Aula de programació interactiva

Arman Pipoyan Paronyan

Resum– L'objectiu d'aquest projecte és augmentar les funcionalitats d'una eina de suport docent integrant diferents tecnologies existents com GitHub, Jupyter o Moodle i afegir altres funcionalitats que no requereixen mòduls externs. El resultat obtingut és una eina desplegada a un servidor del centre que permet als docents crear problemes i monitorar les solucions que van donant els estudiants durant les sessions de classe, amb la possibilitat d'interactuar amb ells de manera online modificant el seu codi. Tot això amb la possibilitat d'utilitzar GitHub per indicar arxius inicials d'un problema, per afegir fitxers mentre s'està resolent un exercici o per guardar el progrés a un repositori privat.

Paraules clau– Eina de suport docent, programació online, docència online, desenvolupament de software

Abstract– The goal of this project is to increase the functionalities of a teaching support tool by integrating different existing technologies such as GitHub, Jupyter or Moodle and adding other functionalities that do not require external modules. The result obtained is a tool deployed on a server at the center that allows teachers to create problems and monitor the solutions given by students during class sessions, with the possibility of interacting with them online by modifying their code. All of this with the possibility of using GitHub to indicate initial problem files, to add files while solving an exercise or to save the progress in a private repository.

Keywords– Teaching support tool, online programming, online teaching, software development

1 Introducció

Per aprendre a programar s'ha de programar, això és el que diuen tots. Però també és important equivocar-se i aprendre dels errors que es cometen. Una bona manera de fer-ho, i més si s'està a una classe, és preguntant a algú que sàpiga molt del tema, que a les aules acostuma a ser el professor. Gran part d'aquests errors comesos són comuns, per tant, tenir la possibilitat d'explicar la solució d'una errata a tot el grup tenint un exemple real al que assenyalar resulta molt útil, ja que alguns conceptes són molt difícils de deixar clar sense pràctica. Tot i ser una cosa tan important és una tasca que es dificulta quan la quantitat d'estudiants és molt gran i es disposa d'un temps limitat.

Un altra problema que ens trobem és que no tots tenen les eines necessàries per poder programar, ja que alguns programes tenen uns requeriments mínims una mica elevats i no totes les màquines les poden executar. Per tant, poder programar en un navegador i que el codi s'executi a una màquina remota fa que tots els usuaris parteixin del mateix entorn de treball.

Per resoldre aquest problemes neix aquesta eina de suport a la docència que permet als estudiants escriure codi que pot ser revisat pel professor que està present a l'aula sense la necessitat de desplaçar-se a la seva taula i, a més, executar-ho remotament per poder veure el resultat de l'execució.

Cal tenir en compte que la primera versió d'aquest projecte es va desenvolupar durant la pandèmia del COVID-19 i que un dels objectius que es perseguia era el de disminuir el contacte entre persones i ajudar a que, si s'estava a l'aula les distàncies mínimes de seguretat es mantinguessin. Com actualment les restriccions han variat aquests objectius ja no tenen tant de pes com abans, però s'han tingut en compte pel desenvolupament d'aquest projecte.

El que s'explicarà en aquest document són els canvis

• E-mail de contacte: arman.pipoyan26@gmail.com
• Menció realitzada: Enginyeria del Software
• Treball tutoritzat per: Ernest Valveny (Computació)
• Curs 2021/22

introduïts tant a nivell de funcionalitats com a nivell visual i el procés que s'ha seguit per tal de poder-los implementar.

2 Estat inicial del projecte

S'ha hagut d'actualitzar una eina desenvolupada en un projecte anterior [1] que permetia a professors interactuar, ja fos editant o només mirant, amb estudiants que estiguessin fent problemes de programació que havien proposat.

L'eina permetia als mestres agregar assignatures i dins d'aquestes afegir, editar, amagar i eliminar problemes. També tenien la possibilitat de mirar, editar i executar l'última solució guardada de qualsevol exercici dels alumnes.

Els alumnes tenien menys permisos, només podien resoldre exercicis eliminant, afegit o modificant els fitxers que el professor establia en crear l'activitat i, a més, podien executar el codi per veure el resultat.

Quant al registre de nous usuari, els alumnes podien crear comptes nous sense cap dificultat, però els professors necessitaven una invitació per part d'altres membres de l'equip docent per tal d'aconseguir el permisos necessaris en crear-se el compte.

A la figura 1 podem veure un diagrama de casos d'ús de les funcionalitats que oferia aquesta versió de l'eina.

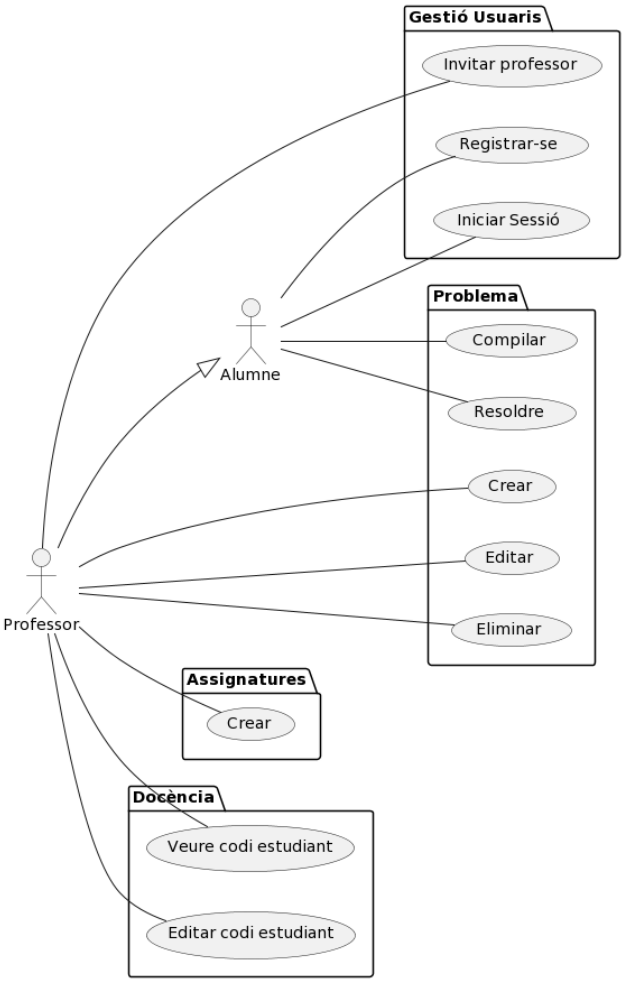


Fig. 1: Casos d'ús implementats en la versió original de l'eina.

