Universitat Oberta de Catalunya

uoc.edu

Recomanador de matrícules semestrals

Miquel Serrabassa Lafont
Grau d'Enginyeria Informàtica
Desenvolupament web

Pablo Pineda Ruipérez
Santi Caballe Llobet

07/01/2021

FITXA DEL TREBALL FINAL

Títol del treball:	*Recomanador de matrícules semestrals*
Nom de l'autor:	*Miquel Serrabassa Lafont*
Nom del consultor/a:	*Pablo Pineda Ruipérez*
Nom del PRA:	*Santi Caballe Llobet*
Data de lliurament (mm/aaaa):	*01/2022*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*Desenvolupament web*
Idioma del treball:	*Català*
Paraules clau	*WebApp, PWA, matrícula, recomanador, Javascript, JSON, HTML5, CSS, localStorage*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Eina per a recomanar a quines assignatures s'ha de matricular un alumne basant-se en la seva disponibilitat. Desenvolupat com a PWA (Progressive Web App) amb una estructura de dades que permeti utilitzar-ho en diferents estudis, tot i que es tractarà sobre el Grau d'Enginyeria Informàtica. Permetrà desar les convalidacions i les assignatures matriculades i els resultats de cada semestre i l'itinerari escollit i generarà un calendari de matriculació proposat segons les hores disponibles per als següents semestres.

Abstract (in English, 250 words or less):

Tool to recommend subjects enrollment based on Student availability. Developed as a PWA (Progressive Web App) with a data structure to be used in diferent studies, although it will be used in the Degree in Computer Engineering. It will Store validations and results of enrolled subjects for each semestre, and the chosen itinerary, and it will generate proposed enrollment calendars according to the students' availability.

Índex

1. Introducció

1.1 Context i justificació del Treball

Quan ja fa uns anys vaig començar a estudiar a la UOC, em vaig adonar que necessitaria disciplina no tant sols a nivell d'estudis sinó també a nivell organitzatiu: un grapat d'assignatures, venint d'una possible convalidació de FPII, amb un itinerari a escollir, optatives que depenien -o tenien recomanacions- d'haver escollit altres assignatures abans...

Al final, suposo que com molts alumnes, he estat reinventant fulles de càlcul i esquemes que revisava i actualitzava cada semestre per saber a què em podria matricular, calcular si tindria temps o no, veure si em podria combinar dates d'exàmens...

Sovint, sobretot al principi, és força complicat saber com arribar a fer tots els crèdits necessaris, arribar a escollir bé les optatives per l'itinerari o saber si tindrem temps durant un semestre si escollim dues, quatre o més assignatures.

Hi ha moltes solucions que s'han anat aplicant de manera individual alguns alumnes, però en general no sembla que hi hagi cap aplicació en forma d'"assistent de matrícula", i sovint els tutors, tot i que ajuden en els tràmits, no coneixen o poden recordar les necessitats individualitzades de cada alumne.

Així, l'objectiu d'aquest treball és preparar un aplicatiu web on poguem indicar els estudis que volem fer (tot i que inicialment la implementació inclou només els d'Enginyeria Informàtica) i ens ajudi a fer les matrícules i tingui un històric d'assignatures fetes i possibles recomanacions de què fer en base a la previsió d'hores lliures que tenim pel proper semestre.

Inicialment tindrà parametritzades una sèrie de dades sobre les assignatures -tot i que podria haver-hi una part de *scrapping* o ús d'APIs del Campus virtual si existís-, de manera que l'usuari no necessiti parametritzar tota una carrera sinó només les seves preferències personals.

També, per tenir una eina amb previsió d'ús i creixement futur, he optat per crear una PWA que fa ús de *localStorage* per desar les dades de l'usuari, minimitzant l'estructura necessària per funcionar. Així, part d'aquest projecte inclou l'estudi necessari que es poden implementar totes les funcionalitats desitjades sense necessitat d'executar codi en un servidor extern.

Així que la meva proposta és crear un aplicatiu que, en primer lloc, sol·liciti una sèrie de dades "personals", com el nom, els estudis que es cursen i la disponibilitat en hores setmanals, i a partir d'aquí generi una

sèrie de propostes basades en això, i també en les assignatures que es vagin superant.

Per tal de realitzar aquests càlculs, tindrem en compte:
- Disponibilitat de l'alumne
- Matèries ja cursades o convalidades
- Matèries que l'alumne pugui voler cursar abans
- Assignatures que només es fan un semestre
- Crèdits de cada assignatura
- Recomanacions de matrícula (quines assignatures cal haver cursat abans que unes altres...)
- Preferències d'itineraris de l'usuari

1.2 Objectius del Treball

La realització del TFG té una implicació bàsica a nivell d'objectius**: la finalització del grau**, amb un pes important a nivell acadèmic i personal, per un costat, i per l'altre, la finalització d'un projecte que he trobat a faltar durant bona part dels estudis: **una aplicació web de tipus progressiu (PWA) de recomanació automàtica d'assignatures a matricular per cada semestre** en funció de criteris com la disponibilitat horària o la preferència de certes assignatures o itineraris.

Així, aprofitant la definició de l'objectiu principal de crear una aplicació de recomanació de matrícula, també vull aprofitar per aprofundir en temes més concrets que es deriven d'aquest objectiu, i que són:

- Treballar amb dades en **JSON** per parametritzar la informació.
    - Per donar l'objectiu principal per superat, hauré de recollir totes les dades relatives als diferents càlculs que implementi i establir la parametrització que acabi realitzant de manera que pugui operar amb aquestes dades. En cas de tenir-hi problemes, no descarto fer-ho en MySQL.

- Desenvolupar l'eina en **HTML5 i Javascript sense llenguatges *server-side***. Aprendre a crear arxius de manifest de Progressive Web Apps (**PWA**) i configurar-lo adequadament. Ús de **localStorage** i, en cas de ser possible, verificació de compatibilitat del navegador.

    - Per donar aquest objectiu per superat, hauré d'haver desenvolupat un prototip funcional (també MVP) que em permeti interactuar amb els càlculs que acabi implementant, que permeti instal·lar-se com a WebApp. Si no em fos possible desenvolupar els càlculs en Javascript, optaria per un llenguatge com PHP.

- Convertir l'experiència que he anat adquirint a l'hora de matricular-me en fòrmules tangibles que permetin **automatitzar la recomanació de matrícula** de cada semestre.

    - Per aquest objectiu, el meu plantejament és aplicar la meva situació personal al llarg dels anys (matriculacions que he fet, en quin ordre, ...) per tal de comparar-ho amb la recomanació que obtingui de la meva pròpia eina. Òbviament, com que he tingut qüestions personals que m'han condicionat, no espero que en resulti exactament el meu mateix itinerari, però sí que n'he d'obtenir una resposta lògica i assumible.

    - Al preparar l'estructura de dades, l'experiència m'ha servit per modelar la informació i he pogut tenir en compte la majoria de problemes que em podria trobar en els càlculs de dades. Tot i això, caldrà esperar a la implementació del desenvolupament perquè sempre pot ser que surti algun problema de darrera hora que no hagi tingut en compte!

- **Obtenir experiència** en l'elaboració de projectes basats primer en la documentació i després en la implementació.

    - A la pràctica, "escriure-ho" tot i després desenvolupar-ho acaba sent una via poc realista i eficient en el món laboral real on els clients volen veure "resultats" i els estudis tècnics i tota la feina sobre el paper és de vegades molt difícil de justificar malgrat ser necessària, fet que implica haver de programar prototips o demostracions abans de poder començar el projecte en si mateix i de vegades el condicionen. Tot i això, al tractar-se d'un cas ideal, i al fer-ho en un entorn acadèmic, és un bon moment per posar-ho en pràctica de cara a establir, amb cert marge i tranquilitat, una sèrie d'objectius propers en la meva carrera laboral. En aquest cas, com a objectiu sembla menys mesurable, però el treball actual i el de la propera PAC en seran els resultats: si amb aquesta informació puc desenvolupar un MVP voldrà dir que ho he assolit.

    - També és important destacar que he tingut alguns problemes, entre la PAC1 i la PAC2, per documentar correctament els passos, sempre per por a no passar-me de llargada en la documentació -i potser resumint massa. És en aquest punt on ja considero estar acomplint objectius tot aprenent a fer documentacions més extenses amb informació d'interès.

A nivell de tasques concretes, m'he plantejat, per ordre de prioritat, una sèrie de punts per anar completant durant el desenvolupament de les PAC:

- Definició de funcionalitats a desenvolupar
    - Definició de la parametrització d'uns estudis d'exemple (Enginyeria informàtica) en format JSON
    - Càlcul de càrrega semestral possible segons hores
    - Càlcul d'assignatures ja realitzades per seguir itinerari recomanat
    - Càlcul de convalidacions
    - Càlcul d'optatives segons itinerari
    - Càlculs per cursar assignatures bisemestrals.
    - Desar i mostrar les notes i semestres d'assignatures ja superades

- Definició de la interfície d'usuari:

    - Sol·licitud de dades:
        - Nom
        - Hores disponibles setmanals
        - Quin grau es vol fer?
        - Ordre de preferència dels itineraris
    - Ampliació de les dades
        - Sol·licitar si ja s'ha avançat en assignatures fetes o convalidacions. En cas afirmatiu:
            - Convalidacions (assignatures convalidades o parametritzar possibles convalidacions segons estudis previs)
            - Assignatures fetes prèviament
            - Assignatures que es vol matricular en un semestre concret o forçar-ne la matrícula
    - Presentació de resultats
        - Gràfic de properes matriculacions
        - Taula de dades informatives

- Investigació per definir quines funcionalitats són viables amb HTML+JS i JSON per parametritzar les dades.

    - Creació de manifest per la PWA
    - Proves amb localStorage i altres elements concrets de les PWA
    - Verificar que es pot desenvolupar tot el projecte amb eines *client-side*, sense necessitat d'un servidor amb programari com intèrpret de PHP o bases de dades MySQL
    - Documentar el procediment com a part de la memòria del TFG.

- Desenvolupament de l'eina (o la part que sigui possible realitzar)

    - Creació de la interfície d'usuari
    - Desenvolupament dels càlculs a realitzar
    - Parametrització en JSON de les dades
    - Integració de càlculs i gràfiques en la interfície

- Proves
  - Comprovació de la interfície en diferents navegadors i mides de pantalla: **proves de compatibilitat.** [2]
  - Comparació de resultats de l'eina amb el meu cas particular (que he anat documentant amb el pas dels semestres): a partir de **tests funcionals** [3] sobre funcionalitats concretes.
  - Proves amb valors invàlids o sense sentit per evitar que provoquin problemes tècnics: altrament anomenat ***destructive testing*** [4]
  - Avaluació de la interfície: **testos d'usabilitat** [5]. En aquest àmbit, descriurem les proves, però per la seva complexitat, no tindrem temps ni infraestructura suficient per dur-les a terme.

1.3 Metodologia

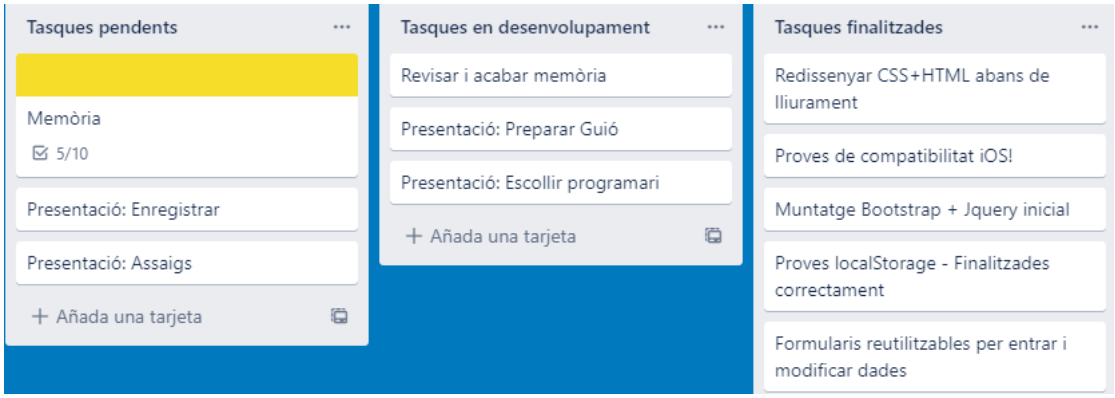
Com que entre els objectius hi ha una part important d'aprenentatge, i com que, a més a més, no he trobat cap producte similar al que jo busco, crec que no hi ha altra opció que desenvolupar un producte nou.

A més, en el meu cas, la meva intenció inicial és tenir un producte que sigui mínimament funcional i fàcilment adaptable, però inicialment destinat a alumnes que el vulguin utilitzar amb objectius personals, no comercials. Això, però no treu que en un futur es pogués convertir en un producte comercial que es pogués adaptar a plataformes existents d'universitats per a generar propostes de matrícula automatitzades.

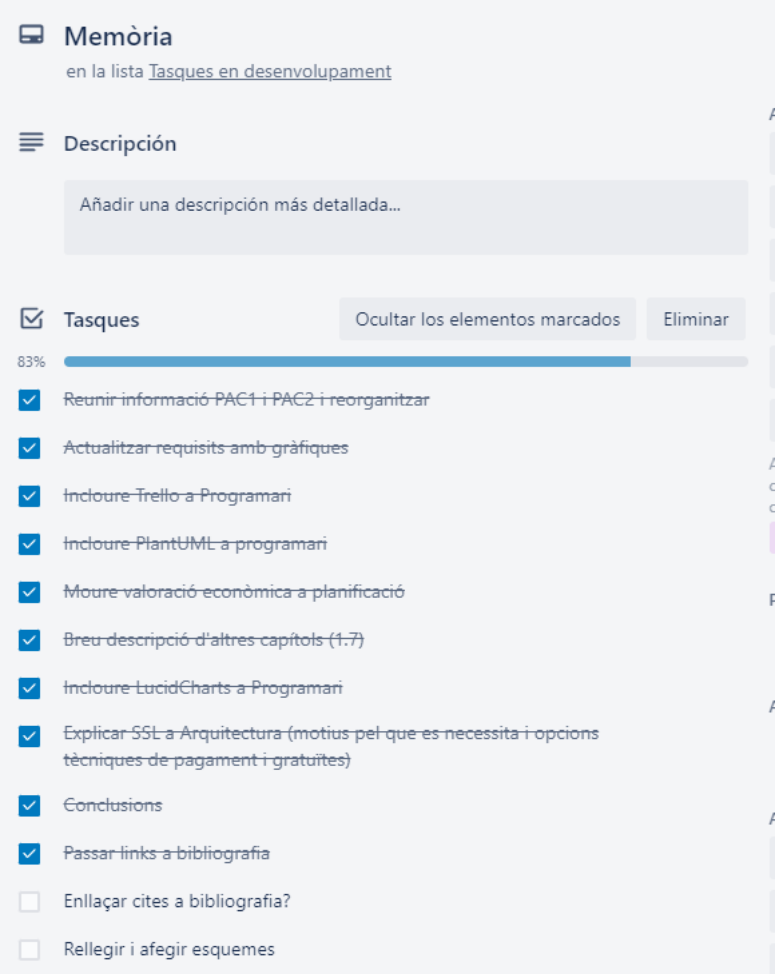
Per tal d'assolir els objectius, cal escollir una metodologia que m'ajudi a dur a terme les tasques necessàries. En aquest cas, he estat treballant amb la metodologia KanBan [6], un tipus de metodologia àgil [7]. Com que estem parlant d'un projecte on el meu objectiu era aprendre i descobrir, prèviament no tenia una idea clara de quines serien les tasques per a fer un desenvolupament en cascada clar. És per això que la metodologia àgil, basada en el desenvolupament iteratiu, era la millor opció. Concretant en el mètode Kanban, aquest es defineix per l'ús de "targetes" de tasques, per tal de limitar el treball de cada iteració i posar objectius concrets i assumibles, a més de donar una visió ràpida de què hi ha fet i què hi ha pendent.

