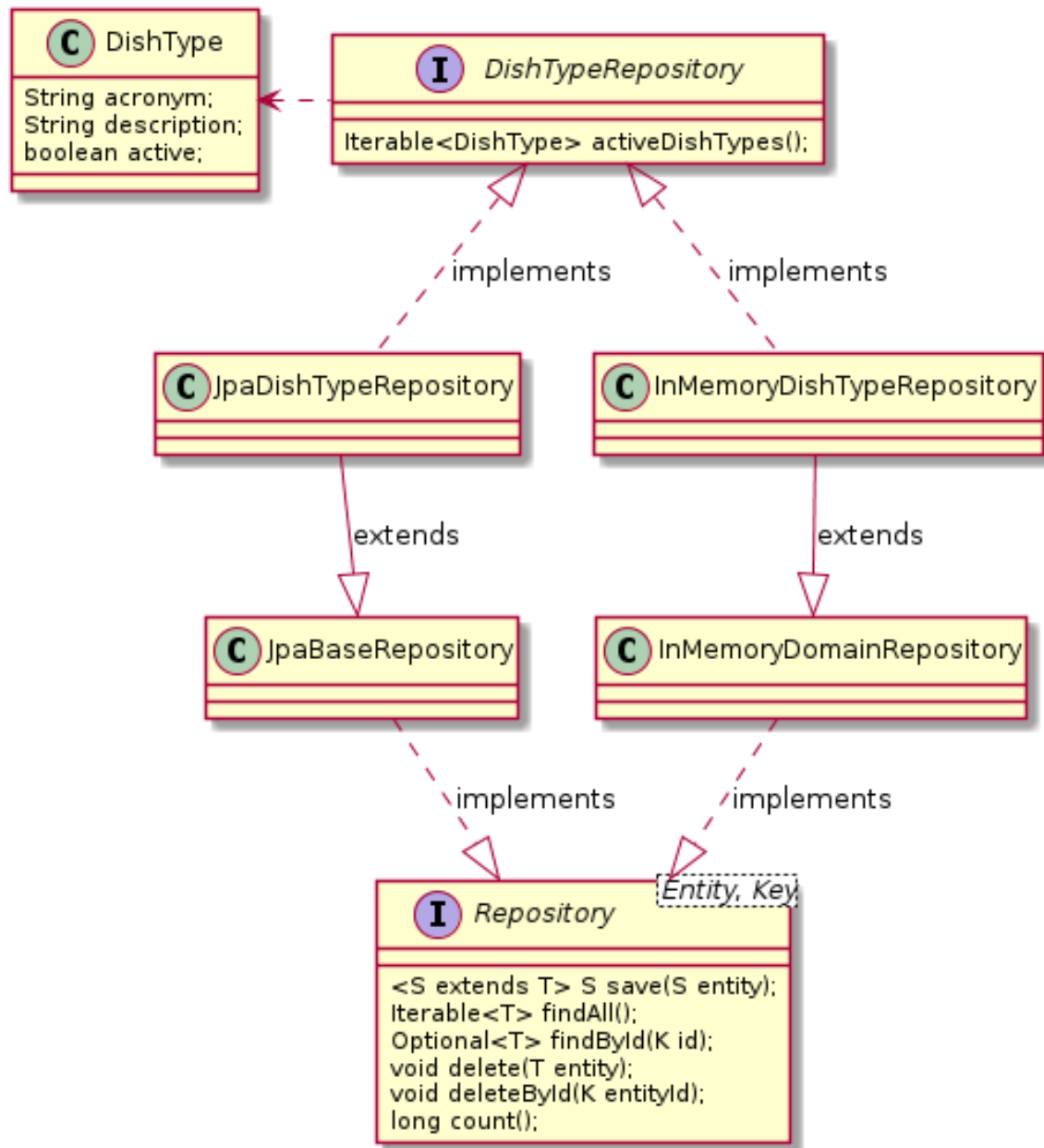


- Separate the definition of repositories (core) from the actual implementation (persistence.impl)
- Apply “Abstract Factory” GoF pattern

# Persistence

- Controller needs to access the repositories
- But we have three repository implementations
  - In memory
  - Relational database thru JPA
  - Relational database thru SpringData
- Hide persistence details from rest of the code
  - Interfaces
  - Dependency injection or Factories