3 Objectius

En aquest projecte s'han introduït canvis a la web que ajuden tant als professors com als alumnes a tenir més facilitats en utilitzar l'aplicació. A la figura 2 tenim representades les funcionalitats que s'han afegit i que s'explicaran en detall a continuació. A la figura 14 dels annexos tenim una visió global del que ens permet fer l'eina.

Cal esmentar que aquests canvis no modifiquen l'objectiu principal de la primera versió que s'havia desenvolupat de la web, que era la creació d'una eina perquè els professors poguessin pujar problemes i que els alumnes els poguessin resoldre dins de la web, tenint sempre l'opció de poder compartir el codi amb el professor de manera senzilla perquè aquest ho pogués corregir i donar *feedback*.

3.1 Refactorització del codi

Aquesta tasca s'ha fet amb la intenció de familiaritzar-se amb el codi i el funcionament general de la web i per resoldre problemes d'estructura que tenia.

3.2 Creació de sessions de classe

L'objectiu d'aquesta funcionalitat és permetre als professors crear agrupacions d'exercicis de manera temporal per tal de comentar-les durant classes presencials. Durant aquestes, els alumnes han d'accedir a la sessió creada pel seu professor per tal de poder tenir la seva supervisió mentre resolen les activitats proposades, ja que el professor només serà capaç de veure les respostes dels estudiants de la seva sessió.

A continuació tenim una breu descripció dels casos d'ús d'aquesta funcionalitat que s'han completat per considerar-la acabada:

- **Iniciar sessions de classe**: Un professor ha de ser capaç de començar una sessió de classe indicant un nom i un conjunt de problemes, que han de pertànyer a la mateixa assignatura que la sessió.

- **Veure sessions actives**: Tant professors com alumnes han de poder veure les sessions actives d'una assignatura.

- **Accedir a sessions**: Els usuaris han de poder accedir a una sessió i resoldre els problemes d'aquesta comptant amb l'ajuda del professor.

  S'ha decidit no implementar cap mesura de prevenció per evitar que els alumnes entrin a sessions que no els pertoca, ja que s'assumeix que no estan interessats a fer-ho.

- **Veure estudiants de la sessió**: Els professors han de tenir un llistat amb tots els membres de la seva sessió i, a més, han de poder accedir a les seves propostes de solució i editar-les si era necessari.

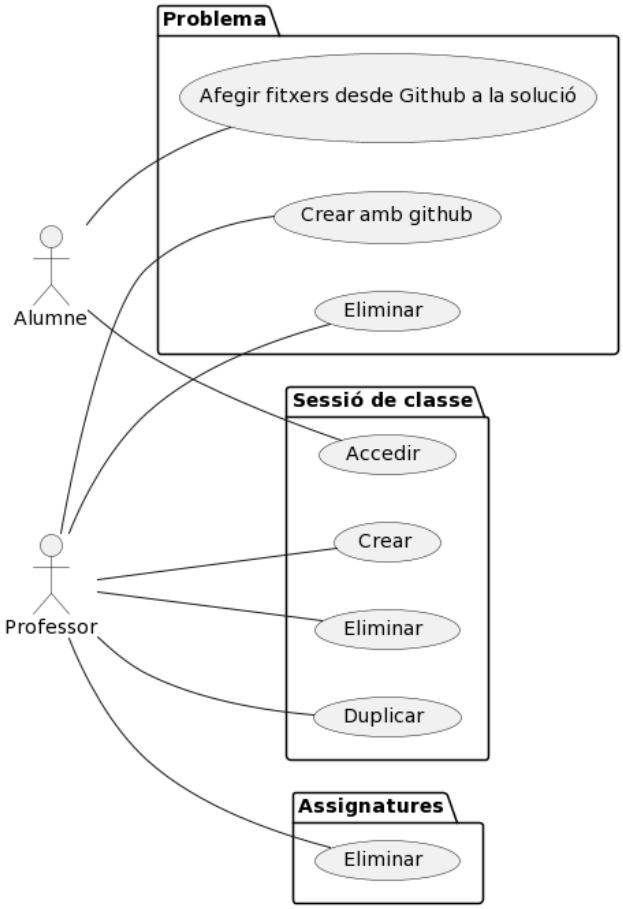


Fig. 2: Casos d'ús a implementar a l'eina.

- **Tancar sessions de classe**: Els professors han de ser capaços d'esborrar les sessions que considerin acabades.

3.3 Integració amb GitHub

Els objectius d'aquesta integració són:

Crear exercicis: Permetre als professors crear problemes utilitzant directament carpetes del seu repositori privat. Els problemes no poden tenir subcarpetes, per tant, totes les carpetes del repositori s'ignoraran i en cas de trobar un fitxer *readme*, el seu contingut s'utilitzarà com descripció de l'exercici.

Guardar solucions: Facilitar als usuari poder guardar el treball que fan durant les sessions als seus repositoris per tal de poder accedir a aquests en un futur sense obrir l'eina.

Afegir fitxers a problemes: Possibilitar poder afegir fitxers des de GitHub indicant l'enllaç a una carpeta o a un fitxer en concret.

3.4 Integració amb Caronte

A les assignatures de programació els professors acostumen a proposar exercicis avaluables a Caronte, el Moodle dedicat a les enginyeries.

La finalitat d'aquesta integració és permetre al professor vincular les activitats de Caronte amb problemes de l'eina, permetent als alumnes fer entregues des de la pàgina web sense la necessitat d'haver de descarregar el codi i tornar a pujar-ho al Moodle.

3.5 Integració amb Jupyter Notebook

A algunes assignatures s'utilitza l'entorn web Jupyter Notebook, que permet crear documents amb cel·les executables que poden contenir codi, text o imatges [2]. Com aquests fitxers es guarden en un format concret, es necessita un editor especial per veure i modificar el contingut d'aquests.

