Pràctica 4
Probes unitàries

Ferran Aran Domingo
Oriol Agost Batalla

48253731F
48257094N

Oriol Agost Batalla 48257094N
Ferran Aran Domingo 48253731F

Git

En aquesta pràctica se'ns demanava emprar l'eina de control de versions Git, per tal de poder mantenir un històric dels canvis del projecte.

La forma en que hem decidit organitzar-nos ha estat, tal com se'ns recomanava, usant una branca per cada membre del grup i, enlloc de posar el codi en comú a la branca *main*, hem incorporat una branca *develop* que ens permet fer una primera posada en comú de la feina dels dos membres, però sense que sigui el resultat final de la combinació, deixant lloc a refinacions com ara refactors o algun petit canvi en el projecte abans de posar-ho finalment a la branca *main*.

D'aquesta manera aconseguim que tot el codi que estigui a la branca *main* sigui la versió final d'aquest (normalment, tot i que sempre pot haver algun canvi), i tenim un lloc de treball per a combinar la feina feta (la branca *develop*).

Tests

Un altre dels objectius d'aquesta pràctica era aprendre a fer proves unitàries, amb l'ajuda de la llibreria *JUnit*, de la qual hem usat la versió 5.7.

Hem optat per fer que tots els tests que miraven excepcions, que normalment és donaven en invocar al constructor de certa classe amb uns paràmetres equivocats, estiguessin implementats com *default void*, en l'interfície de test. Així, els tests que sí que depenien d'una preparació prèvia podrien ser implementats en la classe amb el mètode *@BeforeEach*, sense que aquest mètode és cridi en el cas de la comprovació de les excepcions.

Hem fet un seguit de diferents test per cada paquet.

Paquet controller

En aquest, només hem declara't una interfície que serà la que ens ajudarà a testar la plataforma unificada. Però per tal de simular el funcionament de les autoritats de certificació i poder fer els tests corresponents, hem declarat tres classes dobles (una per cada mètode d'autenticació) que implementaran la interfície *CertificationAuthorityInterface*.

En aquestes, primerament preparem un ciutadà, ja que hem fet que l'autoritat de certificació que necessita la plataforma unificada necessiti un ciutadà, cosa que ens ajudarà molt a l'hora de fer els tests. A aquest ciutadà li posem tots els valors que testarem més tard, com venen a ser el PIN, el número de telèfon, el NIF, etc. Finalment el passem com a paràmetre al constructor corresponent.

Cal destacar que per tal de testar aquelles funcionalitats que emulen alguna acció, com que el fet d'emular quelcom l'hem representat fent un *print* que mostra un missatge indicant l'acció emulada, el que mirarem al test es si s'ha mostrat l'output que volem per la sortida estàndard.

En la interfície que engloba les 3 classes de testing d'aquest apartat em implementat com a tests *default* tots aquells que comproven que es llencin les excepcions en els casos d'error corresponents als mètodes independents de l'autoritat d'autenticació escollida, i hem deixat a càrrec de les classes que la implementen els tests que comproven els mètodes que s'encarreguen d'obtenir els dos possibles certificats, així com el que comprova el mètode que s'encarrega de seleccionar un mètode d'autenticació.

La resta de testos no els hem posat a la interfície per respectar el principi de segregació d'interfícies i aconseguir que tots els mètodes que una classe de testing haurà d'implementar d'aquesta interfície tinguin sentit i siguin aplicables.

La resta de funcionalitats que es testen són:

- *enterNIFandPINObt()*: Es comprova tant el cas en que es fa correctament com els casos en que el NIF no esta registrat, la data de validació no és correcta i el ciutadà no te cap telèfon registrat.
- *enterPIN()*: Comprovem que funcioni tant quan el PIN és correcte com quan no ho és.
- *enterCred()*: Comprovarem que funcioni essent que l'usuari ha escollit el mètode reforçat o no, i també en cadascun dels possibles errors: NIF no registrat, telèfon no registrat, credencials incorrectes.
- *selectCertificate()*: Comprovarem el cas d'exit on s'escull un determinat certificat.
- *enterPassw()*: Comprovarem també el cas d'èxit on s'introdueix la contrasenya correcta i el cas en que aquesta sigui invàlida.

Paquet *data*

En cada un dels tests de les classes d'aquest paquet, hem testat el mètode get, que havia de retornar la principal estructura de dades de la classe i, en totes les classes, un constructor amb Null per paràmetre retornava una *NullPointerException*.

A més a més, també hem testat en cada una de les seves classes les possibles restriccions que tenien, essent les següents:

- *AccredNumb*: La seva mida ha de ser exactament 9 i cap lletra.
- *DocPath*: Al menys ha de tenir un caràcter, no pot ser un *String* buit.
- *Nif*: Ha de tenir exactament 8 números i una lletra majúscula.
- *Password*: La seva mida ha de ser d'almenys 4 i ha de tenir mínim una lletra, un número i un caràcter especial, essent aquests qualsevol caràcter no numèric, alfabètic i no "barra espaiadora".
- *PINCode*: La seva mida ha de ser d'exactament tres, i tots els seus caràcters han de ser números.