A més, hi ha moltes eines que poden donar suport a aquesta metodologia, entre les quals n'hi ha de conegudes com Trello -que he anat utilitzant-, Asana o altres.

A nivell pràctic, he utilitzat Trello, emprant un tauler amb tres llistes: una de tasques pendents, on hi he anotat les tasques que anava detectant i que creia que es podien fer en un *sprint* -que de vegades he hagut de dividir en tasques més petites-, una altra de tasques que estava duent a terme en un moment determinat, i finalment la de tasques finalitzades.



A més, en alguns casos, he optat per fer una targeta de tasques amb *checkpoints*, per poder-ho dividir en subtasques que considerava massa petites per fer una targeta pròpia:



1.4 Planificació del Treball

Tot i que sempre poden sortir canvis i millores, i més en un projecte fet i pensat en tant poc temps, és cert que la planificació semestral permet establir una sèrie de marges de temps màxims per a cada pas. Val a dir que, com que treballo i disposo d'un temps limitat pels meus estudis,

calculo que podré destinar unes 3 hores diàries de mitjana al TFG (entre 2 i 4 en funció del dia, entre setmana).

Així, aprofitant la temporalització que vaig fer per a les PAC i les comparacions dels objectius assolits, he pogut avaluar l'acompliment de les tasques realitzades.

Amb la planificació inicial, que ja vaig intentar que fos mínimament realista a partir de la meva disponibilitat, s'ha pogut acomplir l'objectiu temporal que era el següent:

- Definició de funcionalitats a desenvolupar **(1 setmana aprox.)**
    - Definició de la parametrització d'uns estudis d'exemple (Enginyeria informàtica) en format JSON: 3 dies
    - Càlcul de càrrega semestral possible segons hores (parametritzat per si cal ajustar): 1 dia
    - Càlcul d'assignatures ja realitzades per seguir itinerari recomanat: 1 dia
    - Càlcul de convalidacions: 1 dia
    - Càlcul d'optatives segons itinerari: 1 dia
    - Càlculs per cursar assignatures bisemestrals.: 1 dia
    - Desar i mostrar les notes i semestres d'assignatures ja superades: 1 dia

- Definició de la interfície d'usuari: **(1 setmana aprox)**
    - Quines dades de l'usuari necessitem saber?
        - Nom
        - Hores disponibles setmanals? Crèdits semestrals a cursar?
        - Assignatures ja cursades
        - Convalidacions (assignatures convalidades o parametritzar possibles convalidacions segons estudis previs)
    - Com presentem els resultats?
        - Gràfic de properes matriculacions?
        - Percentatges fet vs pendent?

- Investigació per definir quines funcionalitats són viables amb HTML+JS i JSON per parametritzar les dades. **(2 setmanes aprox)**
    - Creació de manifest per la PWA: 2 dies
    - Fer proves amb localStorage: 3 dies
    - Veure si és possible o si finalment em decanto per PHP+MySQL: 1 setmana
    - Documentar el procediment com a part del TFG: inclòs en la resta

De manera que a partir d'aquest punt, ens quedaran les següents tasques:

- Desenvolupament de l'eina (o la part que sigui possible realitzar): **(5 setmanes)**
    - Creació de la interfície d'usuari: 1 setmana
        - Formulari d'entrada de dades de l'usuari, inicial

        - Formulari per indicar convalidacions i assignatures matriculades/superades/suspeses
        - Mostra de resultats de la recomanació
            - Versió en taules
            - Versió gràfica

- Desenvolupament dels càlculs a realitzar: **1 setmana**
    - Càrrega de l'arxiu JSON al localStorage
    - Escriptura de dades d'usuari al localStorage
    - Càlcul de càrrega setmanal: deduïr crèdits a partir d'hores setmanals.
    - Càlcul d'optatives segons preferència d'itineraris
        - Escollint només l'itinerari preferit
        - Ponderant, si és possible, segons la preferència d'itineraris
        - Càlcul d'assignatures bisemestrals
        - Manteniment d'assignatures convalidades i realitzades

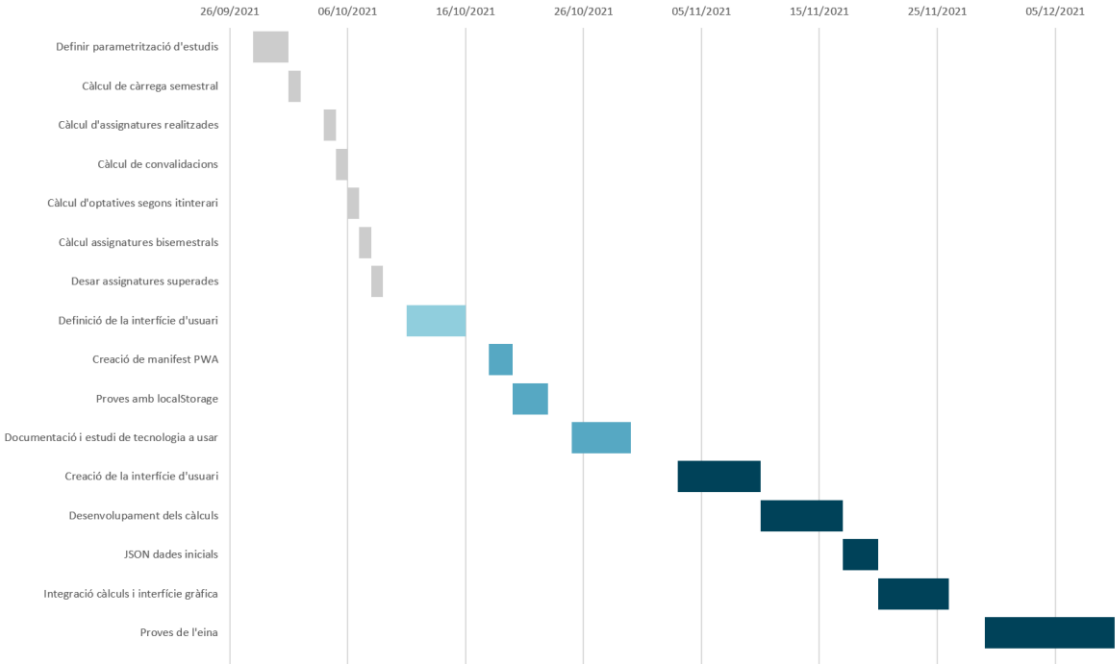
- Revisió de la parametrització en JSON de les dades inicials (Enginyeria Informàtica + convalidacions DAW/DAM): **3 dies**
    - Possibilitat d'afegir nous camps o d'eliminar camps que no facin falta
    - Intentar incloure convalidacions DAW/DAM per entrar-les directament si queda temps.

- Integració de càlculs i gràfiques en la interfície: **1 setmana**
    - Automatització dels càlculs, possibilitat de recalcular en temps real, creació de gràfiques amb D3s o similars.

- Proves: **2 setmanes**
    - Comprovació de la interfície en diferents navegadors i mides de pantalla: **proves de compatibilitat**.
    - Comparació de resultats de l'eina amb el meu cas particular (que he anat documentant amb el pas dels semestres): a partir de **tests funcionals** sobre funcionalitats concretes.
    - Proves amb valors invàlids o sense sentit per evitar que provoquin problemes tècnics: altrament anomenat ***destructive testing***
    - Avaluació de la interfície: planificació dels **testos d'usabilitat**

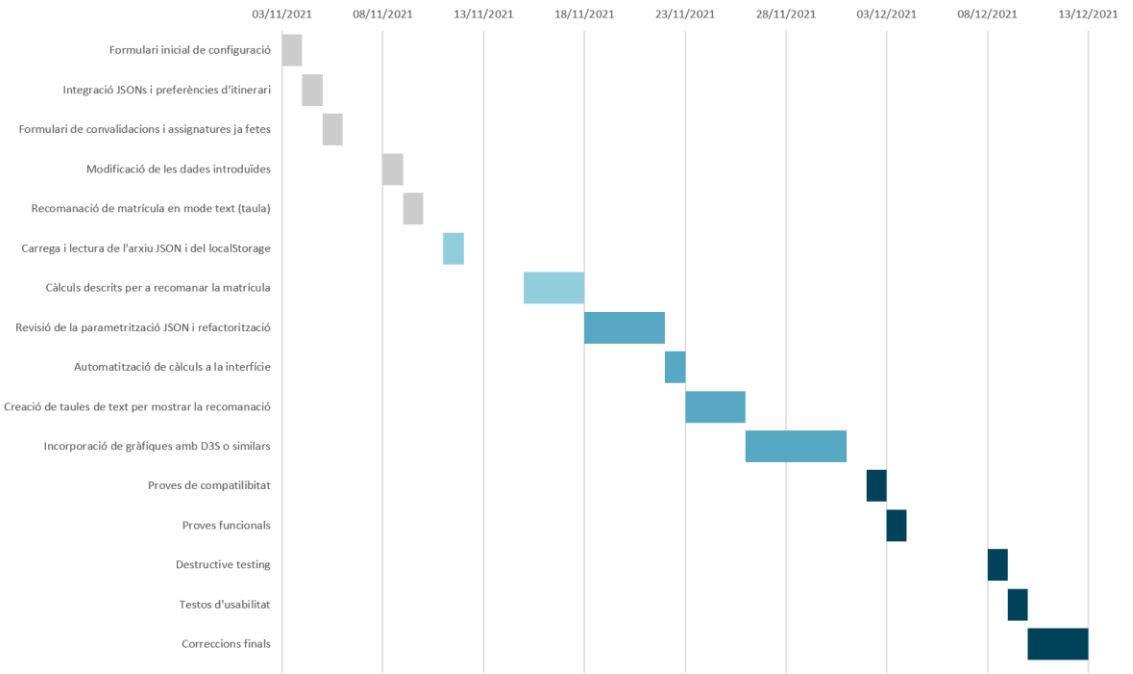
A partir d'aquesta planificació, he utilitzat com a base de seguiment temporal els següents diagrames de Gantt:

TASCA	INICI	FINAL	DURADA	FASE
Definir parametrització d'estudis	28/09/2021	01/10/2021	3	Definició de funcionalitats
Càlcul de càrrega semestral	01/10/2021	02/10/2021	1	Definició de funcionalitats
Càlcul d'assignatures realitzades	04/10/2021	05/10/2021	1	Definició de funcionalitats
Càlcul de convalidacions	05/10/2021	06/10/2021	1	Definició de funcionalitats
Càlcul d'optatives segons itinterari	06/10/2021	07/10/2021	1	Definició de funcionalitats
Càlcul assignatures bisemestrals	07/10/2021	08/10/2021	1	Definició de funcionalitats
Desar assignatures superades	08/10/2021	09/10/2021	1	Definició de funcionalitats
Definició de la interfície d'usuari	11/10/2021	16/10/2021	5	Definició d'interfície
Creació de manifest PWA	18/10/2021	20/10/2021	2	Primers passos i definició de tecnologia a usar
Proves amb localStorage	20/10/2021	23/10/2021	3	Primers passos i definició de tecnologia a usar
Documentació i estudi de tecnologia a	25/10/2021	30/10/2021	5	Primers passos i definició de tecnologia a usar
Creació de la interfície d'usuari	03/11/2021	10/11/2021	7	Desenvolupament de l'eina
Desenvolupament dels càlculs	10/11/2021	17/11/2021	7	Desenvolupament de l'eina
JSON dades inicials	17/11/2021	20/11/2021	3	Desenvolupament de l'eina
Integració càlculs i interfície gràfica	20/11/2021	26/11/2021	6	Desenvolupament de l'eina
Proves de l'eina	29/11/2021	13/12/2021	14	Desenvolupament de l'eina



Un cop arribat al desenvolupament de l'eina, també he pogut elaborar un altre diagrama ja destinat en exclusiva a la part de desenvolupament, granularitzant les tasques un cop les havia detectat:

TASCA	INICI	FINAL	DURADA	FASE
Formulari inicial de configuració	03/11/2021	04/11/2021	1	Creació de la interfície d'usuari
Integració JSONs i preferències d'itinera	04/11/2021	05/11/2021	1	Creació de la interfície d'usuari
Formulari de convalidacions i assignatu	05/11/2021	06/11/2021	1	Creació de la interfície d'usuari
Modificació de les dades introduïdes	08/11/2021	09/11/2021	1	Creació de la interfície d'usuari
Recomanació de matrícula en mode tex	09/11/2021	10/11/2021	1	Creació de la interfície d'usuari
Carrega i lectura de l'arxiu JSON i del lo	11/11/2021	12/11/2021	1	Desenvolupament Javascript dels càlculs
Càlculs descrits per a recomanar la mat	15/11/2021	18/11/2021	3	Desenvolupament Javascript dels càlculs
Revisió de la parametrització JSON i ref	18/11/2021	22/11/2021	4	Revisió JSON i integració de càlculs i gràfiques
Automatització de càlculs a la interfície	22/11/2021	23/11/2021	1	Revisió JSON i integració de càlculs i gràfiques
Creació de taules de text per mostrar la	23/11/2021	26/11/2021	3	Revisió JSON i integració de càlculs i gràfiques
Incorporació de gràfiques amb D3S o sir	26/11/2021	01/12/2021	5	Revisió JSON i integració de càlculs i gràfiques
Proves de compatilibitat	02/12/2021	03/12/2021	1	Proves i correccions
Proves funcionals	03/12/2021	04/12/2021	1	Proves i correccions
Destructive testing	08/12/2021	09/12/2021	1	Proves i correccions
Testos d'usabilitat	09/12/2021	10/12/2021	1	Proves i correccions
Correccions finals	10/12/2021	13/12/2021	3	Proves i correccions



1.5 Valoració econòmica

A partir de la planificació, i ja amb la informació referent a les durades de les diferents tasques, podem realitzar una valoració econòmica per calcular el cost que podria tenir el desenvolupament d'una eina d'aquestes característiques, amb dues possibilitats de preu/hora diferents en funció de consultes que he realitzat a diversos professionals.

Comptarem amb preus de 25€/hora i 40€/hora -també de cara a veure costos reals possibles i també justificació de costos més alts per si ho volguéssim vendre.

Tasca	Hores	Cost 25€/h	Cost 40€/h
Anàlisi i disseny (PAC2)	**69**	**1.725€**	**2.760€**
Definició de funcionalitats	5	125€	200€
Definició d'arquitectura	2	50€	80€
Parametrització i definició de càlculs	42	1.050€	1.680€
Definició de la interfície d'usuari	12	300€	480€
Documentació	8	200€	320€
Desenvolupament (PAC3)	**114**	**1.850€**	**4.560€**
Creació de la interfície (Frontend)	20	500€	800€
Desenvolupament en JS dels càlculs	28	700€	1.120€
Revisió de parametrització JSON	6	150€	240€
Integració de gràfiques i taules	20	500€	800€
Proves i correccions	40	1.000€	1.600€
TOTAL	**183**	**4.575€**	**7.320€**

1.6 Breu sumari de productes obtinguts

Aquest treball es realitza per a obtenir dos productes principals. Per una part, la documentació aportada en aquesta memòria, i per l'altra, el producte mínim viable de l'eina que m'he disposat a desenvolupar.
Aquesta eina, que es podrà publicar a Internet, es descarregarà en un arxiu comprimit que contindrà arxius HTML, Javascript, CSS i JSON, tal com aquesta memòria descriu més endavant de forma detallada.