Això és el que es volia aconseguir amb aquesta modificació: permetre als alumnes obrir i modificar aquests fitxers des de l'eina sense necessitat d'instal·lar o obrir programari addicional.

3.6 Redisseny de la pàgina web

La finalitat d'aquesta tasca és unificar el disseny del que ja estava fet de la pàgina amb els canvis nous que s'han introduït i, en cas que fos necessari, fer un redisseny de tota la pàgina web.

Es farà un cop estiguin totes les funcionalitats implementades (principalment les que requerien afegir o modificar vistes existents). Això permetrà acabar les funcionalitats amb més rapidesa i tenir un disseny més uniforme.

3.7 Desplegar l'eina a un servidor

L'objectiu de fer el desplegament del projecte a una de les màquines del centre docent era provar en un entorn real l'eina per tal de trobar millores que es podrien afegir o errors que es puguin arreglar.

4 METODOLOGIA

Per desenvolupar el projecte s'ha decidit utilitzar una metodologia Agile [3], emprant Trello [4] per poder organitzar les tasques i tenir una visió general del que s'ha fet i el que queda per finalitzar. La durada dels *sprints* serà generalment d'una setmana, i s'intentarà que durant aquesta franja de temps es tingui una porció del mòdul que s'estigui implementant totalment finalitzada, assegurant així que es treballa amb constància.

Per tal de tenir sempre una versió estable del codi s'ha decidit utilitzar GitHub. Es tindran dues branques actives: la principal, on estaria la versió estable del codi i la branca de desenvolupament, on s'implementaran les noves funcionalitats. Cada vegada que un mòdul estigui acabat i testejat els canvis es pujaran a la branca principal i es continuarà treballant a la segona branca.

5 PLANIFICACIÓ

La planificació ha sofert diferents canvis durant el pas del temps perquè algunes tasques van requerir més temps del que s'havia planificat i algunes van haver-se de cancel·lar.

A la planificació inicial es van definir 4 fases, com podem veure a la figura 3:

- La primera d'aquestes fases era la de planificació, que consistia en analitzar els requeriments i les subtasques i decidir si eren viables.
- La següent fase era la d'implementació, que tenia tantes subfases com funcionalitats es volien desenvolupar, però aquestes no estaven desglossades.
- La tercera fase era el testatge general de la web. Tot i que cada fase de desenvolupament contemplava també el temps necessari per provar que funcionava correctament, es va decidir dedicar aquest temps en comprovar que l'eina funcionava tal com s'esperava.
- Durant la quarta es redactaria la memòria final i es començaria a preparar la presentació.

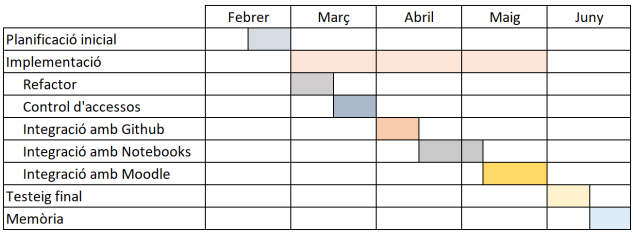


Fig. 3: Planificació inicial del projecte.

Es van anar aplicant diferents canvis a aquesta planificació, resultant en el que podem veure a la figura 4. Els canvis aplicats en ordre cronològic són els següents:

- La tasca de refactorització es va allargar una setmana per falta de temps, fent que la següent tasca es retardés una setmana.
- El nom de la tasca 'Control d'accessos' es va canviar a 'Sessions de classe', ja que el primer era pot explicatiu. A més, es va desglossar tant aquesta tasca com la de la integració amb GitHub.
- Es va afegir la tasca de redisseny de la pàgina web. Aquesta es faria abans de començar la redacció de la memòria final i duraria només una setmana.
- Es va anar desglossant la tasca de la integració amb Jupyter Notebook segons s'anaven fent versions i proves.
- En investigar la integració amb Caronte es van trobar problemes que no permetien implementar les funcionalitats desitjades, fent que es pogués dedicar més temps a altres tasques. Es van allargar les tasques de redisseny i de testatge final una setmana.
- Es va afegir la tasca de fer un desplegament del codi a un servidor de la UAB. Aquest duraria dues setmanes i es faria juntament amb el testatge, abans de començar la memòria.

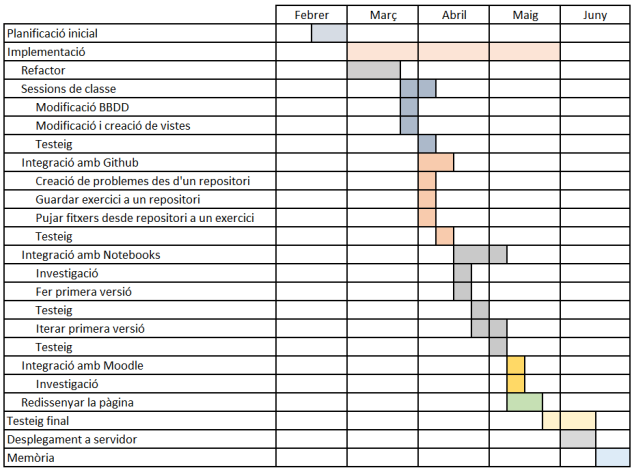


Fig. 4: Planificació final del projecte.

6 DESENVOLUPAMENT

6.1 Refactorització

El que s'ha fet, principalment, durant la refactorització ha sigut:

1. Esborrar mètodes i variables que no s'utilitzaven.
2. Traduir tot el codi a un únic idioma (l'anglès), ja que s'utilitzava tant l'anglès, com el castellà com el català per definir mètodes o variables.
3. Afegir comentaris al codi.
4. Introduir constants per facilitar la lectura del codi.
5. Modificar i estandarditzar el nom de variables, mètodes o fitxers que no eren autoexplicatius i que no sempre seguien el mateix format.
6. Dividir els fitxers molt grans en fitxers petits.
7. Utilitzar un analitzador estàtic de codi, concretament SonarLint [5], per millorar l'estil de tots els fitxers.

