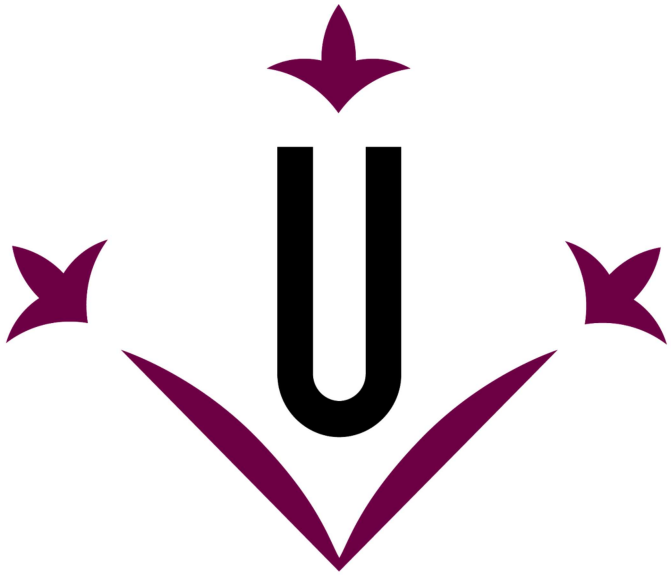
INFORME DE PROVES UNITÀRIES

Data de lliurament: Dia 28 de Desembre del 2025

Autors: Daranuta Zop, Cristian; Esquinas Fernández, Guillermo; Gorga, Sebastian Anthony



Universitat de Lleida

Grau en Enginyeria Informàtica

Enginyeria del Programari

Professorat: Sendin Veloso, Montserrat

CURS: 2025-2026

ÍNDEX

INTRODUCCIÓ

L'objectiu d'aquesta pràctica és implementar i verificar mitjançant proves unitàries una versió simplificada del cas d'ús Supervisar tractament, pertanyent al Sistema Integral Personalitzat de la Història Clínica Electrònica (HCE). Aquest cas d'ús permet al metge revisar i ajustar el tractament d'un pacient, incorporant modificacions a la prescripció mèdica i comptant amb el suport d'una intel·ligència artificial de suport a la presa de decisions.

Per al desenvolupament de la pràctica s'ha seguit un enfocament incremental, implementant progressivament les classes del domini, els serveis col·laboradors i el controlador del cas d'ús. Durant tot el procés s'han aplicat les bones pràctiques de disseny apreses a l'assignatura, posant especial èmfasi en els principis SOLID, els patrons GRASP i la detecció i correcció de possibles code smells mitjançant tècniques de refactoring.

Així mateix, s'ha utilitzat el sistema de control de versions Git amb un repositori remot per facilitar el treball en equip, garantir la traçabilitat dels canvis i assegurar que el codi integrat superés les proves unitàries corresponents.

DECISIONS DE DISSENY

Principis SOLID

Single Responsibility Principle (SRP)
Cada classe del sistema té una responsabilitat única i ben definida. Les classes del paquet data actuen com a classes valor i únicament encapsulen identificadors o dades primitives. Les classes del paquet medicalconsultation modelen els conceptes clínics del domini, mentre que ConsultationTerminal s'encarrega exclusivament d'orquestrar els esdeveniments del cas d'ús,

sense contenir lògica clínica pròpia. Aquesta separació clara facilita la comprensió, manteniment i evolució del sistema.

Open–Closed Principle (OCP)
El disseny permet estendre el comportament del sistema sense modificar el codi existent. L'ús d'interfícies per als serveis externs (com el servei nacional de salut o la IA de suport a la decisió) permet substituir o ampliar implementacions sense afectar les classes del domini ni el controlador del cas d'ús.

Dependency Inversion Principle (DIP)
Les classes de més alt nivell no depenen de classes concretes sinó d'abstraccions. Els serveis externs s'injecten mitjançant setters, fet que redueix l'acoblament, facilita l'ús de dobles de prova i millora la testabilitat del sistema.