1.7 Breu descripció dels altres capítols de la memòria

A continuació, començarem a analitzar els **requisits** de l'eina, a nivell funcional i tècnic, per a tenir una definició clara de quin serà l'àmbit del projecte, amb una planificació temporal i una valoració econòmica basada en aquests indicadors.

Amb això, es passarà a estudiar **l'Estat de l'art**, valorant opcions similars o complementàries que s'hagin trobat per tal de donar context a la decisió de construir aquesta eina.

Seguirem amb l'anàlisi de **tecnologies, frameworks i programari** que utilitzarem en el projecte, per tal de donar una base sobre les eines i tecnologies que volem utilitzar, ja que també són part essencial del projecte i, de fet, fer-ho amb un determinat grup d'eines i no amb unes altres era precisament el repte inicial d'aquest Treball.

Seguirà l'apartat sobre **l'Arquitectura**, amb la definició del concepte, així com dels elements principals a tenir en compte: PWA, la parametrització de les dades en JSON, la interfície d'usuari, els càlculs a desenvolupar, les proves realitzades i planificades, i també les explicacions per a utilitzar-ho en un ordinador local complint les restriccions dels principals navegadors.

Finalment, en traurem **Conclusions** i analitzarem algunes de les millores que es podrien dur a terme a partir dels treballs elaborats durant aquest projecte

2. Estat de l’art

2.1 Recerca d’altres solucions

A l’hora de buscar informació sobre aquest apartat cal recordar que la idea d’aquest projecte la vaig tenir precisament perquè no vaig trobar cap eina similar mentre estava cursant la resta d’assignatures del Grau d’Enginyeria Informàtica.

Si bé existeixen, en grups de Telegram d’alumnes, alguns arxius Excel que s’han anat fent per intentar fer el seguiment, així com els PDFs que es poden trobar al web de la UOC, no existeix cap eina més o menys estandarditzada que pugui utilitzar-se per això.

A més, a l’haver-hi tantes formes diferents de cursar els estudis, es fa difícil que un full de càlcul d’un usuari pugui valer per a altres -sobretot perquè al final, la representació de dades en una sola taula dona menys possibilitats.

En el meu cas, per exemple, he estat fent esquemes gràfics i arxius en processadors de textos.

A més, per a l’elaboració del treball, he ampliat la recerca per diverses vies, però no he aconseguit trobar cap eina que s’adaptés a la descripció del què pretenc fer, i m’ha sorprès però també és cert que la majoria d’universitats presencials tenen un sistema de “semestres” força tancat i per tant només és aplicable a universitats a distància i per a gent que no disposa de temps per cursar els crèdits de 30 en 30.

Sí que existeixen eines que podrien complementar-s’hi molt bé, com una proposada pel meu tutor del TFG que és precisament d’un altre projecte de recerca, i que es basa en la recomanació d’assignatures per part d’altres alumnes per ajudar a decidir sobre quines matricular. La definició de l’eina, que es pot trobar a [8] té algunes diferències que la fan més aviat complementària que competidora, i algunes deficiències que he volgut enfocar ja d’entrada:

- Es basa en una aplicació mòbil només per a Android, de manera que caldria doblar la feina per a desenvolupar-ho també en iOS o possibles nous sistemes operatius futurs. En el meu cas, fer-ho com una aplicació web vol facilitar la integració multiplataforma de l’eina que, a més, també és interessant poder fer servir en ordinadors.

- Es basa en la participació dels usuaris -a mode de web 2.0. Si bé és una aproximació interessant com a complement de la pròpia matrícula, això aporta en realitat informació sobre la experiència personal de cada alumne -que pot dependre del professor que li hagi tocat, de les seves circumstàncies personals i expectatives... En el meu cas, buscava una eina que, més enllà de valoracions qualitatives,

tanqués una proposta “finalista”, entenent que ofereix una opció de matriculació que ja compta amb tots els criteris “tècnics” i ens permet després a nosaltres modificar-la manualment. També ofereix la possibilitat d’utilitzar-la de mode individual -només hauríem de crear nous estudis, que podríem fer-ho manualment, però no caldria la intervenció de tercers puntuant-les.

2.2 Públic objectiu

Així doncs, el públic objectiu de la idea original del meu projecte serien els mateixos estudiants. Això implica que ho imagini com una eina de codi obert i lliure, el més escalable i portable possible, i amb una parametrització dels estudis el més simple possible de cara a assolir tots els objectius.

En cas que funcionés, és evident que al ser només HTML i JSON, seria molt fàcil d’integrar-ho en webs ja existents, com ara les de les Universitats, i a partir d’APIs o JSONs propis, podrien oferir la mateixa funcionalitat de manera simple -tot i que potser voldrien canviar alguna parametrització.

Òbviament, dependria de la tecnologia emprada en cada cas, però la majoria de sistemes de gestió de continguts (CMS) del mercat admeten codi HTML, i també és universal en tots els navegadors sigui quina sigui la tecnologia de servidor, de manera que és l’opció que deixa més opcions obertes.

Evidentment, hi ha altres casos a estudiar: des de convertir en fases posteriors els propis càlculs en funcions consultables a través d’una API fins a intentar tenir integracions més complexes.

Però això seria ja més en vistes a vendre un producte, ja que requeriria servidors més cars i altres opcions que ho allunyarien de la facilitat que busco de cara a que ho utilitzi un estudiant sense massa nocions -més enllà d’editar un arxiu JSON, que en fases posteriors podria incloure’s amb un editor dins de la pròpia eina perquè fos més simple.

3. Requisits de l'eina

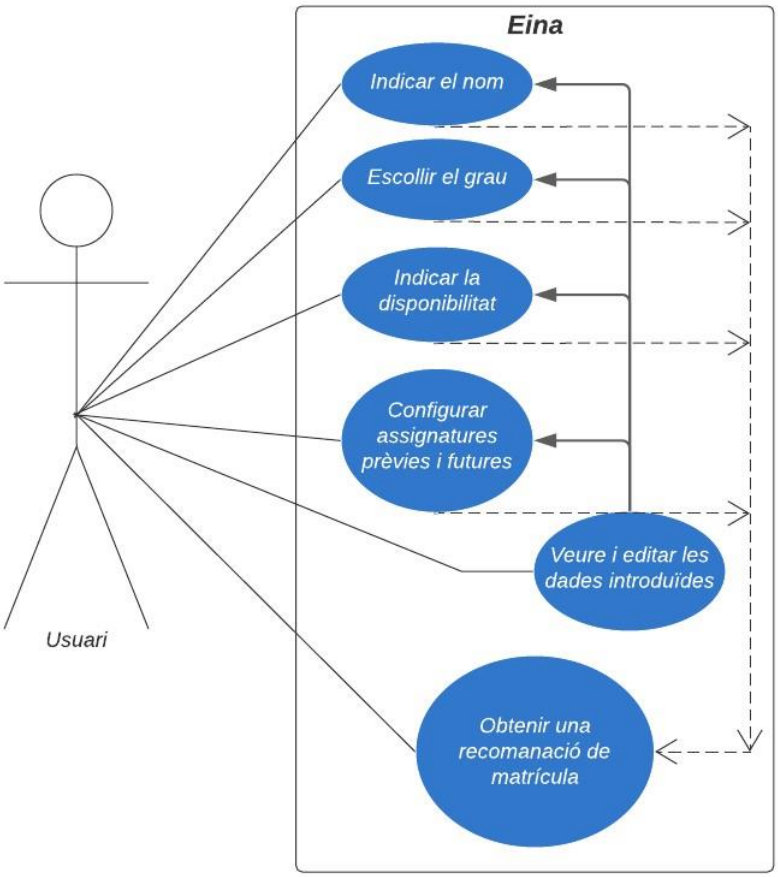
Havent analitzat les necessitats que volem cobrir, i que no hem trobat res similar, però sí alguna eina complementària, podem passar a analitzar quines són les necessitats finals que cobrirem i en base a quines funcionalitats ho volem basar.

Per tal de definir amb més concreció el funcionament final de l'eina a desenvolupar, llistarem una sèrie de requisits i casos d'ús per tal de definir les diferents tasques a generar i poder justificar les diferents decisions de disseny. Com que només hi ha un rol, el d'usuari, només ens centrarem en les funcionalitats que li apliquen:

- Com a usuari vull:
  - Accedir i indicar el meu nom i que es desi
  - Escollir els meus estudis si estan entrats a l'eina
  - Introduir les hores setmanals de disponibilitat per estudiar
  - Ordenar els itineraris per ordre de preferència
  - Indicar si he cursat alguns estudis previs que puguin convalidar assignatures, i quins.
    - I, si no els trobo, poder seguir sense indicar-ne cap.
  - Indicar si ja he cursat o convalidat assignatures, i quines
  - Poder forçar la matrícula d'alguna assignatura en un semestre determinat.
  - Modificar el meu nom i que es desi
  - Rebre recomanacions sobre les hores de disponibilitat per optimitzar els meus recursos, i poder-les modificar
  - Poder afegir, modificar o eliminar assignatures ja fetes, convalidades o matrícules que vulgui forçar.
  - Obtenir una recomanació de matrícula segons les dades que hagi indicat
    - En format gràfic
    - En format taula
  - Poder instal·lar l'eina com si fos una aplicació en múltiples dispositius -Progressive Web App.
  - Poder utilitzar l'eina en qualsevol lloc
    - En diferents dispositius
    - En diferents sistemes operatius
    - Amb diferents navegadors
    - Amb un disseny que s'adapti als diferents tamanys de pantalla

A partir d'aquestes funcionalitats he pogut generar un diagrama de casos d'ús simple per comprendre, a nivell abstracte, quina seria la relació entre les funcionalitats descrites.

En aquest diagrama hi podem veure les diferents accions que pot realitzar l'usuari -i fins a quin punt podem reutilitzar-ne la part tècnica o d'interfície que generem un cop ho desenvolupem- i, al final, la funcionalitat principal, la generació de recomanacions de matrícula.



A més, a partir dels detalls més tècnics, veiem com en podem extreure alguns requisits més concrets i globals. Voldrem, doncs, que l'eina sigui:

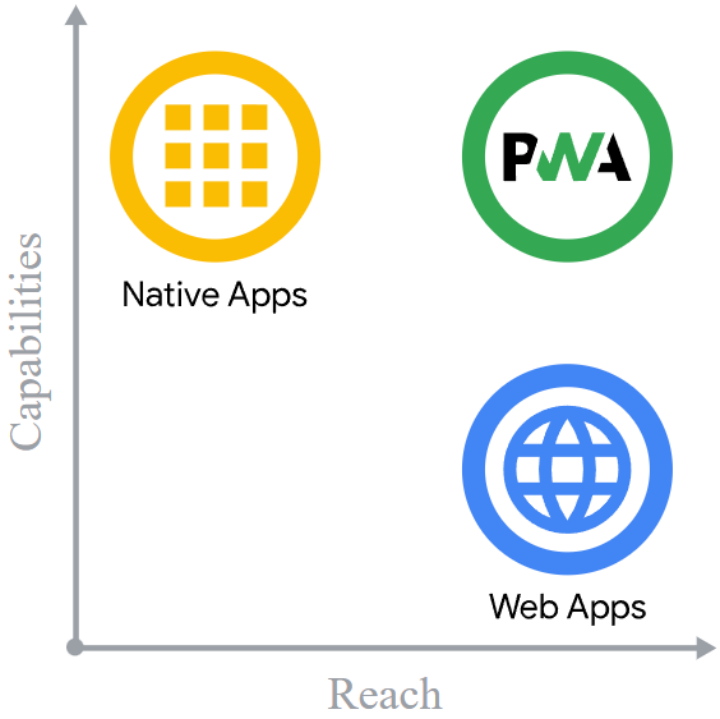
- Multiplataforma [9]: per tal que es pugui executar en qualsevol sistema operatiu que tingui un navegador i connexió a Internet

- Disseny adaptatiu/responsiu (*responsive*) [10]: per poder-se visualitzar correctament en diferents mides de dispositiu -tot i que per la seva tipologia probablement sigui més còmode imaginar-lo en pantalles d'ordinador i/o tauletes.

- Adaptable/Escalable [11]: l'estructura de dades s'ha pensat per poder carregar diferents arxius en funció del Grau a cursar -tot i que faltaria parametritzar-los.

Aquest tres casos fan recomanable el desenvolupament d'una Progressive Web App, que ens permetrà arribar a més dispositius diferents que una app nativa i, a més, ens donarà accés a funcionalitats clau d'aquest tipus d'apps sense necessitat d'aprendre els llenguatges de cada plataforma, sense deixar de ser compatibles amb dispositius

que no tinguin suport per a les funcionalitats avançades. Aquesta informació i el següent gràfic s'han extret de [12].



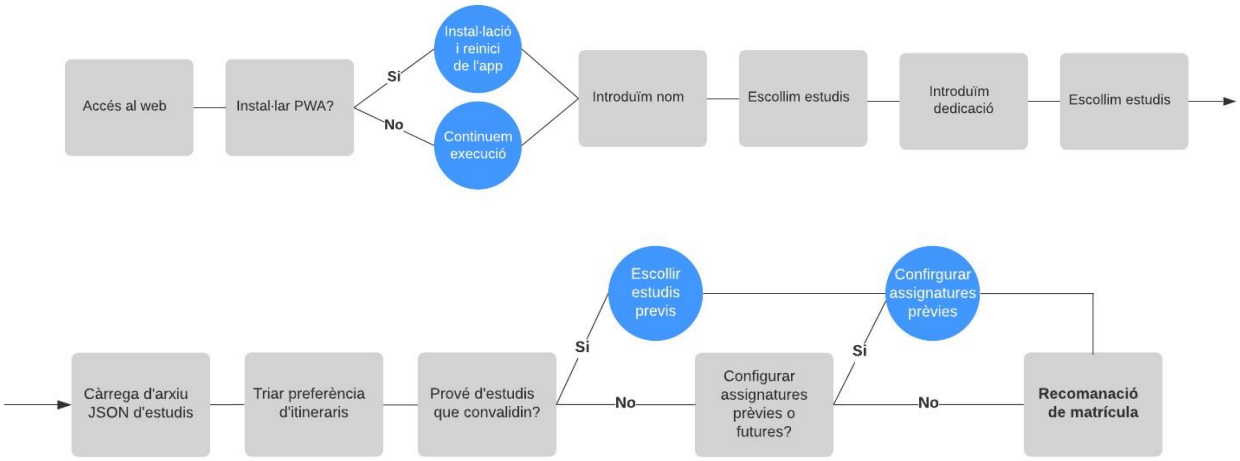
Amb aquesta representació gràfica veiem com les aplicacions natives ofereixen més funcionalitats, mentre un web convencional (o Web App), ens permetrà arribar a més usuaris perdent funcionalitats. Així, l'ús de la tecnologia de les Progressive Web Apps ens permet tenir accés a opcions molt diverses, com l'ús de la memòria cau amb control avançat, la instal·labilitat de l'eina com si fos una aplicació en qualsevol dispositiu compatible o fins i tot l'ús de notificacions *push*, però sense renunciar a tenir un abast més gran amb una sola versió de desenvolupament.