S'han modificat fragments de codi difícils d'entendre com el que es pot veure a la primera imatge de la figura 5. En aquest cas, el que s'ha fet és moure el codi que es trobava a un fitxer '.php' a un fitxer '.js' per tal de no barrejar scripts amb HTML i PHP, a més, s'han afegit comentaris i s'ha modificat el nom de les variables per ajudar a entendre el codi. El resultat del refactor ho veiem a la segona imatge de la figura 5.

6.2 Implementació de les sessions de classe

6.2.1 Modificació de la BBDD

Per satisfer aquest objectiu s'ha hagut de modificar la base de dades afegit les taules 'session' i 'session_problems' on la primera serveix per guardar el seu nom, el professor que l'ha creat i l'assignatura a la que pertany i la segona taula serveix per relacionar una sessió amb els problemes que agrupa. A més, s'ha hagut de modificar la taula 'student' afegint la clau forana 'session_id', que serveix per indicar que un estudiant només pot pertànyer a una única sessió.

A la figura 13 dels annexos podem veure com es relacionen totes les entitats que existeixen a l'eina.

```php
//Estudiante nunca podra estar en modo reiterativo por ende no se puede hacer asi
if (isset($_GET["reiteratiu"])) {
    if ($_GET["reiteratiu"]==1) {
    if ($_SESSION['tipo']==0) {?>
        <script>
            //editor.setReadOnly(true);
            var myVar = setInterval(save, 4000);
        </script>
    <?php }
    }elseif ($_GET["reiteratiu"]==2) { ?>
        <script>
            editor.setReadOnly(true);
        </script>
    <?php }
}?>
```

(a) Codi sense refactoritzar.

```php
// Set the auto check options depending on the user and his actions
// User type 1 is student and 0 professor
if (userType == 1) {
    setInterval(checkChanges, timeout: 3000);
} else if (userType == 0) {
    viewMode = urlParams.get('view-mode');
    // View mode 1 is edit mode and 2 read only
    if (viewMode == "1") {
        setInterval(save, timeout: 4000);
    } else if (viewMode == "2") {
        editor.setReadOnly(true);
    }
}
```

(b) Codi refactoritzat.

Fig. 5: Comparació abans i després del refactor.

6.2.2 Actualització de les vistes existents

Al llistat d'assignatures s'han fet dos canvis:

- S'ha afegit un botó que porta al formulari de creació de sessions.
- S'ha afegit a cada ítem del llistat un botó que porta a la llista de sessions actives, que és visible només si hi ha alguna activa.

Tal com podem veure a la figura 6.

D'altra banda, s'ha fet que el professor no vegi a tots els estudiants que estan resolent un exercici, sinó que només veu als alumnes que són a la seva sessió, fent que sigui més fàcil trobar-los.

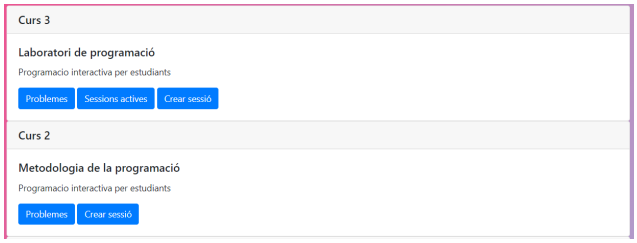


Fig. 6: Ítems del llistat d'assignatures.

6.2.3 Creació de vistes noves

S'han hagut de crear dues vistes noves:

- Un llistat de sessions els ítems del qual només contenen el nom i un botó per esborrar la sessió, però no modifica les solucions dels problemes que els alumnes estaven fent.
- Un formulari de creació de sessions que compta només amb dos camps: el camp del nom i un selector que permet indicar quins exercicis s'han de realitzar durant la sessió, com podem veure a la figura 7.

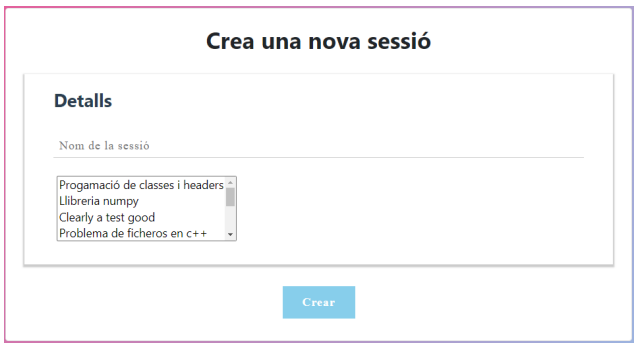


Fig. 7: Formulari per crear una nova sessió.

6.3 Integració amb GitHub

Per implementar aquesta funcionalitat es va utilitzar la REST API de GitHub [6], concretament el wrapper escrit amb PHP [7], que va facilitar la integració de l'API i el tractament d'errors, ja que no s'havia de mirar la resposta es rebia, sinó que en cas d'error saltaven excepcions, que eren més fàcils de tractar.

6.3.1 Configuració prèvia

Abans de començar s'ha hagut de crear una aplicació [8] de GitHub per tal de poder utilitzar les seves APIs. D'aquesta aplicació només es necessiten l'identificador i el codi secret de l'aplicació per fer les primers consultes a GitHub.

6.3.2 Autenticació de l'usuari

Per obtenir el token d'accés de l'usuari de GitHub s'havien d'utilitzar les claus de l'aplicació. El flux per fer el login és el següent:

1. Es fa una redirecció a GitHub on l'usuari s'ha d'identificar amb el seu compte.
2. Es torna a la pàgina on estava l'usuari abans d'intentar autenticar-se amb un codi especial a la URL que l'usuari no veu.
3. Amb el codi i les claus de l'aplicació es fa una petició a GitHub per obtenir l'*access token* de l'usuari amb el que es podran fer les altres peticions.

6.3.3 Descarregar fitxers i carpetes

Per tal de descarregar fitxers primerament s'ha de tenir el token d'accés de l'usuari i una URL que apunti a un objecte que estigui a un repositori de GitHub, ja sigui un fitxer o una carpeta.