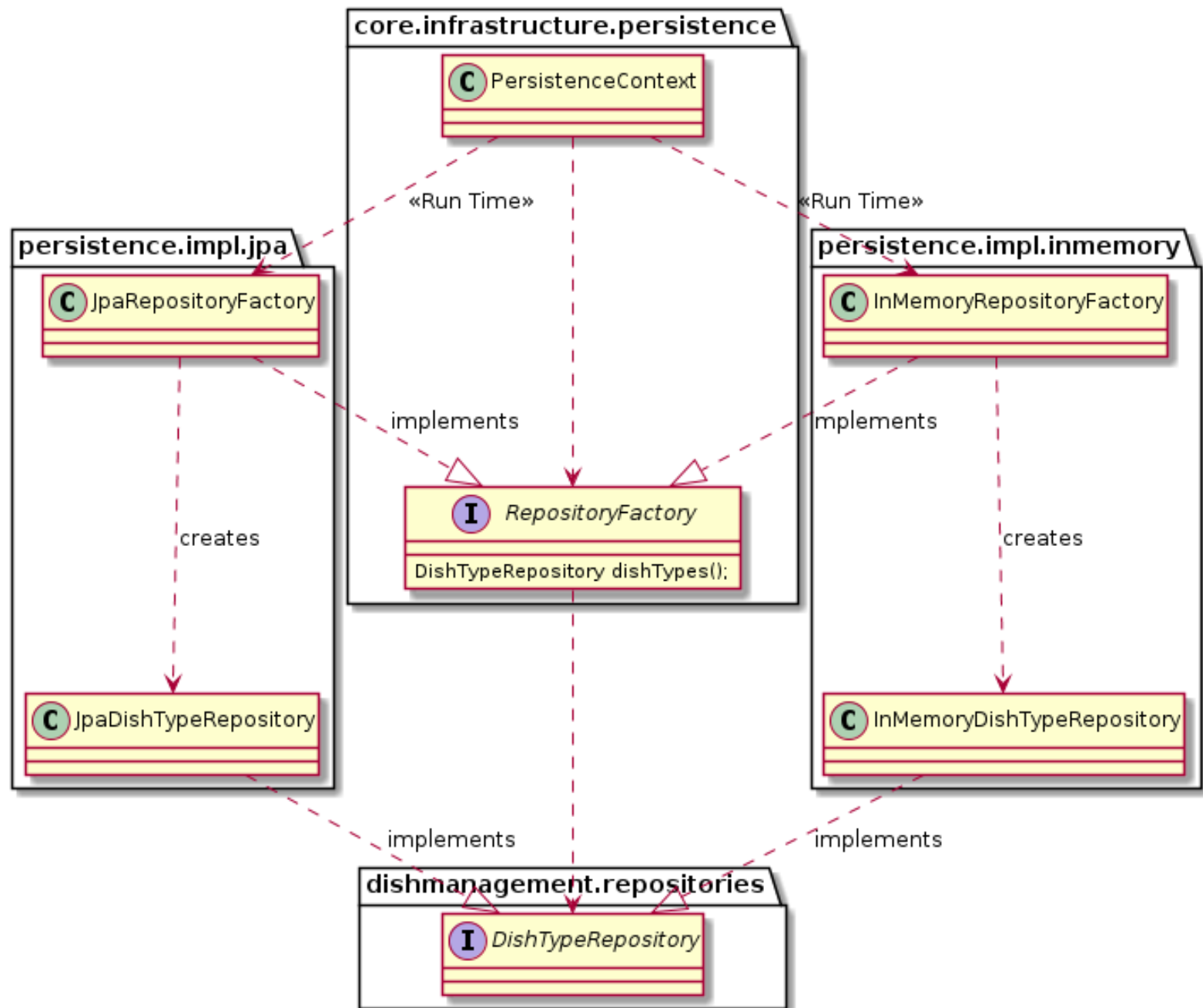
# Repository Implementations

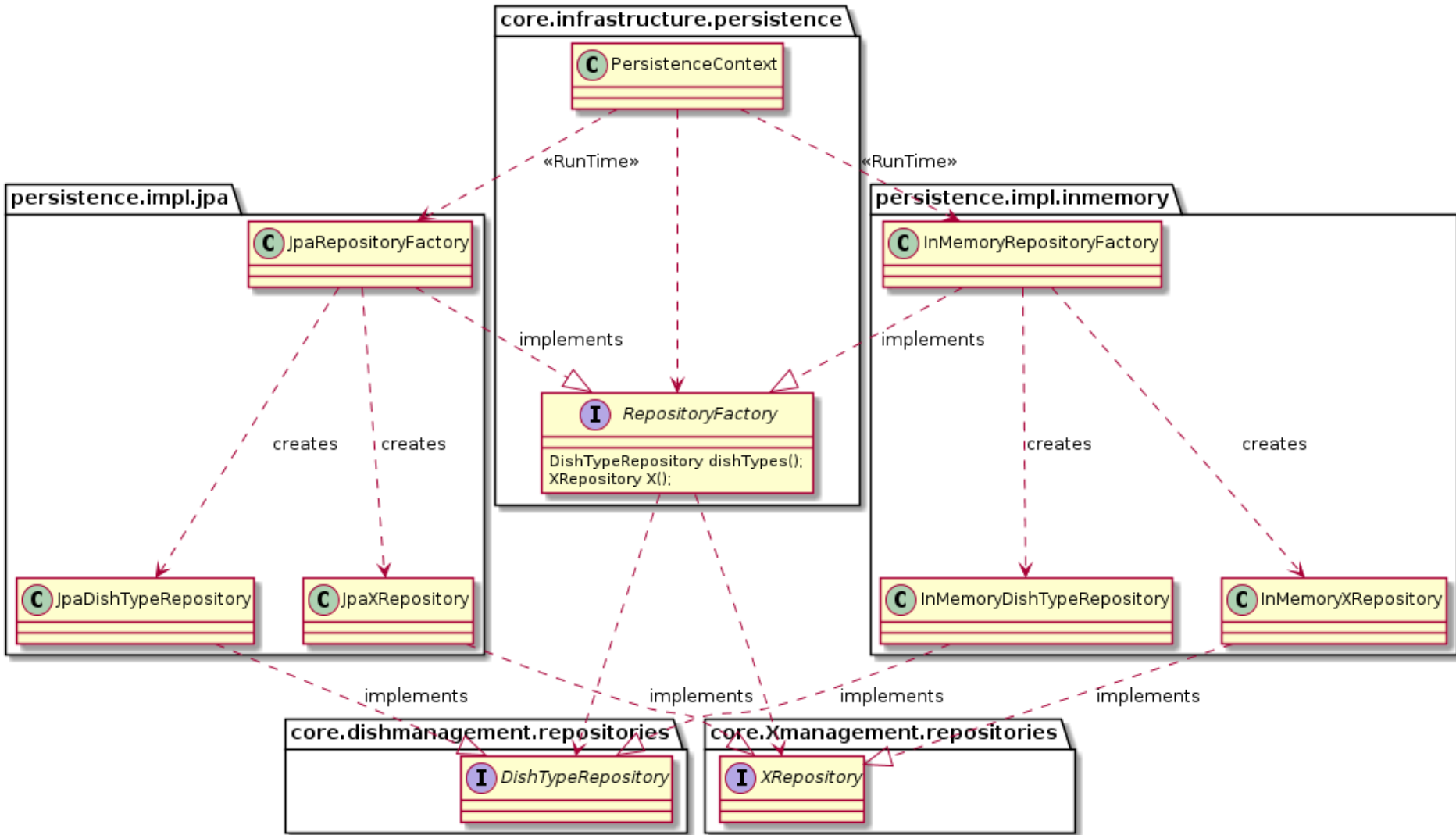
```
1 package eapli.ecafeteria.dishmanagement.repositories;
2
3 import ...
4
5
6 /** the repository for Dish Types. */
7
10 public interface DishTypeRepository extends DomainRepository<String, DishType> {
11
12     /** returns the active dish types ...*/
13
17     Iterable<DishType> activeDishTypes();
18 }
19
```

```
1 package eapli.ecafeteria.persistence.impl.inmemory;
2
3 import ...
4
5
6 /**
7  * Created by MCN on 29/03/2016.
8  */
9
12 public class InMemoryDishTypeRepository
13     extends InMemoryDomainRepository<String, DishType>
14     implements DishTypeRepository {
15
16     static {...}
17
18
19
20     @Override
21     public Iterable<DishType> activeDishTypes() { return match(DishType::isActive); }
22
23
24
```