Això ens porta a imaginar com seria el funcionament desitjat de l'eina:

1. L'usuari visita una adreça URL amb el seu navegador. El navegador li ofereix l'opció d'"instal·lar" la webapp -posant un accés directe al seu dispositiu i desant en la memòria cau els principals recursos per funcionar fora de línia. L'usuari pot instal·lar-ho o no, funcionarà igualment.
2. L'usuari ha de respondre unes primeres dades:
    a. El seu nom
    b. Les hores setmanals de disponibilitat
    c. Els estudis que vol cursar
3. Quan l'usuari accepti, el sistema carregarà l'arxiu JSON referent als estudis escollits i li sol·licitarà que ordeni per ordre de preferència els itineraris a seguir.
4. Quan l'usuari accepti aquesta ordenació, se li demanarà si prové d'uns estudis que convalidin.

5. Si respon que si, se li sol·licitaran quins estudis.
6. Si a 4 ha respost que no, o si ja ha respost el 5, l'usuari haurà de respondre si ja ha cursat assignatures prèviament.
7. Si diu que si, se li mostrarà un desplegable d'assignatures, n'escollirà una i marcarà el semestre en què l'ha fet i la nota que va treure. Podrà escollir les que necessiti successivament i finalment acceptarà la configuració.
8. Quan hagi respost 7, o si respon No a 6, l'usuari ja haurà completat la primera configuració i procedirà a veure la recomanació de matrícula.
9. Des de la recomanació de matrícula, l'usuari podrà modificar les següents dades i es recalcularà la recomanació de matrícula:
    a. El seu nom i hores de disponibilitat.
    b. El grau a cursar -en aquest cas, tornarà a la configuració inicial.
    c. La preferència dels itineraris.
    d. Els estudis previs per a convalidacions.
    e. Les assignatures superades, suspeses.

Podríem descriure gràficament aquest funcionament a partir del següent diagrama:



4. Tecnologies, *frameworks* i programari

4.1 Tecnologies que utilitzarem

Volem aconseguir disposar d'una eina que sigui autònoma -i no necessiti un servidor potent-, de manera que optarem per desenvolupar-ho en HTML, CSS i Javascript en les dades en format JSON. Anem a descriure aquestes tecnologies:

HTML (Hypertext Markup Language): Es tracta d'un llenguatge de marques destinat a l'elaboració de pàgines web. L'estàndard, a càrrec del World Wide Web Consortium (W3C) ha arribat a la versió 5 i ofereix la capacitat de definir objectes i estructures concretes que un navegador ha de saber com mostrar -en general, l'HTML s'encarrega dels continguts. Més informació a [13].

L'HTML té la particularitat de ser molt utilitzat i multiplataforma -pel fet de dependre d'uns navegadors que existeixen per a tots els sistemes operatius i que amb els anys s'han convertit en un dels programes essencials. Per això, en el cas d'aquest projecte, crec que és l'elecció més adequada com a base.

CSS (Cascading Style Sheets): és un llenguatge de fulls d'estil que permet descriure la presentació del contingut d'un llenguatge de marques (en el nostre cas, HTML). CSS es troba a la versió 3 i el manté el CSS Working Group. Utilitzarem, doncs, el CSS tant a nivell de presentació com d'accessibilitat, ja que permet mostrar estils diferents basat en mides de pantalla, per exemple. Més informació a [14].

El CSS és pràcticament ja una part més de l'HTML, per tal de marcar els estils i deixar que l'HTML s'ocupi únicament del contingut. Això facilita l'accessibilitat i el disseny adaptatiu o *responsive*.

Javascript: Es tracta d'un llenguatge script basat en objectes, iniciat per Netscape Communications Corp i que després va evolucionar cap a ECMAScript, l'actual estàndard. Descriure'l a fons seria molt complex -com ho és el propi llenguatge-, però en el nostre cas l'utilitzarem tant per la manipulació de la interfície (modificar continguts HTML o estils CSS) com per a realitzar els càlculs propis necessaris en cada cas. Més informació a [15].

El Javascript, com el CSS, s'ha convertit en un acompanyant inseparable de l'HTML, ocupant-se de la interactivitat de les pàgines. Això ens permet intentar treballar sense necessitat de llenguatges de servidor ni de programari especial, de manera que abaratirem el manteniment de l'eina i facilitarem les tasques d'edició i visualització.

JSON (JavaScript Object Notation): És un estàndard obert (especificació pública que permet assolir una tasca específica) basat en arxiu de text

per a l'intercanvi de dades que empra text llegible per humans. El JSON consta de parells atribut-valor i matrius de dades, a mode d'objectes. Algunes bases de dades conegudes com a NoSQL (perquè no són estrictament relacionals) utilitzen JSON com a format d'emmagatzematge i intercanvi.

Cal tenir en compte que SQL (Structured Query Language) ofereix modes de consulta a partir de sentències o *queries*. Això permet fer cerques amb frases com "SELECT * FROM table WHERE name='John'", una opció que en el cas de bases de dades NoSQL es pot substituir per cerques a partir de plantilles (objectes JSON semiomplerts per tal que es retornin tots aquells que tinguin valors iguals en els camps que s'envien plens, etc...).

Semànticament, el text engloba objectes entre claus { i }, mentre que les matrius (que poden ser d'elements o d'objectes) estan entre claudàtors [i]. Els objectes estan formats per parells d'una clau i un valor. S'utilitza una coma per separar objectes, matrius, parells o valors independents, i en el cas de JSON l'estàndard no admet les anomenades "trailing comma", comes que es deixen darrere l'últim element per no oblidar-les si s'inclouen nous elements al final. Tampoc s'admeten comentaris.
Un exemple de JSON seria el següent:

```
{
  "clau1": "valor1",
  "matriuDeValors1": ["valor1","valor2","valor3"],
  "matriudObjectes1":[
    { "clauObjecte1":"valorObjecte1"},
    { "clauObjecte2":"valorObjecte2"}
  ]
}
```

Més informació a [16].

En el meu cas, JSON és un llenguatge que com ja deia al principi de l'explicació és llegible per a humans i en arxius de text. Això en facilita l'edició i la lectura (en l'exemple he intentat mantenir una forma visual fàcil de seguir) i a més existeixen múltiples validadors [17] que a més d'avaluar que sigui correcte també el mostra de manera que sigui fàcil de llegir. A més, és compatible amb Javascript i qualsevol navegador pot accedir a arxius JSON, carregar-ne el contingut en variables i manipular-los.

És per tot això que en el meu cas el faré servir com a format per a desar l'estructura de dades parametritzades per als estudis, de manera que siguin el més fàcils de crear possible.

Aquest format es pot, a més, desar al navegador gràcies a algunes funcionalitats de Javascript, en concret **localStorage** [18] amb opcions

per controlar la compatibilitat com l'API de StorageManager [19] per advertir l'usuari d'incompatibilitats.

L'únic problema que haurem d'afrontar és que localStorage permet desar al navegador les dades en parells de clau i valor, i ambdues han de ser cadenes de text (*strings*). És per això que, a l'hora de fer la manipulació de les dades, necessitarem:

- La funció JSON.stringify, que converteix un objecte JSON en una cadena de text, per tal de tenir un valor per al parell en format *string* per poder-lo desar.

- La funció JSON.parse, que pren una cadena de text i, si el format és correcte, la converteix en un objecte JSON.

4.2 *Frameworks* de disseny i desenvolupament

A nivell de *frameworks* per a facilitar-nos les tasques de disseny i desenvolupament, hem utilitzat principalment Bootstrap, JQuery i JQuery UI.

Bootstrap [20] és un paquet que inclou principalment un disseny predefinit seguint diversos estàndards d'accessibilitat, i que divideix la pantalla en una sèrie de graelles que s'ajusten a l'amplada. Amb la versió emprada, Boostrap 5, que és la més recent, hem pogut estalviar-nos la tasca de definir tots els estils de base i uns colors per defecte, i hem pogut començar a treballar directament sobre el contingut. Per les primeres proves l'havia incorporat des d'un CDN propi de Bootstrap, però finalment he optat per incloure els arxius necessaris al paquet de l'entregable.

JQuery [21] és una llibreria de Javascript molt completa i extesa que facilita la manipulació del DOM (la interfície del web) i pot conviure amb codi Javascript. Tot i que el projecte s'hauria pogut treballar sense aquest *framework,* si que és cert que en facilita la lectura del codi i ha permès incorporar, per exemple, més fàcilment altres opcions que a continuació comentaré.

JQuery UI [22] és un afegitó de JQuery que aporta algunes millores en la gestió de la interfície d'usuari (UI, de *User Interface*). En el nostre cas ens interessava per poder fer la ordenació de preferència d'itineraris mitjançant el seu objecte *sortable*. Malgrat tot, hem necessitat també l'ampliació externa JQuery UI Touch Punch [23] per tal de fer-ho funcionar en dispositius que no tenen ratolí.

Lodash [24] és una llibreria similar a JQuery, que facilita les crides complexes de Javascript, i que en el meu cas era un requisit de JointJS, que explicaré a continuació.

BackBone [25] també és una llibreria d'ajuda de Javascript, que també és necessària per a JointJS, i que ens dona suport a estructures de dades proveïnt models de parells clau-valor amb esdeveniments personalitzats i facilitant la connexió de dades amb objectes de la interfície.

JointJS [26] en la seva versió OpenSource ofereix l'opció de fer gràfiques molt simples i amb molt poc codi -donant importància al contingut i oferint una sèrie de predefinicions gràfiques que m'han estat molt útils. Per a utilitzar-lo, cal incorporar prèviament algunes llibreries ja citades, com JQuery, Lodash o Backbone.

Per a la realització del treball, a més de la extensió per a Chrome "**Web Server for Chrome**" [27], també hem utilitzat un servidor d'allotjament d'Arsys [28] vinculat a un domini de la meva empresa, Sobrevia.net [29], que ja té un certificat SSL actiu mitjançant l'ús de Cloudflare [30] en el seu pla gratuït.

La versió en línia es pot visitar a l'adreça https://www.sobrevia.net/tfg.

4.3 Programari complementari

A nivell de programari utilitzarem els següents:

- **Visual Studio Code** [31] com a editor de textes per a HTML, CSS, Javascript i JSON. Es tracta d'un potent editor que inclou moltes possibilitats de personalització, que és gratuït i és de Microsoft. És el que utilitzo més sovint (juntament amb Notepad++) i en el que em sento més còmode. Disposa de moltes extensions que permeten, entre d'altres, corregir i acolorir el codi, autompletar per a diferents llenguatges, facilitar la mostra i edició d'arxius JSON... de manera que ens serà molt útil.

- **FileZilla** [32] com a client FTP. Com que necessitarem publicar-ho per provar la funcionalitat de PWA en un servidor amb SSL, utilitzarem aquest client FTP per tal de penjar els arxius al servidor on es podran provar correctament. FileZilla és un client de FTP i SFTP molt complet i també gratuït, que es pot integrar fàcilment amb qualsevol editor per fer que s'obrin arxius automàticament amb un programa determinat.

- Com que necessitem un entorn amb certificat SSL, he optat per utilitzar un **servidor web ja existent**. Podríem instal·lar un servidor Apache o NGinx, entre d'altres, i un certificat amb Let's Encrypt, però això implicaria més hores i ja anem justos de temps per a crear una màquina virtual Linux amb tot el programari associat. Com que disposo d'accés a un servidor dedicat CloudBuilder de la companyia Arsys [28], amb Linux com a sistema operatiu i Apache amb un certificat SSL vinculat a través de Cloudflare [30] -de manera que surt gratuït com un Let's Encrypt [33]-, el farem servir per fer les proves ja que no ens implica cap feina ni cost addicional.



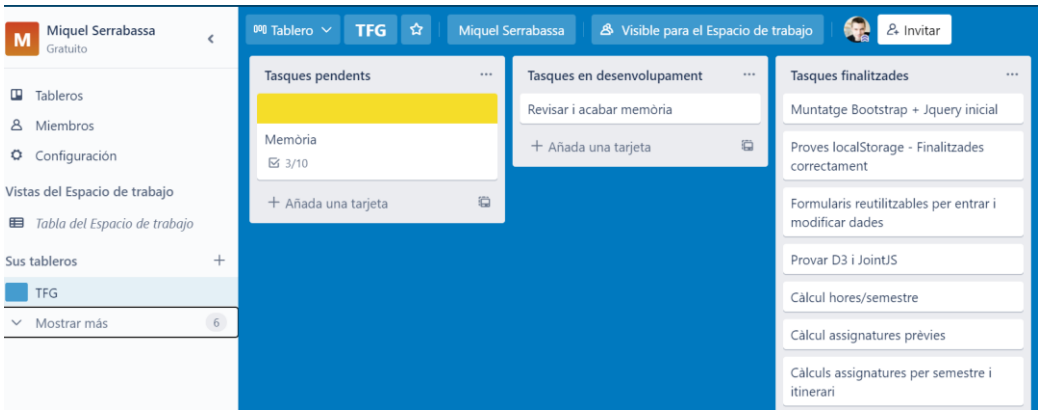
- **Navegadors** per provar i comprovar que funciona en diferents entorns. Utilitzare principalment Google Chrome [34], així com Mozilla Firefox [35] i Microsoft Edge [36]. En el meu cas, faré servir Chrome com a navegador principal perquè amb les **DevTools** (F12) hi estic acostumat a treballar i em serviran per a detectar possibles problemes tant en l'HTML, CSS com en el Javascript o el JSON, i després faré les proves pertinents a la resta de navegadors. Cal destacar que, com que les darreres versions de Microsoft Edge estan basades en Chromium, és pràcticament el mateix navegador a nivell de funcionalitats HTML, CSS i Javascript.

- **Adobe Photoshop** [37] com a aplicació per a generar material gràfic que necessitem. Probablement, haurem d'establir algun logotip -encara que sigui només de mostra-, capturar algun color,

etc... i això amb aquesta eina ens serà molt senzill -i a més, la tinc instal·lada al meu ordinador de manera que també em facilita la feina de posar-me a desenvolupar.

- **Trello** [38]: A través d'aquesta eina he pogut organitzar les tasques segons la metodologia escollida, àgil i amb KanBan, a mode de targetes. Trello ofereix la possibilitat de crear taulers, amb diferents llistes (en el meu cas, he escollit utilitzar una llista de tasques pendents, una altra on tenir-hi coses concretes que estava desenvolupant per poder tenir clar els diferents fronts oberts, i finalment una llista de tasques ja realitzades on anar-hi deixant les targetes finalitzades).

A mode d'exemple, el tauler a unes setmanes del lliurament i amb la part de desenvolupament finalitzada es veu així:



Hi ha altres eines similars com Asana, que fan tasques semblants, i totes elles tenen versions gratuïtes o de pagament. En el cas de Trello, que porto utilitzant ja alguns anys, l'opció gratuïta és suficient, i el fet que sigui propietat de la companyia Atlassian [39] (al darrere també de Jira, una eina de documentació i de treball amb un ús corporatiu molt extès) li dóna encara més fiabilitat, aparentment.

- PlantUML: Per a generar fàcilment diagrames UML a partir d'un arxiu JSON vaig descobrir aquesta eina. En realitat, és especialment útil sobretot com a aplicació, però al seu web hi ha l'opció de provar-ho en línia [40] amb una sèrie de paràmetres que s'admeten, entre els que s'hi troba @startjson i @endjson amb una estructura de dades JSON vàlida. És amb aquesta eina que he pogut generar els PNG amb diagrames UML de les estructures de dades dels arxius JSON.