Amb aquestes dues dades podem llançar una petició a GitHub per recuperar la informació del fitxer (o fitxers). Dins d'aquesta informació tenim el seu contingut i nom, que és el que utilitzem per crear un fitxer a la màquina a la

carpeta on toqui, dins d'un problema o dins de la solució d'un estudiant. A la figura 8 podem veure el flux que es segueix al descarregar fitxers.

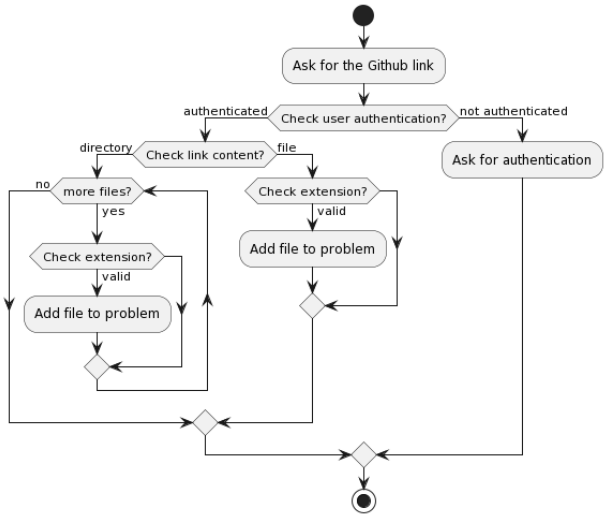


Fig. 8: Diagrama de flux de com es descarreguen fitxers des de GitHub.

6.3.4 Pujar fitxers i carpetes

Per poder pujar fitxers els requeriments són els mateixos que per descarregar fitxers: el token d'accés de l'usuari i la URL a un repositori o carpeta de GitHub.

Al pujar fitxers o carpetes a GitHub el que es fa és crear una carpeta a la ubicació que s'ha indicat amb la URL i fer un *commit* per cadascun dels fitxers. No es fa un *commit* amb tots el canvis perquè la API només permet fer *commits* amb un únic fitxer.

6.3.5 Vistes modificades

Al llistat de problemes s'ha afegit un botó que permet crear problemes utilitzant GitHub. A aquest formulari es demana principalment el nom del problema i el link a la carpeta contenidora.

A la vista de l'editor s'ha afegit tant l'opció de pujar la solució a GitHub com el de descarregar fitxers. Quan es prem un dels botons es demana el link a un repositori.

6.4 Integració amb Jupyter Notebooks

Per implementar aquesta funcionalitat s'havia d'executar un servidor de Jupyter, ja fos en local o utilitzant un servidor remot, que servís els fitxers que demanava l'usuari i mostrar-los utilitzant un iframe d'HTML.

6.4.1 Idea inicial

Durant els primers dies es va estar fent una tasca d'investigació per decidir si era viable o no, i es va acabar concloent que sí que ho era.

La idea que es tenia abans de començar, i la que es va acabar utilitzant, era crear un iframe d'HTML (una pàgina encastada) a la vista de l'editor que substituís l'editor per defecte que hi havia.

6.4.2 Primera versió

A la primera versió es va decidir entre utilitzar un servidor extern com Binder [9] o un servidor Jupyter que servís els fitxers '.ipynb' per tal de crear el contingut de l'iframe. Es va decidir fer servir el servidor extern, ja que aquest oferia un entorn totalment aïllat a cada estudiant on aquest podria executar qualsevol comanda disponible de Python (com per exemple intentar esborrar tots els fitxers de l'entorn). Desafortunadament, aquesta idea no es va poder dur a terme perquè Binder, a dia 20 de juny de 2022, no permet encastar els entorns que genera en altres pàgines web.

6.4.3 Segona versió

Per la segona versió es va crear un servidor de Jupyter que servia els fitxers accedint a un URL local. Aquest servidor s'executava a la carpeta on estaven totes les soluciones dels alumnes fent que amb l'URL no es pogués accedir als fitxers del sistema.

Tenint el servidor en local els estudiants podien visualitzar i interactuar amb el Notebook com si ho tinguessin a la seva pròpia màquina , com podem veure a la figura 9. Però això comportava un problema: tenien accés il·limitat a qualsevol fitxer de la màquina on s'executava el servidor si executaven comandes de Python.

Com no es va trobar la manera de crear un *pre-execute hook* (una acció que s'executés abans de que el codi de l'estudiant fos interpretat) per evitar que s'executessin línies de codi malicioses, es va haver de modificar com estava plantejat el servidor Jupyter.

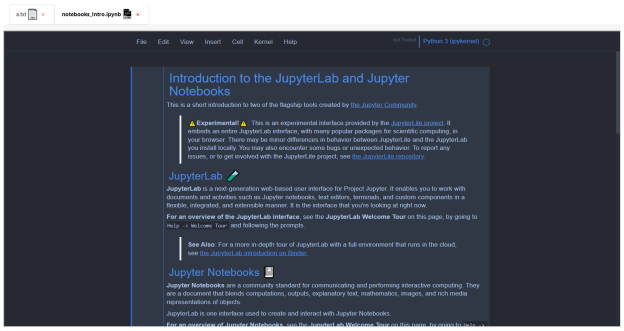


Fig. 9: Notebook visualitzat a la pàgina web.

6.4.4 Tercera versió

A la tercera iteració es va decidir utilitzar un contenidor de Docker que contingués el servidor de Jupyter. Això permetia aïllar el servidor de Jupyter de la màquina física on s'estava executant el codi.

Aquest contenidor tenia enllaçada la carpeta de solucions de la màquina contenidora amb la seva carpeta *home* i se servien els fitxers d'aquesta carpeta. Això feia que amb l'URL només es pogués accedir a les solucions i en cas que un estudiant intentés executar codi maliciós no afectés la

màquina real.

Amb aquest canvi els estudiants només tindrien accés al contenidor i la màquina física no podria ser modificada. Però això encara no evitava que els alumnes poguessin accedir als fitxers dels altres fent servir comandes Python.