Paquet *publicadministration*

En aquests, a part de mirar si els constructors són *null* i llençar l'excepció *NullPointerException* en conseqüència, mire'm:

- *LaboralLifeDoc*: És comprova que donat un nif i una col·lecció de períodes de cotització, aquest prenguin el valor que toquen en l'atribut corresponent del document.
- *MemberAccreditationDoc*: És comprova que donat un nif *Nif* i un nombre d'acreditació *AccredNum*, aquests prenen el valor correcte en l'atribut corresponent del document.
- *PDFDocument*: És comprova que donat en un document PDF, podem aconseguir el seu path, el fitxer, pode'm moure'l (comprovant que els *paths* del fitxer canvien) i podem obrir-lo (ho comprovem obrint-lo en pantalla).
- *QuotePeriod*: És comprova que els *getters* dels atributs funcioni correctament, apart de comprovar que els nombres no siguin negatius ni 0, també ens assegurem que el nombre de dies més la data inicial no superen la data actual.
- *QuotePeriodColl*: Que donada una *QuotePeriodColl*, les seves longituds siguin les correctes en afegir o sostreure i que estiguin ordenats per data, tal i com se'ns demana.

Oriol Agost Batalla 48257094N
Ferran Aran Domingo 48253731F

Decisions de disseny

Classe *Main*

Degut als comentaris del fòrum, hem pensat que seria una bona idea implementar una classe *Main* que emuli la funcionalitat de la interfície gràfica, d'aquesta forma podem determinar un ordre d'execució, aconseguint per exemple, que un ciutadà no pugui cercar una paraula clau sense haver desplegat primer el cercador.

D'aquesta forma, apart de poder representar el funcionament de la GUI (executant aquesta classe i seguint les indicacions del terminal) també ens serveix a nosaltres per a ajudar-nos a entendre la estructura del projecte, veient les conseqüències dels canvis que fem repercutits en el que veuria el ciutadà.

Setters de *UnifiedPlatform*

El constructor de la classe *UnifiedPlatform* té una sèrie de mètodes de preparació (*SetUp*) que hem implementat al voltant del Principi Open-Closed (O de SOLID), ja que permetrien de forma molt fàcil i sistemàtica afegir tant nous serveis, com AAPPs, com mètodes d'autenticació al sistema.

Aconseguint així que el codi estigui obert a extensions però tancat a la modificacions.

Classe *Citizen*

A l'hora d'implementar/testar la classe *UnifiedPlatform* ens vam adonar que hi havia moltes dependències amb dades del ciutadà (com el NIF, el número de telèfon, ...), per tant vam creure necessari implementar una classe *Citizen* que tingues aquesta responsabilitat.

A més, així es mostra en el diagrama de classes, raó de més per considerar la seva implementació. Un cop va estar implementada va quedar clara la seva necessitat, ja que l'estructura tant dels tests com de la classe controladora va fer-se immediatament més clara i visible.

Per resoldre aquesta situació ens vam basar en el Principi d'Única Responsabilitat (S de SOLID).

Classes dobles

Tal com se'ns ha indicat, per tal de poder testar el funcionament de la classe *UnifiedPlatform* hem implementat classes dobles tant de la AAPP Seguretat Social, com dels diferents mètodes d'autenticació.

Per tal de fer que el procés de testeig amb classes dobles fos àgil, hem emprat una instància de la interfície que implementen totes les classes dobles d'autoritat de certificació. De manera que cridem els mètodes d'autenticació sobre aquesta instància, i no sobre una determinada autoritat, aconseguint així que la classe *UnifiedPlatform* tingui la mateixa estructura per diferents autoritats de certificació.

Per tant, per tal de discernir entre el cas en que s'ha triat una autoritat o l'altra, utilitzem dues injeccions de tipus; una per l'AAPP i l'altra per l'autoritat de certificació.

Certificat Digital

Per tal d'emular el xifratge i des xifratge amb les claus públiques i privades, el que hem fet ha estat determinar, pel be de la implementació del projecte, que les claus publiques i privades tinguessin el mateix valor numèric.

Per tant ara xifrar consisteix en convertir l'*String* en un *byte[]* i als valors dels bytes sumar el valor de la clau publica, finalment reconvertint el *byte[]* en *String* un altre cop (però ara amb uns caràcters totalment diferents).

Per tal de desxifrar farem els passos anteriors a la inversa però en lloc de sumar el valor de la clau pública, restarem el de la clau privada (què són el mateix), aconseguint així els valors del *byte[]* inicial i podent convertir aquest en l'*String* original.

Comentaris

Ens agradaria destacar que en determinades situacions en que ens hem anat trobant al llarg del desenvolupament del projecte, hem hagut d'escollir entre respectar l'estructura que se'ns proporciona o bé introduir un canvi que tindria repercussió en aquesta.

És aquest el cas de la interfície *CertificationAuthorityInterface*, on detectem un *code smell* de la categoria dels *object-oriented abusers*, concretament es tracta d'una situació de *refused bequest*, provocada pel fet que en la interfície apareixen els mètodes que usaran les diferents autoritats, però cadascuna d'elles només en donarà ús a uns en concret.

Provocant així que en cadascuna de les classes que representaran a les autoritats de certificació rebutgin part del llegat de la seva superclasse (o interfície en aquest cas), violant així el Principi de Segregació d'Interfícies (I de SOLID).