- LucidCharts [41]**:** A l'hora d'elaborar altres gràfiques, també m'ha estat molt útil la versió gratuïta -de proves- de LucidCharts, a nivell de diagrames de decisió i de casos d'ús per incloure en aquesta memòria.

5. Arquitectura

5.1 Definició de l'arquitectura

L'eina està pensada per a poder-se utilitzar en qualsevol plataforma que disposi d'un navegador web. Així, ho tractarem inicialment com una aplicació web (*WebApp*), sobre la que hi afegirem una sèrie de funcionalitats convertint-la, allà on sigui possible, en una Progressive Web App o PWA.

Això implicarà que, com a eina, podrà funcionar en un servidor sense massa requisits especials a nivell tècnic, amb les dades emmagatzemades en arxius JSON, per tal de no necessitar servidors de bases de dades, APIs o altres punts intermediaris. En resum, l'entregable serà un conjunt d'arxius HTML, Javascript i JSON (que en el fons és un format d'objectes de Javascript), així com fulles d'estils CSS o imatges.

Sí que necessitarem un certificat SSL instal·lat al servidor on hi publiquem els arxius per tal que la WebApp funcioni com a PWA i es pugui instal·lar. La decisió de fer-ho així ens ofereix la possibilitat de tenir més escalabilitat i portabilitat: fins i tot podríem executar-ho en local, ja que en realitat, la funcionalitat de PWA -instal·lació i memòria cau- és la que ens requereix el certificat SSL, però si tenim una carpeta amb els arxius al disc dur també podrem utilitzar-lo amb la instal·lació d'una extensió de Chrome de funcionament molt senzill.

D'aquesta manera, només haurem de disposar d'una sèrie d'arxius JSON que parametritzin els diferents graus.

Així, doncs, disposarem de diferents opcions de funcionament a partir de l'entregable.

L'entregable en si mateix serà un arxiu comprimit que contindrà:

- Un arxiu index.html que contindrà el marcat HTML de l'eina així com les crides als arxius CSS i JS
- Un arxiu manifest.json a l'arrel amb les dades relatives a la PWA.
- Un arxiu .css que contindrà les definicions d'estils de l'eina
- Un arxiu .js que contindrà la programació de les rutines de càlcul i interacció de l'eina
- Una carpeta JSON que contindrà arxius en format JSON
    - Config.json, un arxiu que llistarà la resta de JSON vinculats a estudis, indicant el nom de cada arxiu i els estudis del qual conté les dades
    - Un arxiu JSON per a cada grau parametritzat. En el nostre cas, GEI.json, pel Grau d'Enginyeria informàtica.
- Una sèrie d'arxius d'imatges per a la part gràfica de l'eina.

- Podem incloure-hi també un arxiu d'instruccions en mode text (readme.txt) amb els punts principals per a fer-ho funcionar o enllaços a informació per modificar els JSON dels graus.

Un cop disposem de l'entregable, tindrem dues opcions:

1. Descomprimir en una carpeta i executar
    a. En aquest cas, funcionarà en local; no caldrà instal·lar-lo com a PWA i haurem de revisar-ne les funcionalitats per si alguna fallés. En tot cas, la principal diferència serà que els arxius JSON dels graus els haurà de mantenir l'usuari que ho utilitzi -potser escrivint-los ell mateix, o baixant-los d'algun web. També és important destacar que, degut a les restriccions de seguretat dels navegadors, serà necessari fer crides mitjançant http. Per solventar els problemes que això pot provocar, es recomana l'ús de l'extensió de Chrome "Web Server for Chrome" [27] amb la que es pot obrir un servidor web local a partir d'una carpeta de manera molt simple. Això és necessari perquè els navegadors actuals no permeten obrir arxius del disc dur directament, de manera que no es podrien llegir els arxius JSON de parametrització.

2. Publicar els arxius en un FTP.
    a. En aquest cas és necessari disposar d'un servidor que admeti arxius HTML (Apache, NGinx, i la gran majoria) i sobretot, tenir un certificat SSL per a fer funcionar la PWA - en cas contrari, funcionaria com en local.
    b. El servidor pot estar ubicat en una empresa de *hosting* o en una màquina virtual o un ordinador de la xarxa local, per exemple.
    c. El certificat SSL pot ser comercial o gratuït -si es té accés a instal·lar LetsEncrypt [33] o a utilitzar Cloudflare [30] o similars.

D'aquesta manera, tenim una eina funcional en pràcticament qualsevol entorn informàtic que disposi d'un navegador web i una interfície gràfica.

Entraré en detall ara també sobre els certificats SSL que he mencionat anteriorment, ja que darrerament, pels requisits imposats principalment per Google amb Chrome i altres navegadors, s'han tornat un recurs pràcticament indispensable a la xarxa. El terme SSL prové de l'anglès Secure Sockets Layer [42], i es refereix a certificats de seguretat (inclosos aquells amb TLS o altres tecnologies més recents) que asseguren una connexió entre dos ordinadors connectats a la xarxa, essent-ne un l'hoste (host, servidor) i l'altre el que hi fa una visita o consulta. El seu ús està molt extès en diversos àmbits (web, correu, VPNs...) però ens centrarem amb el que ens interessa, el de les pàgines web. Es tracta d'un arxiu de dades mitjançant una clau pública i una

privada. Així, la clau pública, que tothom pot conèixer, la utilitzarà el visitant per enviar missatges xifrats al servidor, que podrà desxifrar-los només mitjançant la clau privada, de la que només ell disposa. Aquesta capa de seguretat, que fa que les comunicacions web siguin segures i secretes, és a hores d'ara ja un estàndard que pràcticament es demana en qualsevol entorn web.

Font: [57]

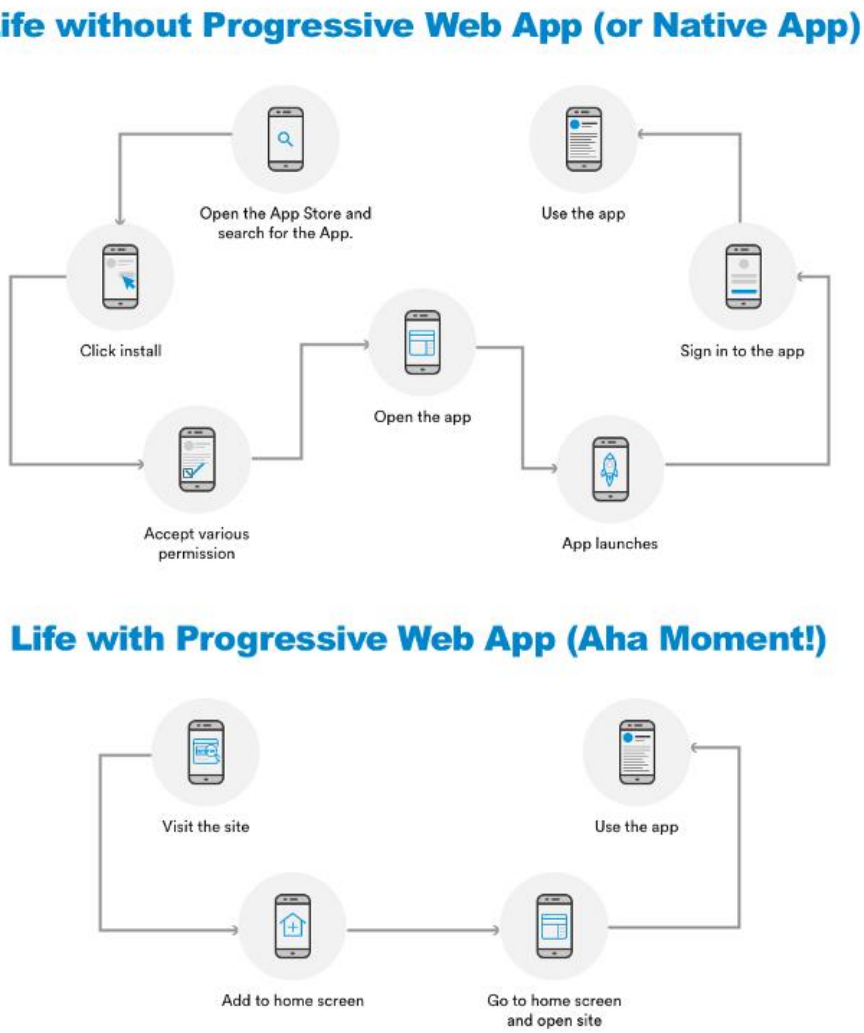
Els certificats SSL es poden adquirir -per un pagament mensual o anual- a diverses empreses certificadores. Això és especialment útil per a grans corporacions, entitats financeres, administracions, o empreses que treballen amb dades que cal protegir especialment, de manera que més enllà de la pròpia protecció en la comunicació, s'assegura que el web que es visita pertany a una entitat concreta, i que ho certifica una altra entitat de certificació real i verificable.

Més enllà d'aquest servei, han començat a créixer opcions gratuïtes o fàcils d'implementar. Una d'elles és a través del servei Cloudflare [30] -que té una versió gratuïta-, i que ofereixen certificats simples per assegurar les comunicacions web. Per altra banda, també existeix LetsEncrypt [33] que és un servei de certificació gratuït mitjançant un programari que es pot instal·lar en un servidor web.

5.2 PWA: diferències de funcionament com a Progressive Web App

La clau del concepte de les PWA (Progressive Web Apps) és el concepte de millora progressiva (o Progressive enhancement [43]): estem parlant d'una estratègia de disseny web que posa èmfasi en l'accessibilitat del contingut en primer lloc, per davant del disseny i les funcionalitats, a diferència de l'estratègia *graceful degradation* [44] que es basa en pensar el web per a dispositius moderns i després adaptar-lo "cap enrere" perquè no falli en versions més antigues.

Amb això, doncs, s'obté una eina que pot tenir funcionalitats addicionals -com podrien ser notificacions push, accés al programari del dispositiu, treball sense connexió, possibilitat d'instal·lar un accés directe com si fos una aplicació nativa...- que si no són crítiques pel funcionament de l'eina, es poden obviar en aquells casos en què no sigui necessari, fent que funcioni tota la resta que sigui compatible amb al configuració actual.

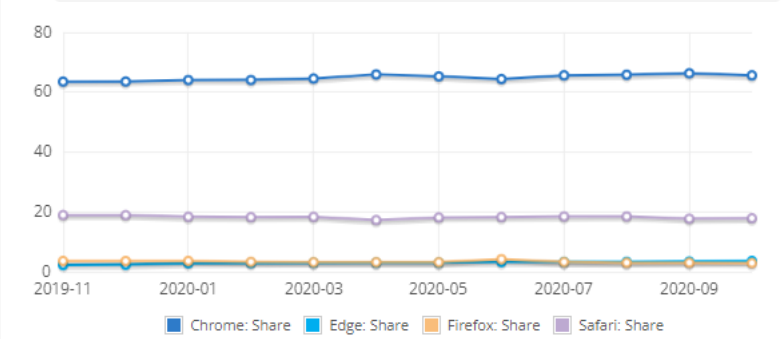


Imatge de Krishankant Singhal [58]

Cal tenir en compte que, malgrat que aquesta és una estratègia de disseny que no és nova (les primeres referències a aplicacions progressives ja provenen de 2015 [45], encara hi ha una certa manca de suport a totes les funcionalitats, sobretot perquè faciliten la posada en marxa d'aplicacions sense necessitat de passar per les "botigues" habituals. Això implica que les empreses comercialitzadores de dispositius i programari com Apple o Google, per un costat, perden control econòmic sobre l'ús de les seves plataformes (i en el cas d'Apple, el judici amb Epic Games és un clar exemple que mou xifres estratosfèriques de diners [46]), i per l'altre, també perden la possibilitat de vetllar per la seguretat dels usuaris (tot i que cada dia trobem notícies

referents a *malware* en aplicacions pujades a les botigues oficials [47] o errors 0-day [48]) .

En tot cas, actualment el suport de les funcionalitats creix cada cop més gràcies, en part, a l'adopció de Chrome i Chromium (que ofereixen suport en tots els dispositius on poden utilitzar el seu motor: Windows, Linux, macOS i Android), mentre que en iOS només Safari ofereix aquesta opció a partir de la versió 11.3 amb algunes restriccions. Qui, sorprenentment, no ho ofereix, és Firefox en la seva versió d'escriptori. Tot i això, entre octubre i novembre de 2021, la quota de mercat dels navegadors era la següent:



Show 10 entries Search:

Browser	Share
Chrome	65.01%
Safari	18.22%
Firefox	3.32%
Edge	3.04%
Samsung Browser	2.73%

Font: NetMarketShare [49]

De manera que a nivell percentual, el suport és ja força gran.

A nivell tècnic, cal tenir en compte que la majoria de les funcionalitats pròpies de les PWA les podrem gaudir amb els anomenats *Service workers* [50]: permeten contingut fora de línia, sincronització en segon pla en temps real, notificacions push... Que tot i que en gran part no utilitzaré inicialment en aquest TFG, fan que si l'opció es vol ampliar, es disposi d'un camp ampli per recórrer: estem, en realitat, desenvolupant una aplicació multiplataforma que tindrà suport per a múltiples funcionalitats i accés a eines del dispositiu -ubicació geogràfica, etc.... De manera que hi ha opcions que ara no ens podem imaginar però que podríem arribar a utilitzar si detectem les necessitats, que poden anar molt més enllà d'un web tradicional i que, a més, ens eviten haver d'estar treballant en versions diferents d'una mateixa app en diferents llenguatges de programació i plataformes que ens demanaran taxes per a la seva posada en marxa.

Per tal de començar a posar-ho en marxa, doncs, disposarem d'una crida al principi del codi Javascript, quan la pàgina es carrega, que informarà al navegador tot registrant el Service Worker, que serà un altre arxiu Javascript amb una sèrie d'instruccions que ja anirem veient.

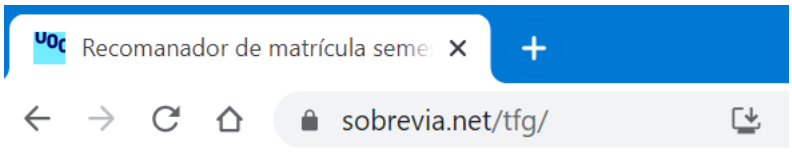
Per fer aquest registre, primer es comprova que el navegador sigui compatible amb els *service workers* i després s'envia la comanda navigator.serviceWorker.register juntament amb la ruta de l'arxiu i el camí relatiu on s'executarà.

Per continuar amb el nostre codi, seguirem l'exemple que podem trobar a [51], i que ofereix l'opció de guardar en memòria cau els arxius locals propis per tal de poder realitzar la instal·lació, tot oferint una manera fàcil d'actualitzar-ho per forçar reinstal·lacions en cas d'haver de canviar arxius.

Aquest exemple ens ofereix per un costat, doncs, l'opció de desar una sèrie d'arxius en memòria cau per tal de tenir-los precarregats a l'iniciar. Això serà, doncs, la llista de tots els arxius que tinguem. Finalment, també ofereix l'opció d'anar desant en memòria cau temporal les crides que es facin a recursos que no haguem inclòs en aquesta llista -per exemple, crides externes-.