# Repository Implementations

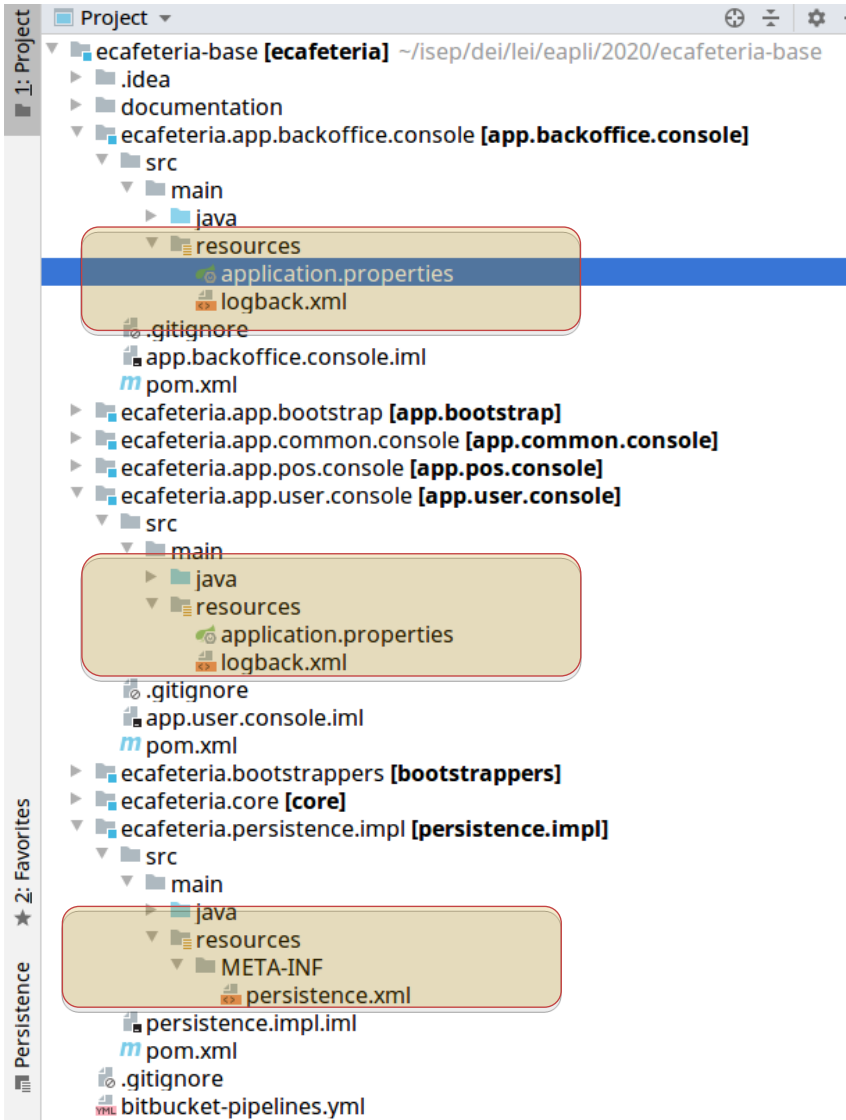
```
1  package eapli.ecafeteria.persistence.impl.jpa;
2
3  import ...
4
5
6
7
8  /**
9   * Created by MCN on 29/03/2016.
10  */
11  class JpaDishTypeRepository
12      extends CafeteriaJpaRepositoryBase<DishType, Long, String>
13      implements DishTypeRepository {
14
15      JpaDishTypeRepository() { super( identityFieldName: "acronym"); }
16
17
18
19      @Override
20      public Iterable<DishType> activeDishTypes() {
21          return match( where: "e.active=true");
22      }
23  }
```