Patrons GRASP

Cada classe conté la lògica associada amb la informació que gestiona. Per exemple, MedicalPrescription és la responsable d'afegir, eliminar i modificar línies de prescripció, ja que és qui coneix el conjunt de medicaments prescrits.

La classe ConsultationTerminal actua com a controlador de façana del cas d'ús Supervisar tractament. Aquesta classe és responsable de gestionar els esdeveniments d'entrada del sistema i de coordinar les interaccions entre la interfície d'usuari, el domini i els serveis externs.

Alta Cohesió i Baix Acoblament
Les classes presenten un alt grau de cohesió, ja que els seus mètodes estan directament relacionats amb la responsabilitat principal de la classe. Alhora, l'acoblament entre components es manté baix gràcies a l'ús d'interfícies i a la separació clara entre domini, serveis i controlador.

Les classes presenten mètodes directament relacionats amb el propòsit principal, evitant funcionalitats innecessàries o fora de context. Això es reflecteix especialment en Posology i TakingGuideline, que actuen com a classes de suport amb una estructura clara i cohesionada.

Refactoring aplicat

Durant el desenvolupament s'han aplicat refactoritzacions de manera preventiva amb l'objectiu d'evitar l'aparició de code smells en fases posteriors. Entre les principals decisions destacables es troben:

La reorganització de les classes en paquets segons la seva responsabilitat (data, medicalconsultation, services), millorant la llegibilitat i la navegació del projecte (move class).

La substitució d'estructures de dades menys eficients per col·leccions adequades, com l'ús d'un Map per gestionar les línies de prescripció mèdica, evitant cerques seqüencials innecessàries.

La separació clara entre la lògica de control del cas d'ús i la lògica del domini, evitant que el controlador acumuli responsabilitats que no li corresponen (extract class / separation of concerns).

Aquestes refactoritzacions han contribuït a obtenir un codi més net, extensible i fàcil de provar, alineat amb els criteris de qualitat exigits a la pràctica.

Tests

El testing s'ha realitzat mitjançant classes específiques per cadascuna de les classes que formen el projecte. D'aquesta forma, l'objectiu és testejar el bon funcionament de cadascuna d'elles de forma individualitzada.

Tenint això en compte s'han realitzat comprovacions exhaustives sobre el control i llançament d'excepcions al mateix temps que s'ha buscat garantitzar el bon funcionament de les funcionalitats més essencials de cada classe/mètode.

D'altra banda, la clase `DecisionMakingAIImpl` no implementa una IA real, sino que el que fa és utilitzar un *stub* determinista disenyat per a poder permetre probes unitaries del fluxe de decisió sense dependències externes

Depuració

La depuració del projecte s'ha fet principalment mitjançant l'execució contínua de proves unitàries i la comprovació dels valors d'entrada als mètodes més importants. Això ha permès detectar errors ràpidament i evitar estats incorrectes del sistema.

L'ús d'excepcions ha ajudat a identificar quan no s'estava utilitzant correctament alguna cosa, facilitant la localització de fallades durant el desenvolupament. A més, provar el codi després de cada canvi ha permès corregir errores de manera progresiva.

CONCLUSIÓ FINAL

En aquesta pràctica s"ha desenvolupat una implementació simplificada del cas d"ús Supervisar tractament dins del Sistema Integral de la Història Clínica Electrònica.

El disseny del sistema s'ha basat en una separació clara de responsabilitats, recolzada en els principis SOLID i els patrons GRASP. La diferenciació entre el domini, els serveis externs i el controlador del cas d'ús ha estat clau per millorar la mantenibilitat i facilitar la realització de proves unitàries.

Les proves desenvolupades, recolzades en l'ús de dobles de prova, han permès validar el comportament del sistema en diferents escenaris i comprovar el compliment dels contractes d'operació definits. Finalment, l'ús de Git com a sistema de control de versions ha facilitat el treball en equip i el seguiment de l'evolució del projecte.