6.4.5 Quarta versió

A la quarta i última versió el que s'ha decidit fer és el següent:

1. Quan un estudiant entra a un problema de Jupyter es crea un contenidor per ell que només contindrà les solucions dels problemes que ell ha resolt. Aquest contenidor s'assignarà a un port de la màquina física i, tant el port com l'identificador del contenidor es guardaran a la sessió de l'usuari.

2. Quan l'estudiant surt de la pàgina s'utilitza l'identificador que s'ha guardat i s'esborra tant el contenidor com les dades de la sessió relacionades amb el contenidor.

Això permet als alumnes utilitzar qualsevol comanda al Notebook i assegurar que la màquina física estarà protegida contra estudiants que vulguin executar codi maliciós.

S'han implementat algunes millores més:

- S'ha tingut en compte que un estudiant pot tenir més d'una finestra activa que requereixi un servidor de Jupyter. Per tant, s'ha fet que el contenidor sigui un *singleton*. És a dir, a la sessió es guarda quants problemes de Jupyter l'usuari té oberts i fins que el comptador no arriba a 0, aquest no s'esborra.

- S'ha tingut en compte que a cada contenidor se l'ha d'escoltar per un port diferent. Com els ports d'usuari van del 1024 al 49151 i el màxim d'alumnes que estaran utilitzant l'eina alhora serà 40, el que es farà és escollir un port aleatori i mirar si hi cap contenidor utilitzant aquell port, en cas de col·lisió es tornarà a escollir un port aleatori i a tornar a fer la comprovació. Com el nombre de ports disponibles és molt superior als que es poden arribar a ocupar simultàniament no s'ha trobat necessària la utilització de cap sistema complex de gestió de ports.

6.5 Integració amb Caronte

Com s'ha esmentat en l'apartat 3.4, a les assignatures de programació s'acostuma a utilitzar Caronte per crear exercicis avaluables de programació. Aquesta funcionalitat no la proporciona Moodle, sinó un *plug-in* d'aquest que es diu *Virtual Programming Lab (VPL)*. Per tant, per implementar aquesta funcionalitat s'havia d'investigar com funcionava aquest mòdul per esbrinar si oferia APIs públiques que permetessin pujar fitxers i fer un lliurament.

6.5.1 Investigació

Analitzant la informació disponible de *VPL* i també el codi font d'aquest [10] per trobar coses que no estiguessin especificades a la seva documentació s'ha conclòs que és impossible pels següents motius:

- S'ha de tenir accés a Caronte per poder configurar els serveis necessaris per fer consultes a aquest des de l'eina. Una mala utilització d'aquests permisos representaria una bretxa de seguretat si s'activen serveis sense configurar-los i provar-los apropiadament. Aquesta tasca requereix utilitzar molt temps del projecte resoldre tasques administratives que no estan relacionades amb aquest projecte.

- Encara que activem i configurem els serveis correctament hi ha un problema més. Revisant el codi disponible VPL s'ha conclòs que aquest és un mòdul autocontingut on els lliuraments de fitxers es fan cridant a funcions que no estan publicades a cap API.

6.6 Redisseny de la pàgina web

6.6.1 Paleta de colors foscos i clars

Aquesta és una funcionalitat que molts IDE tenen i que agrada molt als desenvolupadors perquè no tots tenen la mateixa preferència respecte a la paleta de colors a utilitzar.

La preferència seleccionada per l'usuari es guarda en la seva sessió, és a dir, quan aquest tanqui sessió la paleta de colors tornarà a ser la per defecte, la de colors clars. Com es vol donar la possibilitat als usuaris no registrats de modificar la paleta de colors, s'ha fet que si no es té una sessió iniciada les preferències es guardin al navegador. És molt importat que es guardi perquè sinó, en canviar de vista es tornaria a utilitzar la paleta per defecte.

A la segona imatge de la figura 10 podem veure una llista redissenyada que fa servir colors clars i a la segona imatge de la figura 11 podem veure un formulari utilitzant colors foscos.

6.6.2 Llistats

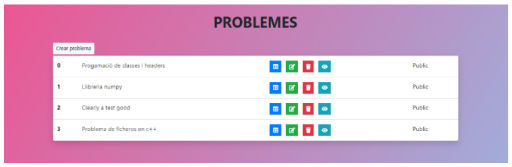
S'ha creat una vista genèrica parametritzable d'un llistat per tal de poder mantenir una consistència entre les vistes i facilitar i mecanitzar la creació de noves vistes d'aquest tipus (vegeu figura 10). El funcionament d'aquesta vista és el següent:

1. Quan l'usuari accedeix a una vista, es crida al controlador d'aquesta.

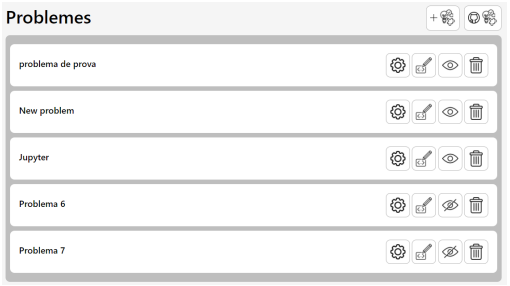
2. El controlador s'encarrega de fer consultes a la BBDD per obtenir els objectes que s'han de mostrar.

3. Amb les dades dels objectes es genera un mapa *hash* que conté: el títol de la pàgina, una llista de mapes *hash* addicional que conté les accions visibles només pel professor (com el botó per crear objectes d'aquella vista) i, per últim, una llista de mapes *hash* més on està la informació de cada ítem (principalment títol, descripció i llista d'accions de l'element).

   Opcionalment, es pot afegir un camp més al mapa *hash* de la pàgina per crear finestres emergents indicant només el títol i el contingut d'aquest (que podia ser un conjunt de camps de formulari o un text).



(a) Abans del redisseny.



(b) Després del redisseny.

Fig. 10: Comparació d'abans i després del redisseny.

6.6.3 Formularis

Pels formularies s'ha seguit la mateixa idea. S'ha creat una vista parametritzable per crear formularis de manera ràpida i fàcil. Amb aquest sistema afegir, treure i modificar camps es pot fer tocant només 1 línia del controlador.