# Resources



- Persistence.impl has persistence.xml
- Application projects define the properties file

# Persistence Context

```
39  /** provides easy access to the persistence layer. works as a factory of ...*/
45  @Utility
46  public final class PersistenceContext {
47      private static final Logger LOGGER = LogManager.getLogger(PersistenceContext.class);
48      private static RepositoryFactory theFactory;
49      private PersistenceContext() {}
52
53      public static RepositoryFactory repositories() {
54          if (theFactory == null) {
55              final String factoryClassName = Application.settings().getRepositoryFactory();
56              try {
57                  theFactory = (RepositoryFactory) Class.forName(factoryClassName).newInstance();
58              } catch (ClassNotFoundException | IllegalAccessException | InstantiationException ex) {
59                  LOGGER.error(s: "Unable to dynamically load the Repository Factory", ex);
60                  throw new IllegalStateException(
61                      "Unable to dynamically load the Repository Factory: " + factoryClassName, ex);
62              }
63          }
64          return theFactory;
65      }
66  }
```

```
4  persistence.persistanceUnit=eapli.eCafeteriaPU
5  persistence.repositoryFactory=eapli.ecafeteria.persistence.jpa.JpaRepositoryFactory
6  #persistence.repositoryFactory=eapli.ecafeteria.persistence.inmemory.InMemoryRepositoryFactory
7  ui.menu.layout=horizontal
8  HighCaloriesDishLimit=300
```

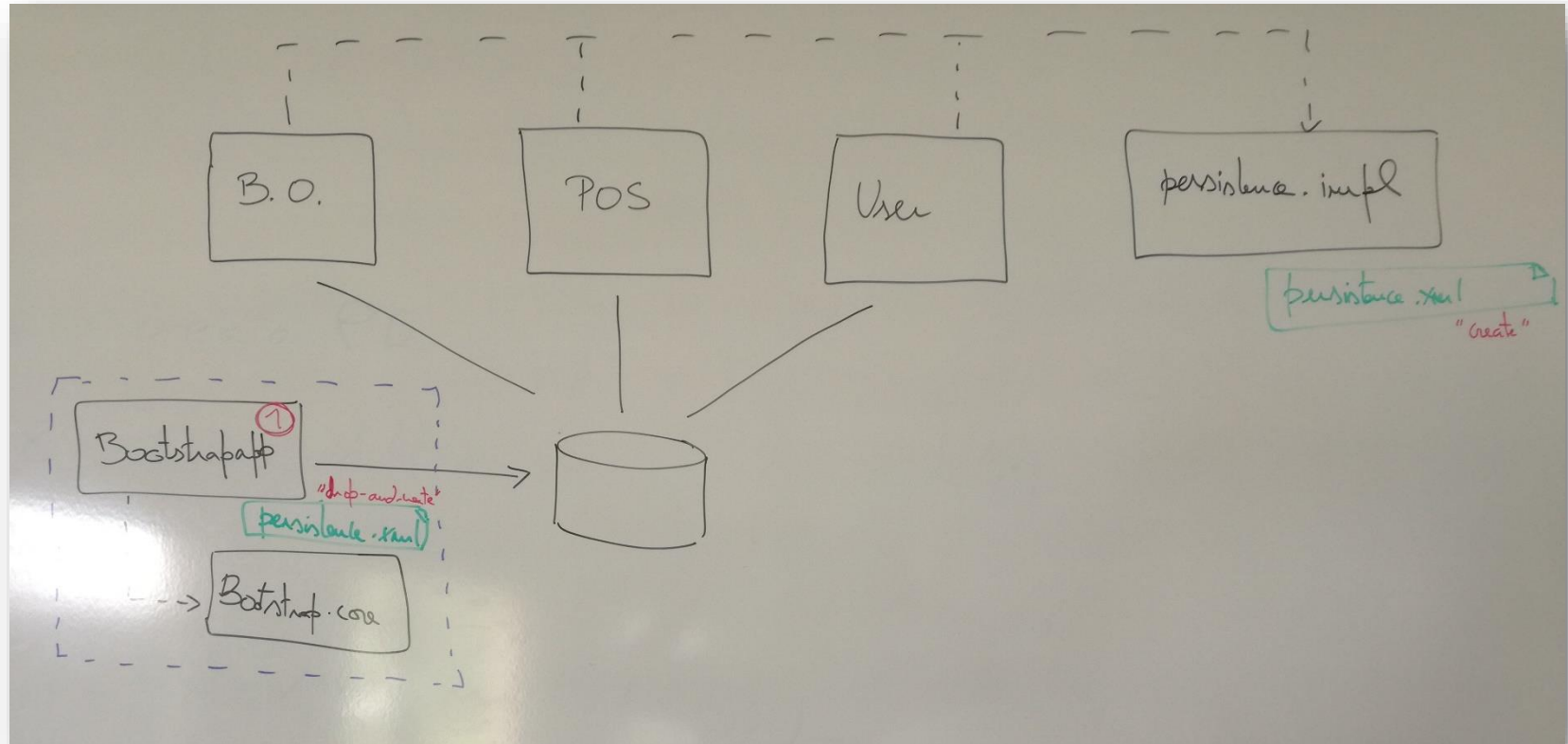
# Persistence Context Usage

```
24 package eapli.ecafeteria.dishmanagement.application;
25
26 import ...
27
33
34 /**...*/
38 public class RegisterDishTypeController implements UseCaseController {
39
40     private final AuthorizationService authz = AuthzRegistry.authorizationService();
41     private final DishTypeRepository repository = PersistenceContext.repositories().dishTypes();
42
43     public DishType registerDishType(final String acronym, final String description) {
44         authz.ensureAuthenticatedUserHasAnyOf(CafeteriaRoles.POWER_USER,
45         CafeteriaRoles.MENU_MANAGER);
46
47         final DishType newDishType = new DishType(acronym, description);
48         return this.repository.save(newDishType);
49     }
50 }
51
```