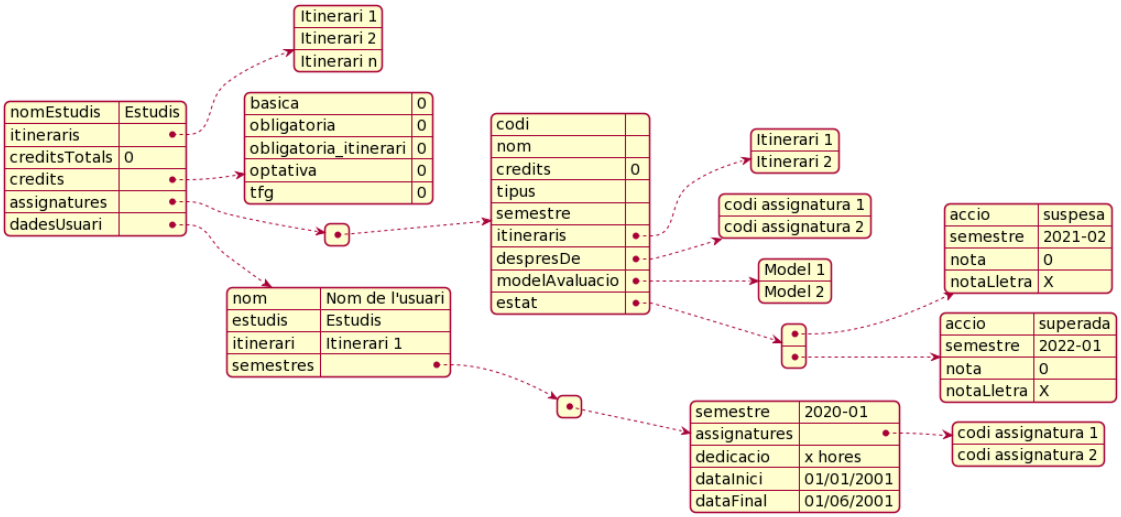
Per fer-ho, captura alguns tipus de crida com ara Install per a desar els arxius a la precàrrega, "activate" per detectar quan cal actualitzar els arxius de la memòria cau, i "fetch", que és quan captura les crides a la xarxa i, si té l'arxiu en memòria cau, evita que es faci la crida fora de la memòria cau, mentre que si no el detecta, el desa a la llista d'arxius en temps d'execució capturant la resposta per tenir-la també ja desada a la memòria cau provisional.

Un cop preparat tot el *service worker* podem verificar el funcionament tant a les *devTools* de Chrome o Edge com veient que a la barra de tasques ja hi figura la icona que es pot clicar per instal·lar la PWA perquè funcioni fora de línia:



5.3 Parametrització JSON i variables en temps d'execució

Per assolir l'objectiu de poder desar els estudis parametritzats i poder realitzar els càlculs necessaris, vaig elaborar la següent estructura de dades:

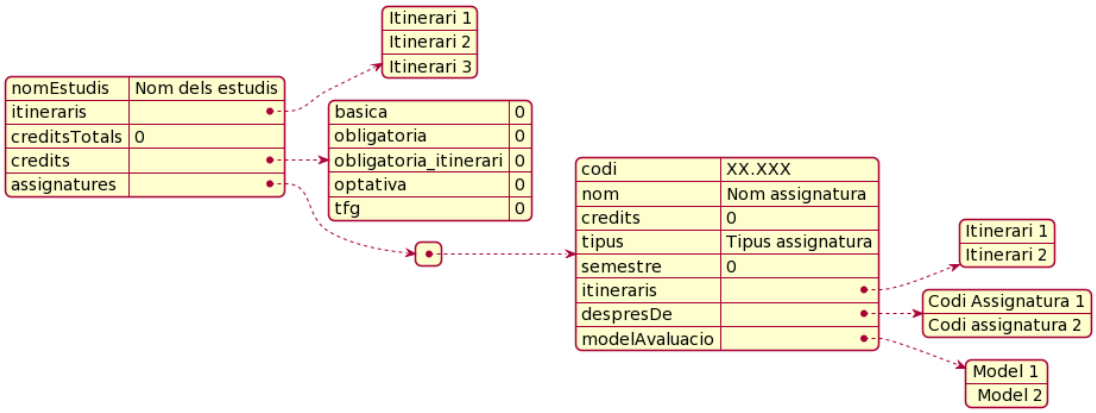


Aquesta parametrització es basa en un sol objecte JSON que conté una estructura diferenciada per diversos grups de dades:

- La informació general dels estudis, que és la informació que podríem anomenar “mestra” ja que tota la resta de dades hi estan vinculades.
- La informació relativa als itineraris (que és una llista de noms d’itinerari, a itineraris).
- La informació relativa al número de credits segons tipologia d’assignatura (credits)
- Les dades de l’usuari, que inclouen informació que haurem demanat a l’usuari per poder fer els càlculs necessaris, així com les seves matriculacions prèvies (dins de dadesUsuari/semestres)
- Les dades d’assignatures. Aquesta part és la més complexa, i inclou:
    - El codi, nom i número de crèdits de l’assignatura
    - El tipus per saber si és obligatòria, bàsica o optativa
    - Els semestres en què es cursa (tots, setembre o febrer)
    - Itineraris dels quals forma part (i llavors passa d’optativa a obligatòria)
    - Recomanacions d’assignatures que cal haver fet abans (despresDe[codis d’assignatures,...]).
    - El model o models d’avaluació per saber si tindrà examen o prova de síntesi, o si es pot superar amb AC o Pràctica.
    - L’estat, que és la llista d’intents que ja s’han fet (i coincidirà amb les assignatures matriculades a dadesUsuari/semestres). En guardarem l’acció (superada, convalidada o suspesa), el semestre i la nota numèrica en en lletra.

Si bé és cert que he pogut treballar sempre amb aquesta estructura, finalment hi ha certs grups de dades que no s’utilitzen, com ara el d'estat o el de dades d’usuari, ja que finalment he mogut aquestes estructures al *localStorage* per tal de treballar-hi en temps real i desar-les al

navegador. Així, doncs, finalment l'estructura de l'arxiu JSON de cada estudis és la següent:



Amb aquesta estructura ja definitiva, podríem clonar-ho per a múltiples estudis. Com a part de l'entregable hi ha precisament l'arxiu GEI.json que conté tota la informació relativa als estudis del Grau d'Enginyeria Informàtica de la UOC.

Tota la informació que s'utilitza és pública i es pot trobar a la informació de les assignatures al web de la UOC, de manera que mitjançant *scrapping* o una possible API en un futur es podria arribar a construir automàticament i de forma simple, algun tipus d'assistent de creació o actualització dels JSONs d'estudis. Així, les vinculacions entre els diferents objectes són mitjançant noms en el cas d'Itineraris, i codis en les assignatures. Això implica que no necessitem generar identificadors ni altres valors que no preexistissin i que és senzill de fer canvis i d'implementar.

A més, de cara a que la parametrització serveixi realment per a més d'uns estudis, optem per un altre arxiu JSON que conté parells amb el nom dels estudis i un nom d'arxiu. Això, que inicialment servia per tenir diversos estudis a escollir, ho he ampliat també per a preveure noves funcionalitats que puguin necessitar també de parametrització amb arxius JSON. Tot i que el catàleg d'opcions podria ser molt llarg, he optat per una funcionalitat que he cregut simple d'implementar però que, ben parametritzada pot aportar valor: les convalidacions.

Així doncs, l'estructura és molt simple i inicialment es divideix en dos grups: Graus i Convalidacions. Cada grup conté una llista de parells de textos: la clau conté el nom dels estudis i el valor és el nom de l'arxiu on estan parametritzats:

```
{
   "Graus": {
      "Grau d'Enginyeria Informàtica": "GEI.json",
      "Prova": "prova.json"
```

```
    },
    "Convalidacions": {
        "FPII DAM/DAW": "conv_DAM.json"
    }
}
```

Aquest arxiu config.json està pensat per ser el repositori d'arxius de dades de la instal·lació. Inicialment només incloïa una estructura simple amb noms d'arxiu i el nom dels estudis que parametritzaven. Però per fer-ho més modular, hem creat un altre nivell que permet indicar el tipus d'arxiu. D'aquesta manera, l'estructura del JSON pot créixer indefinidament amb nous tipus d'arxiu -que, tot i això, després caldria programar les funcionalitats incloses. De moment, el desenvolupament inclourà tant la parametrització d'estudis -assignatures, etc...- com un nou nivell de convalidacions, amb uns arxius que només han d'informar dels codis d'assignatura que convaliden uns estudis determinats.

Finalment, sobre l'objecte que ja estàvem apuntant referent a les convalidacions, es tractarà d'un arxiu que durà només una llista de codis d'assignatura que queden convalidats pels estudis als que defineixen. L'exemple de l'arxiu conv_DAM.json (que són els estudis pels quals vaig accedir jo a la UOC i per tant m'era molt més fàcil obtenir la llista de codis al tenir-los directament a l'expedient):

```
["05.554","05.561","05.564","05.567","05.575","05.590","05.596","05.615"
]
```

Veiem que només llistem, en format JSON, una sèrie de codis.

A més d'aquestes parametritzacions, l'eina també inclou objectes que es desen a l'emmagatzematge local del navegador (*localStorage*). Com que el localStorage només admet parells clau i valor on la clau i el valor han de ser cadenes de text, hi desem objectes en format JSON convertits a cadena de text. Això es pot fer gràcies a les funcions JSON.stringify i JSON.parse.

Aquests objectes són:

- dedicacio: Conté un número amb les hores setmanals
- estudis: Conté el nom d'arxiu dels estudis escollits.
- nomUsuari: Conté el nom que ha indicat l'usuari en el procés de configuració
- itinerarisEscollits: conté una llista itemX,itemY,itemZ, que manté l'ordre de preferència dels itineraris dels estudis. Cada número que acompanya la paraula item identifica l'ordre per defecte dels itineraris, com si en fos un identificador.
- estatsAssignatura: els estats de les assignatures preseleccionades per l'usuari. Aquest objecte es desa com a cadena de text, però convertit a JSON conté, per a cada assignatura que s'ha indicat a la configuració:

- Codi de l'assignatura
- Acció (convalidada, superada, suspesa, o per matricular en un semestre concret)
- Nota (si cal)
- Semestre (en format AAAA-S).

Finalment, enumeraré aquí també les estructures de dades en forma de matrius o variables que s'utilitzen per a la realització dels càlculs, explicada més endavant, per tal de donar context als propis càlculs, així com a la disposició i presentació de la interfície.

Per un costat, disposem d'una sèrie de matrius que utilitzem per a poder desar versions curtes o simplificades de cadenes de text (per exemple, per obviar accents que podríen dur problemes de codificació), i que ens les tradueixen:

- SemestresEscrits conté els noms dels semestres (Febrer-Juny i Setembre-Febrer)
- pluralTipusEscrits i singularTipusEscrits conté els noms dels tipus d'assignatura en plural i singular, i amb els accents i majúscules que corresponen.
- accionsEscrites, per escriure -també amb colors- les diferents accions que poden tenir les assignatures configurades per l'usuari.

Tenim també una variable *endStep* que utilitzem com a semàfor, per tal de fer funcionar les opcions configurables com a passos únics o com a un assistent. Així, si s'obre un dels passos de la configuració amb endStep amb 1 com a valor, no es va al següent pas sinó que es desen les dades i es torna a la interfície de proposta.

Existeix també una variable horesSetmanalsPerAssignatura, amb valor 10 per defecte, que calcula quantes hores setmanals cal destinar a una assignatura segons els càlculs que s'expliquen més endavant, però que al tenir-ho aquí parametritzat ens permet canviar-ho fàcilment.

Tenim també altres variables de comptador com estatsAssig, per tenir un comptador d'estats d'assignatures afegits -així podem utilitzar aquest número per a gestionar les interaccions de l'usuari amb cada línia de forma unívoca-, i altres que inicialitzem per a utilitzar més endavant:

- jsonFile, a la que assignarem el valor en format JSON de l'arxiu d'estudis que carreguem

- nomUsuari i estudis, que tindran el valor que es configuri o que hi hagi desat al *localStorage*

I tota una sèrie de matrius que utilitzarem per tenir les dades organitzades durant els càlculs:

- assignaturesPerCodi[] contindrà un índex de les assignatures segons el seu codi per accedir-hi ràpidament allà on només disposem del codi.
- assignaturesPerTipus[] és una matriu que tindrà com a primers índexs els diferents itineraris, i per a cadascun, matrius de les assignatures segons el seu tipus -ja que segons l'itinerari, algunes optatives es converteixen en obligatòries.
- assignaturesSegonsItinerari[] per desar quines assignatures hi ha optatives d'aquell itinerari en concret.
- assignaturesColocades[] on tindrem una llista de codis d'assignatura que ja han estat assignats a algun semestre o acabades.
- assignaturesColocadesPerSemestre[] ens serveix per tenir un accés ràpid a les assignatures d'un semestre determinat.
- creditsFets[] és una matriu que conté els sumatoris de crèdits de les propostes -que es fan per a cada itinerari.
- creditsFetsPerItinerari[] és una matriu de suport, on per a cada proposta, controlem les assignatures de cada itinerari que fem, per assegurar-nos que els càlculs fets són correctes.
- assignaturesPerItinerari és l'equivalent al de crèdits però per número d'assignatures.
- assignaturesOperativesNecessaries[] conté, per itinerari, aquelles assignatures optatives que no són del propi itinerari però les de l'itinerari les tenen com a recomanades d'haver fet prèviament.
- assignaturesItera conté inicialment una còpia de les assignatures extretes del JSON, i es van esborrant aquelles que ja s'han col·locat per a tenir un iterador.
- proposta[] i propostaNova[] és on desem les propostes. Inicialment tot el procés el farem sobre proposta[] tot col·locant absolutament totes les assignatures, i a propostaNova[], que és la proposta final per a cada itinerari, s'hi trasllada la proposta definitiva sense crèdits sobrants. En aquests objectes, el primer nivell és l'any, el segon el semestre, i en cada semestre, les assignatures que s'hi hagin col·locat.

5.4 Interfície d'usuari bàsica

Aquí haurem de diferenciar dues parts principals:

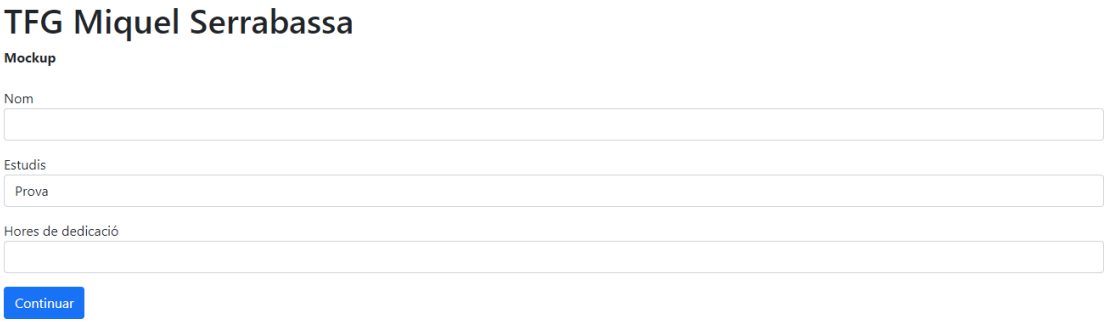
1. Fase on l'usuari omple les dades per inicialitzar l'eina
2. Presentació de la proposta de matrícules (històric+futur) amb possibilitat d'incloure-hi alguna dada com hores de dedicació setmanals per tal de poder fer canvis ràpids i fer comprovacions.

Amb aquesta premisa en ment, la meva primera aproximació va ser pensar primer en un formulari d'entrada de dades inicial, que després passés a la proposta.