A la figura 11 podem veure una comparació entre el disseny anterior del formulari i el nou. L'estructura dels camps del formulari de registre s'ha mantingut, però s'han modificat els camps i els botons d'aquest.

6.6.4 Editor

Per la vista de l'editor s'ha utilitzat cap vista genèrica ja que aquesta no es reutilitza.

Amb el redisseny de la pàgina s'ha fet l'editor més gran, ja que era massa petit sent el component central d'aquesta vista. S'han modificat també els botons que apareixien a sobre de l'editor, fent que aquests segueixin l'estil de les altres vistes.

Com s'ha incorporat la funcionalitat de canviar entre tema fosc i tema clar s'ha hagut de fer que l'editor es modifiqui com la resta de la pàgina. A la figura 12 podem veure la diferència entre l'editor anterior (imatge *a*) i el nou (imatge *b*) que està utilitzant el tema fosc com la resta de la pàgina.

7 TESTATGE

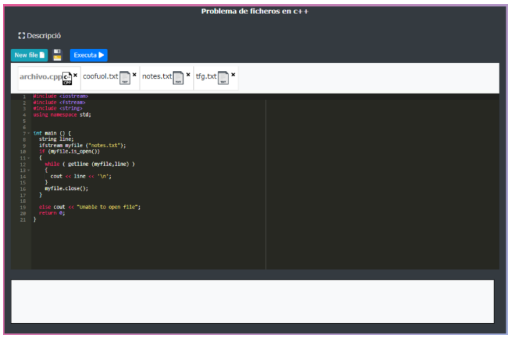
S'ha fet un testage global de la web provant els fluxos d'execució que un usuari normal faria. S'ha creat un document en el que s'indiquen tots els casos avaluats, a més, en els casos d'error, s'ha descrit perquè s'havia originat i com s'ha arreglat.

Un dels errors que s'ha detectat i no s'ha pogut solucionar és que quan un usuari entra a un exercici de Jupyter s'hauria de llançar un contenidor de Docker amb un servidor que li servís fitxers. Com la pàgina carrega més ràpid que el contenidor, l'usuari veu en comptes de l'editor de Jupyter una pàgina d'error. Un *fix* que s'ha trobat al problema és tenir un fitxer de text buit a l'exercici i fer

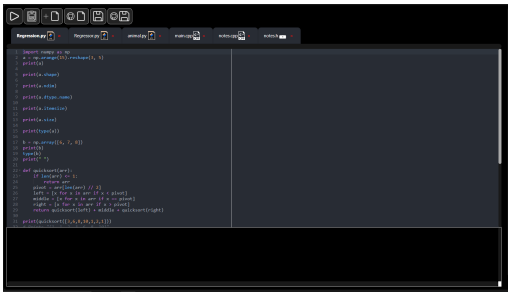
(a) Abans del redisseny.

(b) Després del redisseny.

Fig. 11: Comparació d'abans i després del redisseny.



(a) Abans del redisseny.



(b) Després del redisseny.

Fig. 12: Comparació d'abans i després del redisseny.

que es carregui primer aquest fitxer. Així, quan l'usuari canviï al fitxer '.ipynb' el contenidor ja estarà iniciat i es veurà correctament.

8 DESPLEGAMENT A SERVIDOR

Aquesta ha sigut l'última tasca que s'ha realitzat. Ha consistit en pujar el codi a una màquina del centre educatiu i fer que funcioni tal com ho feia a l'entorn de desenvolupament.

8.1 Entorn Windows

Tot i que les eines que s'havien utilitzat (Docker i Apache) estaven pensades per ser utilitzades a entorns Linux es va decidir demanar un Windows perquè el desenvolupament s'havia fet amb el mateix sistema operatiu i se sabia que funcionaria correctament i no s'hauria d'adaptar cap part del codi.

Com a la màquina s'havien d'executar molts contenidors de Docker, es va haver de fer una aproximació dels recursos necessaris per suportar la càrrega màxima del sistema, és a dir, quan 40 alumnes estiguessin executant un servidor cadascú. Cada contenidor requeria 200 MB de memòria en el millor dels casos, per tant, es va demanar que la màquina tingués com a mínim 16 GB.

Amb aquest entorn va sorgir un problema greu. Com als contenidors de Docker s'executava un sistema operatiu de Linux amb Jupyter incorporat es necessitava tenir la virtualització (una tecnologia que permet executar diferents SO dins de Windows) activa, però com l'entorn ja era una màquina virtual no va ser possible fer que funcionés.

8.2 Entorn Linux

Es va decidir canviar el SO a un Linux i tornar a començar amb l'instal·lació. Es van trobar diferents problemes amb en aquest procés:

- Algunes funciones de PHP no funcionaven de la mateixa manera que a l'anterior sistema operatiu. Per tant, es va haver d'analitzar tot el codi i solucionar els errors que anaven sorgint per aquest motiu. El que es va fer arreglar aquest problema va ser substituir les crides a funcions a execucions de comandes de Linux, tot i que això ha fet que el codi estigui molt lligat al SO.
- Els contenidors de Docker es podien executar però no eren accessibles des de fora de la xarxa local de la màquina degut a les mesures de seguretat que tenia implementades el centre docent. De les dues solucions que es van trobar per arreglar el problema, fer una petició perquè s'obrissin 40 ports i canviar la manera en la que estava pensada l'eina, es va realitzar la primera i es va deixar per escrit al dossier com s'hauria de fer la segona.

9 CONCLUSIONS

9.1 Objectius assolits

Els objectius que s'han assolit han sigut els següents:

- Fer una neteja del codi i estructurar-ho de manera que tot estigui separat per capes ben diferenciades.
- Implementar el concepte de sessions de classe a l'eina.
- Integrar amb GitHub, sigui per pujar solucions o per descarregar fitxers per afegir-los a una tasca o per crear un exercici nou.
- Poder visualitzar i modificar notebooks de Jupyter des de l'eina.
- Redissenyar la pàgina web.
- Permetre tenir dos problemes amb el mateix nom a assignatures diferents.