# Persistence

Bootstrap

# Bootstrap



Separate BootstrapApp for database initialization

# ECafeteriaBootstrapApp

```
ECafeteriaBootstrap.java
53 public static void main(final String[] args) {
54
55     AuthzRegistry.configure(PersistenceContext.repositories().users(),
56         new CafeteriaPasswordPolicy(),
57         new PlainTextEncoder());
58
59     new ECafeteriaBootstrap().run(args);
60 }
61
62 @Override
63 protected void doMain(final String[] args) {
64     handleArgs(args);
65
66     System.out.println("\n\n----- MASTER DATA -----");
67     new ECafeteriaBootstrapper().execute();
68
69     if (isToBootstrapDemoData) {
70         System.out.println("\n\n----- DEMO DATA -----");
71         new ECafeteriaDemoBootstrapper().execute();
72     }
73     if (isToRunSampleE2E) {
74         System.out.println("\n\n----- BASIC SCENARIO -----");
75         new ECafeteriaDemoSmokeTester().execute();
76     }
77 }
78
79 private void handleArgs(final String[] args) {
80     isToRunSampleE2E = Arrays.contains(args, "-smoke:basic");
81     if (isToRunSampleE2E) {
82         isToBootstrapDemoData = true;
83     } else {
84         isToBootstrapDemoData = Arrays.contains(args, "-bootstrap:demo");
85     }
86 }
87
88 @Override
```

# Bootstrappers

```
▼ ecafeteria.bootstrappers [ecafeteria-base master 11]
  ▼ src/main/java
    ▼ eapli.ecafeteria.infrastructure.bootstrappers
      > ECafeteriaBootstrapper.java
      > MasterUsersBootstrapper.java
      > TestDataConstants.java
      > UsersBootstrapperBase.java
    ▼ eapli.ecafeteria.infrastructure.bootstrappers.demo
      > BackofficeUsersBootstrapper.java
      > BookingBootstrapper.java
      > CafeteriaUserBootstrapper.java
      > DishBootstrapper.java
      > DishTypesBootstrapper.java
      > ECafeteriaDemoBootstrapper.java
      > MaterialsBootstrapper.java
      > MealBootstrapper.java
      > RechargeUserCardBootstrapper.java
    ▼ eapli.ecafeteria.infrastructure.smoketests
      > ECafeteriaDemoSmokeTester.java
    ▼ eapli.ecafeteria.infrastructure.smoketests.backoffice
      > DishManagementSmokeTester.java
      > DishTypeCRUDSmokeTester.java
```

- Reference data
- Demo data
- Smoke tests

# Demo data

```
E CafeteriaBootstrap.java  E CafeteriaBootstrapper.java  E CafeteriaDemoBootstrapper.java
34  * eCafeteria Bootstrapping data app
35  *
36  * @author Paulo Gandra de Sousa
37  */
38  @SuppressWarnings("squid:S106")
39  public class ECafeteriaDemoBootstrapper implements Action {
40
41      private static final String POWERUSER_A1 = "poweruserA1";
42      private static final String POWERUSER = "poweruser";
43
44      private final AuthorizationService authz = AuthzRegistry.authorizationService();
45      private final AuthenticationService authenticationService = AuthzRegistry
46          .authenticationService();
47
48      @Override
49      public boolean execute() {
50          // declare bootstrap actions
51          final Action[] actions = { new BackofficeUsersBootstrapper(), new DishTypesBootstrapper(),
52              new CafeteriaUserBootstrapper(), new RechargeUserCardBootstrapper(),
53              new DishBootstrapper(),
54              new MaterialsBootstrapper(), new MealBootstrapper(), new BookingBootstrapper() };
55
56          authenticateForBootstrapping();
57
58          // execute all bootstrapping
59          boolean ret = true;
60          for (final Action boot : actions) {
61              System.out.println("Bootstrapping " + nameOfEntity(boot) + "...");
62              ret &= boot.execute();
63          }
64          return ret;
65      }
66
67      /**
68       * authenticate a super user to be able to register new users
69       */
```