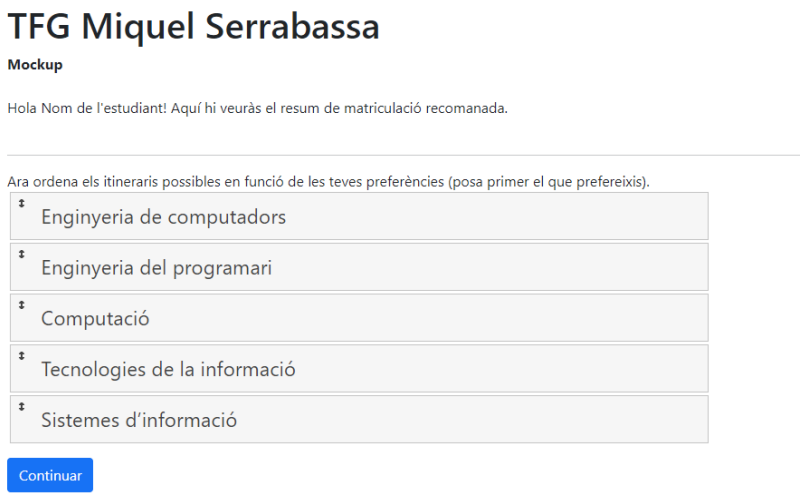
Així, el primer cop que hi accedíssim, se'ns hauria de presentar un formulari que ens demanés:

- El nostre nom, per tenir una referència de personalització
- Els estudis que cursarem -que, per ara, seran els del Grau d'Enginyeria Informàtica.
- Hores disponibles setmanals

Una primera idea per aquest formulari era la següent:

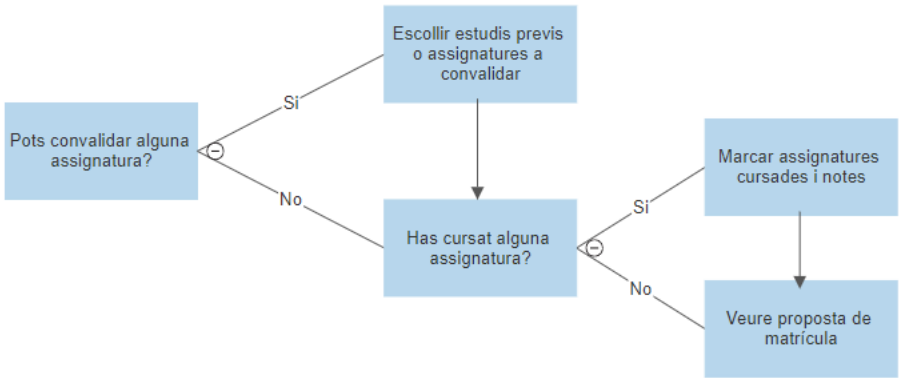


A l'escollir uns estudis, el sistema carregaria l'arxiu amb la informació d'aquells estudis, de manera que al continuar, mostraríem els itineraris per tal de configurar-ne l'ordre de prioritat.



Aquesta llista d'itineraris es mostra en una llista sense ordenar en HTML (<ul> i <li> per a cada element) per després vincular-ho a un *sortable* de JQuery UI, que vinculat a JQuery UI Touch Punch ens permet fer que l'ordenació sigui del tipus *drag and drop* per canviar l'ordre movent els elements amb el ratolí o amb el dit en el cas de dispositius mòbils.

Després, podríem establir un flux de preguntes simples com el que es veu a continuació:



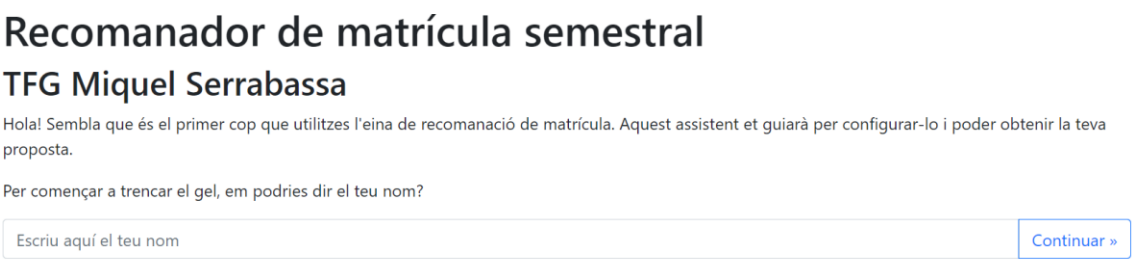
Per tal d'identificar si cal configurar convalidacions o assignatures ja cursades. En aquest cas, hi hauria dues pantalles:

- Selecció d'estudis previs. En aquest cas entenem que escolliríem uns estudis previs (en l'exemple que farem mirarem que sigui CFGS DAW/DAM) i ja en tindríem prou
- Marcar assignatures cursades i notes. En aquest cas és més complex a nivell d'interfície, ja que hem de veure com ho fem. Potser un desplegable amb les assignatures no cursades, i al triar-ne una sol·licitar algunes dades com la nota que es va treure i el semestre/any, i desar-ho i treure aquesta assignatura del desplegable. En tot cas, aprofundirem en aquesta opció quan ho desenvolupem.

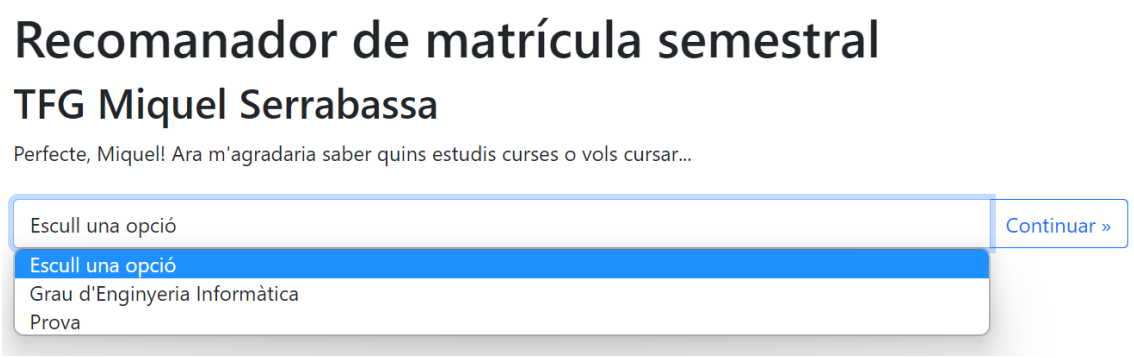
A continuació, ja s'haurien recollit les dades per començar a fer càlculs i presentar una proposta. També podríem modificar les hores de disponibilitat i si tenim assignatures ja convalidades o superades, marcar-les:

- Desplegable d'assignatures
- Desplegable de superada/convalidada/suspesa
- Si superada o suspesa, demanarem nota numèrica i calcularem la lletra.
- Podem demanar també semestre en què s'ha superat, però de forma opcional per no allargar-ho.
- * En aquest cas, podríem arribar a definir una matriu de convalidacions, tot i que hi ha molts estudis i probablement la parametrització per a provar-ho ja ens ocuparia més temps del que tenim per acabar.

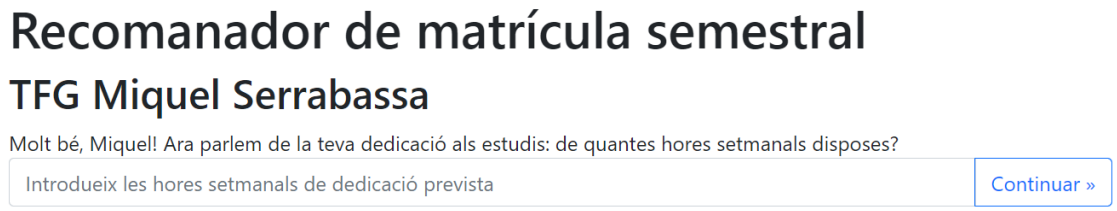
Tot i això, al començar-ho a desenvolupar integrant la recollida i modificació de dades, em vaig adonar que potser era més intuïtiu per l'usuari -i feia més fàcil la precàrrega de dades i la modificació de cada dada, si ho plantejava com un assistent (un dels típics *wizards* d'instal·lació), on cada pas era una pregunta concreta. Així, el resultat final seria:



Un primer pas ens demana el nom, i així als següents ja ens parla directament:



El segon pas ens demanaria els estudis -i aquí ja s'aprofitaria per precarregar la informació dels estudis escollits.



Després ja passaríem a introduir les hores per, al següent pas, establir la preferència dels itineraris, pas que es mantindria com a la idea inicial, juntament amb el flux de preguntes següent, amb algun canvi:

- Primer preguntem si es pot convalidar alguna assignatura.
    - En cas afirmatiu, l'usuari tria entre un llistat d'estudis previs i se'l condueix a la pantalla de configuració d'assignatures, saltant-se el pas intermig, i afegint les assignatures dels estudis previs ja com a convalidades.
    - En cas negatiu, es va al pas intermig on se li pregunta si vol configurar assignatures de forma manual (perquè ja les hagi cursat o perquè les vulgui forçar en algun semestre concret).

La pantalla on escollir uns estudis previs simplement seria igual que el pas d'escollir el grau que vol cursar, mentre que l'apartat on configurar assignatures mostraria:

- Un desplegable amb les assignatures (s'anirien eliminant del desplegable les incorporades a sota)

- A sota, les assignatures que s'afegissin. Per a cada assignatura hi hauria múltiples estats
    - Cada estat d'una assignatura informaria de si és convalidada, suspesa, aprovada o perquè es vol forçar a matricular, així com la nota (si és suspesa o aprovada) i el semestre (opcional si és convalidada, aprovada o suspesa).

El fet de treballar els passos un a un ens facilita reutilitzar la interfície per a reconfigurar granularment cada opció que s'ha escollit. Així, podríem mostrar un resum de les dades triades amb l'estructura següent i l'opció d'anar modificant una a una les tries:

De manera que clicant a cada llapis tornaríem al pas de configuració on l'hem introduit, i al prémer el botó d'acceptar tornaríem directament a aquest resum, ja actualitzat.

A continuació ja podríem realitzar els càlculs i presentar-ho. L'elaboració dels càlculs s'explica en el proper apartat, i a partir del resultat obtingut, mostrarem les propostes en dos models: una taula semestral amb informació de les assignatures, i un diagrama on cada assignatura es representarà amb un requadre enllaçat a les assignatures de les que depèn o que en depenen, i amb un color diferencial per a cada semestre.

La mostra en forma de taula es fa en format HTML, amb un objecte <table> que es limita a mostrar les dades en format tabular i organitzat per anys i semestres, informant de l'assignatura, el tipus, els crèdits i si s'ha proposat o era indicada per l'usuari, amb un resum final a mode de sumatoris de crèdits.

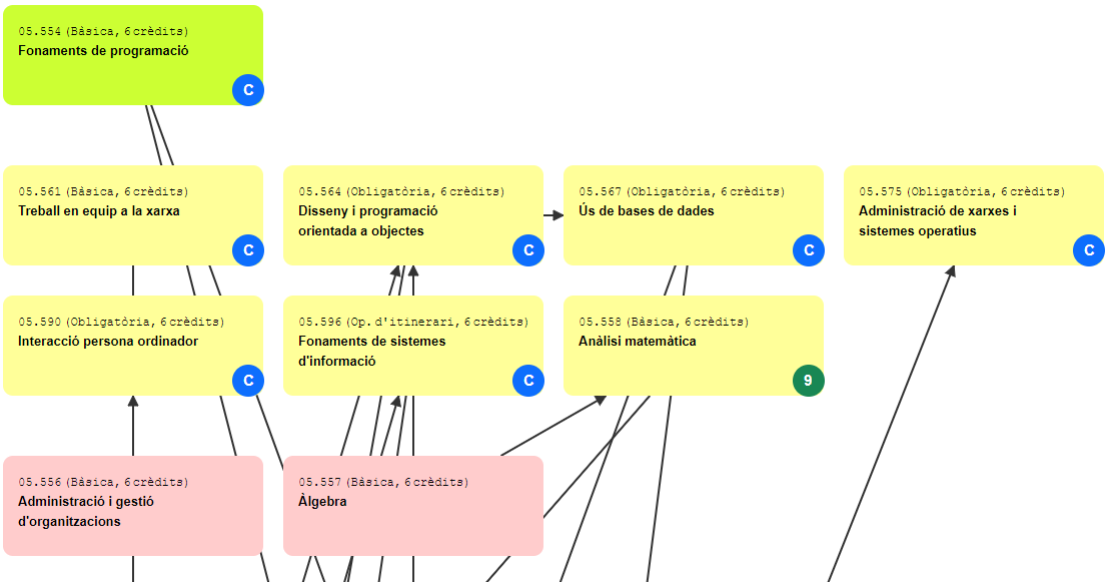
Per a la mostra gràfica en forma de diagrama hem optat per l'eina JointJS, explicada a l'apartat de *frameworks*, i que en el nostre cas es basa en un

exemple concret de la versió *openSource* de l'eina: l'arbre d'organització [52]

D'aquest gràfic, m'ha semblat especialment interessant la facilitat que tenia per mostrar la informació, tot definint uns "membres" i enllaçant-los. Tot i això, ens hem trobat amb limitacions:

- JointJS només admet un gràfic per pàgina. Per solucionar-ho i tenir-ne un per proposta, he fet que cada cop que es canvia de pestanya es regeneri el gràfic actiu, deixant els altres inactius, i així no tenim problemes.
- L'objecte org.Member que s'utilitza als exemples aportava facilitat perquè estava molt "tancat", de manera que he creat un objecte assignatura.Member a partir de l'exemple, eliminant la imatge i afegint una nova línia de text, i reubicant els continguts, així com modificant les alçades i amplades.
- Com que hi ha assignatures amb noms molt llargs, he trobat també una funció que retalla cadenes sense tallar paraules [53] i a partir d'això, he utilitzat el nou espai creat a l'objecte per incorporar una segona línia opcional de text per si el nom era molt llarg.
- Finalment, la funció "link" que duia l'exemple era una línia (objecte org.Arrow) que permetia posar-hi *breakpoints*, que en el meu cas no era tan interessant com que tingués sentit -apuntés d'una assignatura a una altra per saber quina depèn de quina-. Així que a la documentació he vist que l'objecte "Link" genèric, no el modificat de la demostració, tenia precisament aquesta opció i també era molt senzill d'implementar, de manera que ho he substituït per tal de mostrar la línia de forma correcta. Finalment, he hagut d'enviar la línia al darrere de les caixes amb una funció de JointJS anomenada toBack().
- En aquest darrer cas, però, he obviat alguns requisits que farien molt difícil la lectura (com ara relacionar el TFG amb 180 crèdits, que no està clar com s'hauria de fer, o totes les assignatures de l'itinerari, que generaria moltes línies i ho faria intel·ligible).

La mostra gràfica es veu d'una manera semblant a la de la següent imatge:



Així, disposem d'una matriu de colors diferencials que es van iterant per diferenciar els semestres. Per a cada assignatura veiem el codi, el tipus i número de crèdits, i el seu nom. I, a més, per les assignatures convalidades s'afegeix una rodona blava amb la lletra C i per les superades una rodona verda amb la nota, deixant de banda les suspeses en aquesta visualització. A més, les fletxes denoten l'ordre recomanat de realització d'assignatures.