9.2 Objectius no aconseguits

Els objectius que no s'han assolit han sigut els següents:

- Integrar l'eina amb Caronte. No s'ha pogut complir aquesta fita per com està plantejat el mòdul que s'utilitza per crear els exercicis avaluables de programació.
- Fer que els contenidors de Docker que conten els servidors de Jupyter siguin accessibles per ordinadors que no estan a la xarxa local de la màquina. Aquest objectiu no s'ha pogut assolir perquè la universitat té mesures de seguretat aplicades i no obren ports.

En general, s'han assolit una gran part dels objectius que s'havien plantejat inicialment. Amb aquests canvis l'aplicació pot arribar a ser molt útil pels estudiants dels primers cursos, que són principalment els que més supervisió necessiten al programar.

No haver pogut aconseguir que Jupyter funcionés a la màquina del centre ha suposat un problema que no s'havia plantejat i ha fet que una funcionalitat important no es pugui utilitzar. Tot i això, els coneixements que s'han adquirit sobre la utilització d'APIs, els llenguatges de programació que s'han utilitzat i la utilització de contenidors Docker considero que són molt valuosos.

REFERÈNCIES

[1] Y. A. Asbahi, "Aula de programació interactiva." https://shorturl.at/jkruG, 2022.

[2] "Jupyter notebook." https://en.wikipedia.org/wiki/Project_Jupyter.

[3] "What is agile?." https://www.atlassian.com/agile.

[4] "Trello." https://trello.com/en.

[5] "Sonarlint." https://www.sonarlint.org/.

[6] "Github rest api." https://docs.github.com/en/rest.

[7] "Github rest api wrapper for php." https://github.com/KnpLabs/php-github-api.

[8] "Oauth apps creation." https://docs.github.com/es/developers/apps/building-oauth-apps.

[9] "Binder." https://jupyter.org/binder.

[10] "Vpl plug-in for moodle." https://github.com/jcrodriguez-dis/moodle-mod_vpl.

APÈNDIX

A.1 Diagrama d'entitats i relacions

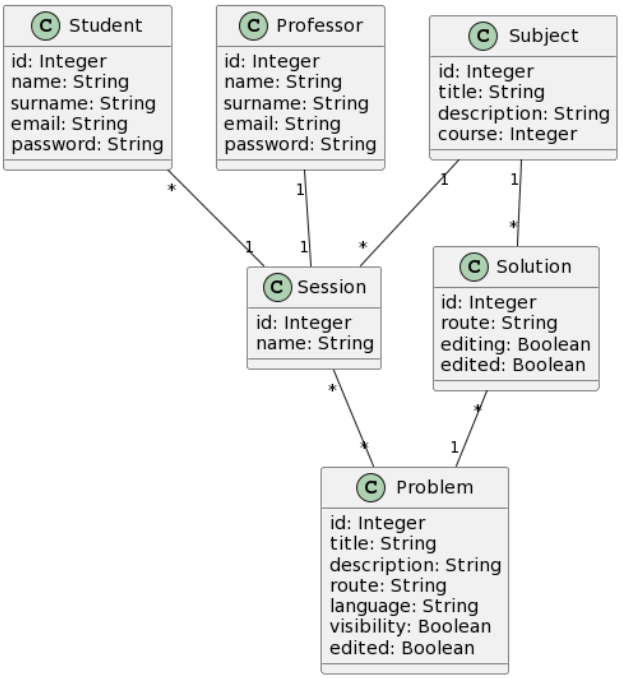


Fig. 13: Diagrama d'entitat-relació de la web.

A.2 Diagrama de casos d'ús

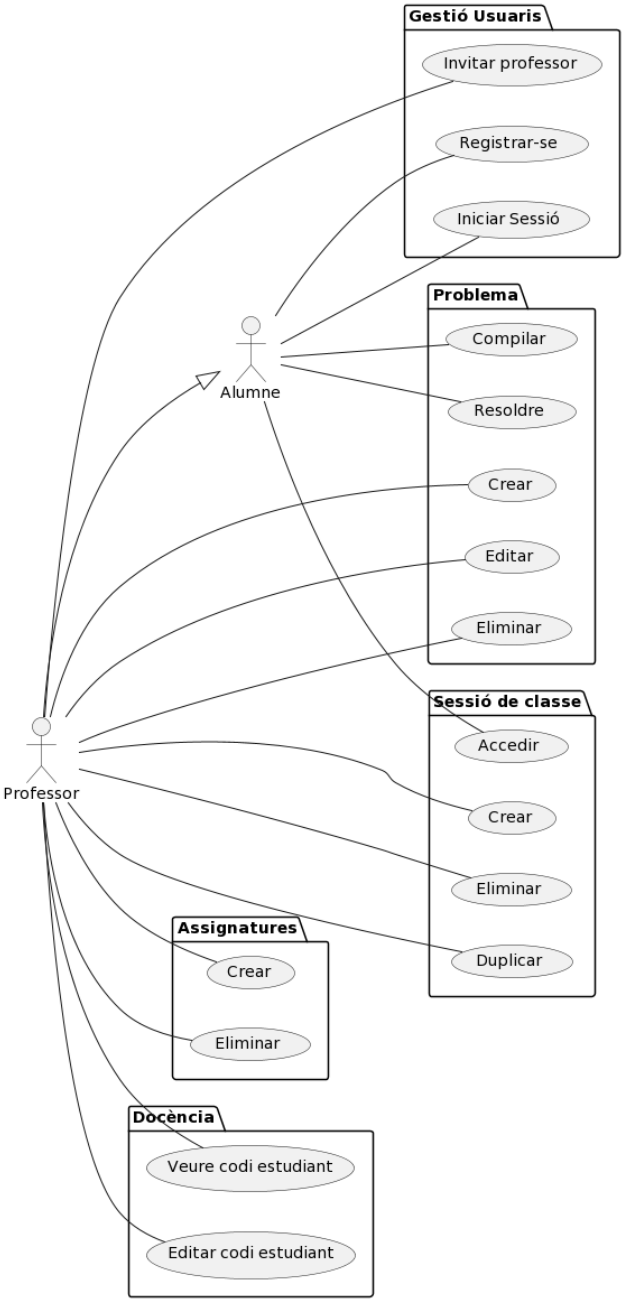


Fig. 14: Diagrama de casos d'ús de la web.