# Smoke test

```
ECafeteriaBootstra... ECafeteriaBootstra... ECafeteriaDemoBo... ECafeteriaDemoS... DishManagementSm...
33+ import org.apache.logging.log4j.LogManager;
51
52+ /**
53+  *
54+  * @author Paulo Gandra de Sousa
55+  *
56+  */
57 public class DishManagementSmokeTester implements Action {
58     private static final Logger LOGGER = LogManager.getLogger(DishManagementSmokeTester.class);
59
60     final ChangeDishTypeController changeDishTypeController = new ChangeDishTypeController();
61     final ChangeDishController changeDishController = new ChangeDishController();
62     final DishTypeCRUDSmokeTester crudTester = new DishTypeCRUDSmokeTester();
63     final ActivateDeactivateDishController activateDishController = new ActivateDeactivateDishController();
64     final DishReportingController dishReportingController = new DishReportingController();
65     final ActivateDeactivateDishTypeController activateDishTypeController = new ActivateDeactivateDishTypeController();
66
67+ @Override
68 public boolean execute() {
69     testDishTypeCRUD();
70     testActivateDeactivateDishType();
71     testChangeDishType();
72
73     testActivateDeactivateDish();
74     testChangeDish();
75     testReportingDish();
76
77     testDishJsonRepresentation();
78
79     // nothing else to do
80     return true;
81 }
82
83+ private void testDishJsonRepresentation() {
84     final Iterable<Dish> l = changeDishController.allDishes();
85     Invariants.nonNull(l);
```

# Smoke test (spring version)

```
ApplicationT... SignupResou... ApplicationT... ECafeteriaBo... ECafeteriaSa...
35     private DeliverMealsBootstrapper deliverMealsBootstrapper;
36
37     @Override
38     public boolean execute() {
39         authenticateForBootstrapping();
40
41         if (!sampleEndToEndScenario()) {
42             return false;
43         }
44
45         // nothing more to do; everything went well
46         return true;
47     }
48
49     private boolean sampleEndToEndScenario() {
50         if (!usersSignupAndApprove()) {
51             return false;
52         }
53         if (!posOpenStationAndChargeCard()) {
54             return false;
55         }
56
57         final List<BookingToken> bookings = usersBookMeal();
58         if (bookings.isEmpty()) {
59             return false;
60         }
61
62         if (!posDeliver(bookings)) {
63             return false;
64         }
65         return true;
66     }
67
```

- End to end scenario

# Persistence

Controlo Transicional

# JPA Repositories (framework)

- JpaBaseRepository
  - Generic repository implementation that expects the entity manager factory to be injected by a container, e.g., web server
- JpaNotRunningInContainerBaseRepository
  - For scenarios where the code is not running in a container but transaction is managed by the outside, e.g., controller
- JpaTransactionalBaseRepository
  - For scenarios not running in a container but transactions are created and committed by each repository method; the connection is also closed automatically in each method.
- JpaAutoTxRepository
  - Dual behaviour to either have outside transactional control or explicit transaction in each method

# Full transaction control by the repository

```
9  class JpaMaterialRepository extends CafeteriaJpaRepositoryBase<Material, Long>
10      implements MaterialRepository {
11
12      @Override
13      public Material findByAcronym(String acronym) {
14          return matchOne("e.acronym=:acronym", "acronym", acronym);
15      }
16  }
17
```

```
17  class CafeteriaJpaRepositoryBase<T, K extends Serializable>
18      extends JpaTransactionalRepository<T, K> {
19
20      CafeteriaJpaRepositoryBase(String persistenceUnitName) {
21          super(persistenceUnitName);
22      }
23
24      CafeteriaJpaRepositoryBase() {
25          super(Application.settings().getPersistenceUnitName());
26      }
27  }
```

# Explicit Transaction Control

```
public CardMovement rechargeUserCard(final CafeteriaUser user, final double ammount) {
    authz.ensureAuthenticatedUserHasAnyOf(CafeteriaRoles.POWER_USER, CafeteriaRoles.CASHIER);

    final TransactionalContext ctx = PersistenceContext.repositories()
        .newTransactionalContext();
    final CreditRechargeRepository rechargeRepo = PersistenceContext.repositories()
        .creditRecharges(ctx);
    final CardMovementRepository movementRepo = PersistenceContext.repositories()
        .cardMovements(ctx);

    ctx.beginTransaction();

    // credit recharge (POS)
    rechargeRepo.save(new CreditRecharge(user, Money.euros(ammount)));

    // credit recharge (cafeteria user's account)
    CardMovement creditMovement = new CardMovement(MovementType.RECHARGE, Money.euros(ammount),
        user);
    creditMovement = movementRepo.save(creditMovement);

    ctx.commit();

    return creditMovement;
}
```

# Transaction control (1)

- Accepting a signup request needs to
  - Create a system user
  - Create a cafeteria user
  - Change the status of the signup request
- Three different aggregates!

# Transaction control (2): use JpaAutoTxRepository

```
14 class JpaUserRepository extends JpaAutoTxRepository<SystemUser, Username>
15     implements UserRepository {
16
17     public JpaUserRepository(TransactionalContext autoTx) {
18         super(Application.settings().getPersistenceUnitName(), autoTx);
19     }
20
21 class JpaCafeteriaUserRepository
22     extends JpaAutoTxRepository<CafeteriaUser, MekanographicNumber>
23     implements CafeteriaUserRepository {
24
25     public JpaCafeteriaUserRepository(TransactionalContext autoTx) {
26         super(Application.settings().getPersistenceUnitName(), autoTx);
27     }
28
29 class JpaSignupRequestRepository
30     extends JpaAutoTxRepository<SignupRequest, Username>
31     implements SignupRequestRepository {
32
33     public JpaSignupRequestRepository(TransactionalContext autoTx) {
34         super(Application.settings().getPersistenceUnitName(), autoTx);
35     }
36 }
```



# Transaction control (3): explicit control by the controller

```
38 public class AcceptRefuseSignupRequestController implements Controller {
39
40     private final TransactionalContext TxCtx
41     = PersistenceContext.repositories().buildTransactionalContext();
42     private final UserRepository userRepository
43     = PersistenceContext.repositories().users(TxCtx);
44     private final CafeteriaUserRepository cafeteriaUserRepository
45     = PersistenceContext.repositories().cafeteriaUsers(TxCtx);
46     private final SignupRequestRepository signupRequestsRepository
47     = PersistenceContext.repositories().signupRequests(TxCtx);
48 }
```

# Transaction control (3): explicit control by the controller

```
49 public SignupRequest acceptSignupRequest(SignupRequest theSignupRequest)
50     throws DataIntegrityViolationException, DataConcurrencyException {
51     Application.ensurePermissionOfLoggedInUser(ActionRight.ADMINISTER);
52
53     if (theSignupRequest == null) {
54         throw new IllegalStateException();
55     }
56
57     // explicitly begin a transaction
58     TxCtx.beginTransaction();
59
60     SystemUser newUser = createSystemUserForCafeteriaUser(theSignupRequest);
61     createCafeteriaUser(theSignupRequest, newUser);
62     theSignupRequest = acceptTheSignupRequest(theSignupRequest);
63
64     // explicitly commit the transaction
65     TxCtx.commit();
66
67     return theSignupRequest;
68 }
```

# Persistence

Acesso concorrente

# Problema – Acesso Concorrente

## ■ Exemplo Optimistic

1. Utilizador A inicia a alteração do DishType "Fish" com a descrição "Fish dish" e altera para "Fly-fisch"
2. Utilizador B inicia a alteração do DishType "Fish" com a descrição "Fish dish" e altera para "Cat-fisch"
3. Utilizador A grava as alterações com sucesso
4. Utilizador B tenta gravar as alterações mas não tem sucesso. Pois entretanto o registo já tinha sido alterado pela utilizador A.

## ■ Exemplo Pessimistic

1. Utilizador A inicia a alteração do DishType "Fish" com a descrição "Fish dish" e altera para "Fly-fisch"
2. Utilizador B tenta iniciar a alteração do DishType "Fish" sem sucesso. O registo está bloqueado.
3. Utilizador A grava as alterações com sucesso
4. Utilizador B inicia a alteração do DishType "Fish" com a descrição "Fly-fisch"

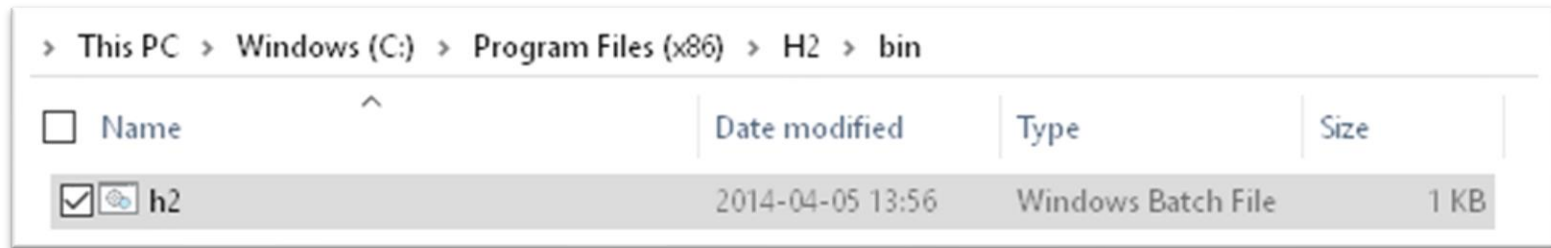
# Solução

- Como implementar a abordagem Optimistic?
  1. Setup: ambiente multi-utilizador
  2. Preparar as classes persistentes para acesso concorrente
  3. Apanhar a exceção de acesso concorrente
  4. Tratamento e propagação da exceção até ao ecrã do utilizador

# 1. Setup servidor

- H2 para modo servidor

1. Arrancar com o H2 localhost



2. Alterar a connection string no ficheiro persistence.xml:

```
<property name="javax.persistence.jdbc.url"
value="jdbc:h2:tcp://localhost/~ecafeteria;" />
```

## 2. Classes persistentes

```
@Entity
public class DishType implements AggregateRoot<String>, Serializable {


    private static final long serialVersionUID = 1L;

    // ORM primary key
    @Id
    @GeneratedValue
    private Long id;
    @Version
    private Long version;

    // business ID
    @Column(unique = true)
    private String acronym;
    private String description;
```

## 2. Classes persistentes

- O atributo version será incrementado automaticamente sempre que exista uma alteração ao registo.
- Será este atributo que permitirá ao JPA perceber que a versão que está a ser gravada está desatualizada.



#	ID	ACRONYM	ACTIVE	DESCRIPTION	VERSION
1		10 vegie	<input checked="" type="checkbox"/>	vegetarian dish	1
2		11 fish	<input checked="" type="checkbox"/>	Fly-fish	2
3		12 meat	<input checked="" type="checkbox"/>	meat dish	1



# 3. Apanhar a exceção

- É necessário apanhar a exceção nos métodos save e delete (classe JpaTransactionalRepository > JpaBaseRepository)
- O JPA lança a exceção OptimisticLockException
  - "...cannot be merged because it has changed or been deleted since it was last read"

```
public <S extends T> S save(final S entity) {  
    try {  
        return entityManager().merge(entity);  
    } catch (final OptimisticLockException ex) {  
        throw new ConcurrencyException(ex);  
    } catch (final PersistenceException ex) {  
        if (ex.getCause() instanceof OptimisticLockException) {  
            throw new ConcurrencyException(ex);  
        }  
        throw new IntegrityViolationException(ex);  
    }  
}
```

## 4. Tratamento e propagação

```
public class ChangeDishTypeUI extends AbstractUI {
    private final ChangeDishTypeController controller = new ChangeDishTypeController();

    @Override
    protected boolean doShow() {
        final Iterable<DishType> dishTypes = this.controller.dishTypes();
        final SelectWidget<DishType> selector = new SelectWidget<>("Dish types:", dishTypes,
            new DishTypePrinter());
        selector.show();
        final DishType theDishType = selector.selectedElement();
        if (theDishType != null) {
            final String newDescription = Console
                .readLine("Enter new description for " + theDishType.description() + ": ");
            try {
                this.controller.changeDishType(theDishType, newDescription);
            } catch (@SuppressWarnings("unused") final ConcurrencyException ex) {
                System.out.println(
                    "WARNING: That entity has already been changed or deleted since you last read
                );
            }
        }
        return false;
    }
}
```

# Demonstração

H2 Console

jdbc:h2:tcp://localhost/~ecafeteri

Run Run Selected Auto complete Clear SQL statement:

SELECT \* FROM DISHTYPE

ABSTRACTBOOKINGEVENT  
BOOKING  
BOOKING\_ABSTRACTBOO  
CAFETERIAUSER  
CARDMOVEMENT  
CREDITRECHARGE  
DISH  
DISHTYPE  
PK  
ACRONYM  
ACTIVE  
DESCRIPTION  
VERSION  
Indexes  
MATERIAL  
MEAL  
ROLEASSIGNMENT  
ROLESET  
ROLESET\_ROLEASSIGNME  
SIGNUPREQUEST  
SYSTEMUSER  
INFORMATION\_SCHEMA  
Sequences  
Users  
H2 1.4.195 (2017-04-23)

SELECT \* FROM DISHTYPE;

PK	ACRONYM	ACTIVE	DESCRIPTION	VERSION
11	vegie	FALSE	vegetarian dish	1
12	fish	TRUE	peixe	3
13	meat	TRUE	meat	6

(3 rows, 18 ms)

Edit

```
Windows PowerShell
11:50:30.077 [main] WARN o.h.orm.connections.pooling - HHH10001002: Using Hiber
nate built-in connection pool (not for production use!)
11:50:30.077 [main] WARN o.h.orm.connections.pooling - HHH10001002: Using Hiber
nate built-in connection pool (not for production use!)
=====
eCafeteria Back Office 5.1.0
(C) 2016 - 2020, ISEP's Professors of EAPLI
=====
11:50:33.507 [main] DEBUG e.f.i.e.i.i.InProcessPublisherDispatcher - NewUserRegi
steredFromSignupWatchDog subscribed to events [class eapli.ecafeteria.cafeteriau
sermanagement.domain.events.NewUserRegisteredFromSignupEvent]
11:50:33.508 [main] DEBUG e.f.i.e.i.i.InProcessPublisherDispatcher - SignupAccep
tedWatchDog subscribed to events [class eapli.ecafeteria.cafeteriausermanagement
.domain.events.SignupAcceptedEvent]

+= Login =====+

Username:
chef
Password:
Password1
=====+

+= eCAFETERIA [ @chef ] =====+

| 1. My account > | 5. Dishes > | 7. Menus > | 8. Reporting Dishes > | 0. Exit |
Please choose an option
5
>> Dishes >
1. Register new Dish Type
2. List all Dish Type
3. Change Dish Type description
4. Activate/Deactivate Dish Type
5. Register new Dish
6. List all Dish
7. Register new Dish (via DTO)
8. List all Dish (via DTO)
9. Activate/Deactivate Dish
10. Change Dish >
0. Return
Please choose an option
```

```
11:52:59.700 [main] DEBUG e.f.i.e.i.i.InProcessPublisherDispatcher - NewUserRegisteredFromSignupWatchDog subscri
11:52:59.704 [main] DEBUG e.f.i.e.i.i.InProcessPublisherDispatcher - SignupAcceptedWatchDog subscribed to events

+= Login =====+

Username:
chef
Password:
Password1
=====+

+= eCAFETERIA [ @chef ] =====+

| 1. My account > | 5. Dishes > | 7. Menus > | 8. Reporting Dishes > | 0. Exit |
Please choose an option
```

# JPA –SQL executado pelo update

```
UPDATE DISHTYPE  
SET DESCRIPTION = 'veggggg',  
    VERSION = 2  
WHERE ID = 99  
    AND VERSION = 1
```

- Além de alterar a descrição, incrementa a versão do registo para 2
- No Where além de pesquisar pela chave(ID), é confirmado que se está a alterar o registo com a versão 1. O que acontece se esta versão já não for 1?