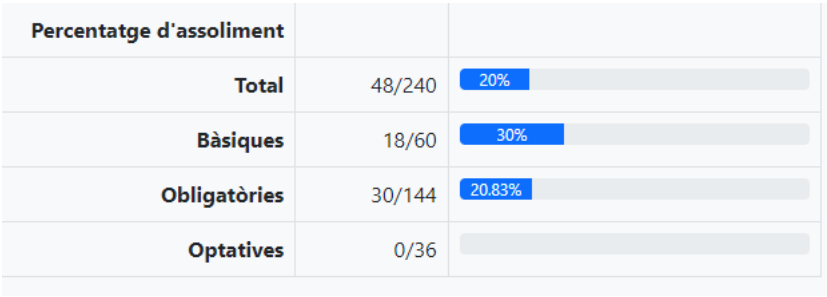
A nivell de dades tabulars, mostrem les dades de la següent manera:

Assignatura	Tipus	Crèdits	Observacions
Assignatures ja superades o convalidades			
05.554 Fonaments de programació	Bàsica	6	Convalidada
2020 Setembre-febrer			
05.561 Treball en equip a la xarxa	Bàsica	6	Convalidada
05.564 Disseny i programació orientada a objectes	Obligatòria	6	Convalidada
05.567 Ús de bases de dades	Obligatòria	6	Convalidada
05.575 Administració de xarxes i sistemes operatius	Obligatòria	6	Convalidada
05.590 Interacció persona ordinador	Obligatòria	6	Convalidada
05.596 Fonaments de sistemes d'informació	Obligatòria	6	Convalidada
05.558 Anàlisi matemàtica	Bàsica	6	Superada (9)
2021 Setembre-febrer			
05.556 Administració i gestió d'organitzacions	Bàsica	6	Proposada
05.557 Àlgebra	Bàsica	6	Proposada
2022 Febrer-juny			
05.568 Estadística	Bàsica	6	Proposada
05.611 Fonaments físics de la informàtica	Bàsica	6	Proposada

Dividint la mostra en semestres, i mostrant també codi d'assignatura i nom, tipus, número de crèdits i l'acció: convalidada, proposada -per l'eina-,

matriculada -forçada per l'usuari-, superada o suspesa, i en els dos darrers supòsits, si n'hi ha, es mostrarà la nota informada.

Finalment, per tancar la taula, es mostra un petit espai de percentatge d'assoliment, semblant al que es pot veure a l'Expedient del Campus Virtual de la UOC:



A nivell tècnic, per tal de reutilitzar elements de la interfície, hem utilitzat funcions en Javascript i classes que ens han ajudat:
- La pròpia funció d'inicialització -que llegeix l'arxiu config.json- omple quan es carrega la pàgina la llista d'estudis i de convalidacions possibles a partir de les dades de config.json.
- La funció *creaSelectAssignatures* aprofita l'estructura jsonFile.assignatures per crear el desplegable complet per configurar les assignatures prèvies.
- La funció *sortableItineraris* construeix la llista d'itineraris per ordenar amb *sortable de JQuery UI* a partir o bé de la llista original rebuda de l'arxiu sense ordenar, o bé a partir de la matriu d'ordre predefinit.
- La funció *llistaItinerarisOrdre* s'encarrega de mostrar els noms dels itineraris en l'ordre escollit per l'usuari.
- La funció *seguentpas* dibuixa els diferents passos de la configuració inicial. Malgrat tot, té una variable semàfor *endStep* que permet fer que només mostri un pas concret. D'aquesta manera, utilitzem una funció anomenada *editaConfig* amb un número de pas, per tal de cridar un pas específic de seguentPas i fer que només demani aquesta informació.
- Per a la configuració d'assignatures utilitzem una funció *afegirAssig* i *delAssig* per tal d'afegir línies d'assignatura o eliminar-les. A la vegada, tenim també la funció *afegirEstatAssig* per afegir estats a les assignatures (ja que pot haver-se suspès abans i superat o convalidat, o programat per un futur després, i així en tenim constància). *checkSiPosaNota* ens permet mostrar o amagar el camp de Nota per a cada estat en funció de l'acció -si és convalidat o matriculat no cal.
- Finalment, *calculaProposta* fa el primer càlcul de recomanació d'hores i si tot és correcte crida a *calculaProposta2*, que fa els càlculs per a la vegada cridar *showProposta* que s'encarregarà de dibuixar la taula, i de fer una darrera crida a *graphPropostes*, que genera la gràfica amb l'ajuda de JointJS.

5.5 Interfície d'usuari amb disseny millorat

Un cop finalitzades totes les tasques de desenvolupament i la majoria de les proves, em vaig centrar en intentar millorar la part estètica de l'eina, centrant-me en el disseny de la interfície. Val a dir que he d'agraïr els consells de disseny que en José Manuel Gutiérrez Caballero (Sobrevia.net, NacióDigital) em va donar per tal de fer-ho més atractiu.

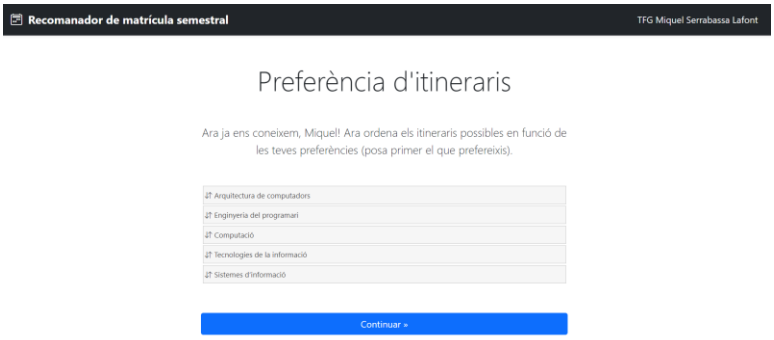
En primer lloc, les tasques es van centrar en fer una capçalera que fos més petita però que es veiés igualment. Per fer-ho, vaig optar per una franja de color fosca, per fer contrast, que ocupés tota l'amplada de la finestra.

A continuació, basant-nos en la mateixa disposició que tenien la resta d'apartats, vaig fer canvis en el marcat per tal d'aplicar-hi estils que donessin més espai entre els diferents elements de la interfície, centrant el text i afegint títols en gran per dirigir la lectura:

Així, ja en la primera pantalla, hi veiem una millora en l'aspecte. També he canviat els colors dels botons, per fer-los tots iguals si tenien una tasca similar -passar de pas, acceptar, ...-

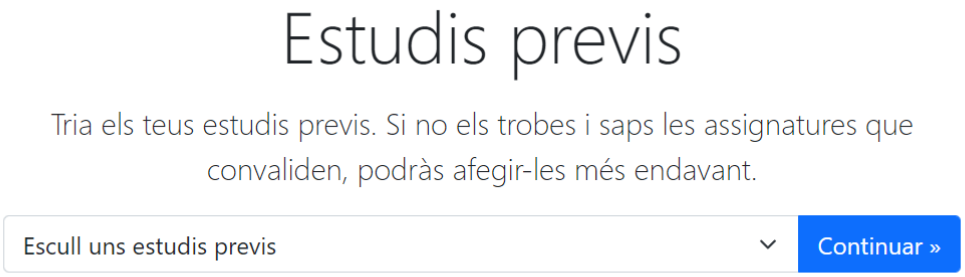
En els apartats posteriors, he seguit mantenint l'espai amb lletra més gran a mode de titular, i la pregunta central per triar o escriure.

Aquestes millores també fan més fàcil d'ordenar la llista de prioritats dels itineraris, a més de compensar molt més els elements a la pantalla:

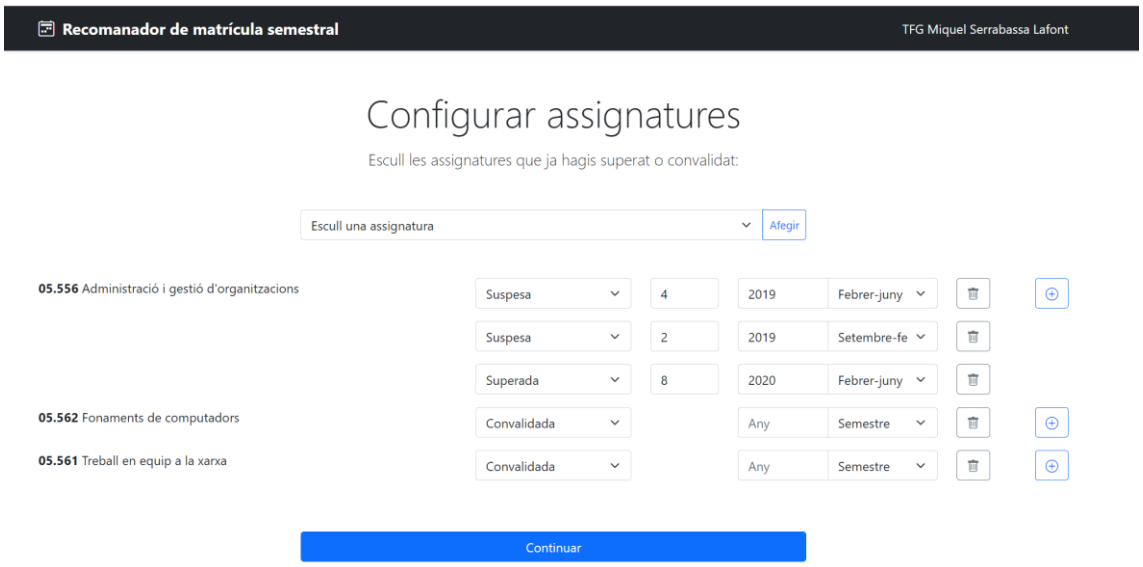


Els canvis ens han permès fer que les preguntes tinguin botons de resposta més ben diferenciats:

I també he detectat que els desplegables, amb el format que havia preparat inicialment, perdien la fletxeta que indicava que eren un desplegable, i l'hem recuperar corregint-ho arreu:

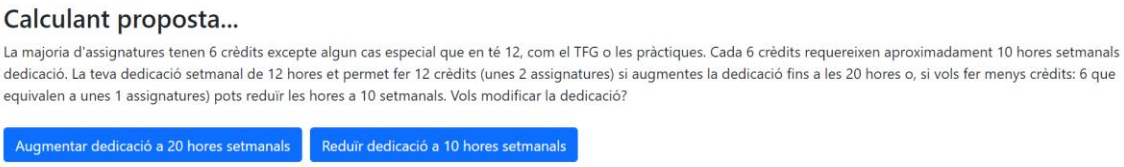


També hem reestructurat lleugerament la pantalla de selecció d'assignatures, fent-la molt més amable i simple d'utilitzar:

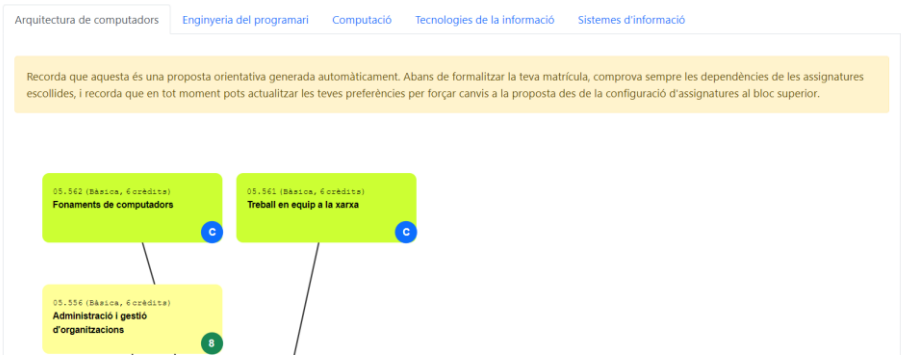


En aquest cas, a més dels marges i de moure alguns espais, hem afegit el botó per esborrar cada estat -quan ja no en queden s'elimina l'assignatura-, i hem recalculat els espais per a cada opció, de manera que quedin sense tallar els títols.

També hem deixat en fons blanc i botons blaus -com la resta d'opcions de configuració- l'avís que dóna el càlcul per tenir les hores justes per calcular:



I, finalment, en el càlcul pròpiament dit, hem englobat la gràfica de manera diferent, per fer que quedés enquadrada visualment com a dependència de l'opció d'itinerari escollida:



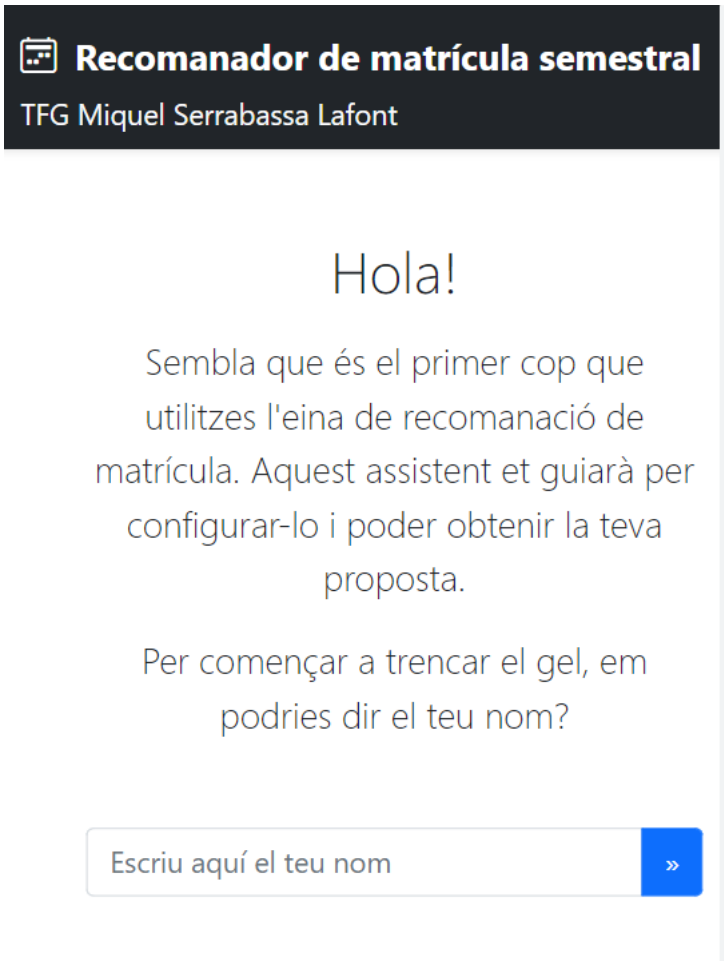
I a la taula hem eliminat línies verticals, canviat colors i afegit títols als resums finals:

Assignatura	Tipus	Crèdits	Observacions
Assignatures ja superades o convalidades			
05.562 Fonaments de computadors	Bàsica	6	Convalidada
05.561 Treball en equip a la xarxa	Bàsica	6	Convalidada
2020 Febrer-juny			
05.556 Administració i gestió d'organitzacions	Bàsica	6	Superada (8)
2021 Setembre-febrer			
05.557 Àlgebra	Bàsica	6	Proposada
05.568 Estadística	Bàsica	6	Proposada
2022 Febrer-juny			
05.558 Anàlisi matemàtica	Bàsica	6	Proposada
05.554 Fonaments de programació	Bàsica	6	Proposada

Crèdits proposats / necessaris		
	Total	240/240
	Bàsiques	60/60
	Obligatòries	144/144
	Optatives	36/36

Percentatge d'assoliment		
	Total	18/240
	Bàsiques	18/60
	Obligatòries	0/144
	Optatives	0/36

També hem revisat, de nou, la interfície en dispositius mòbils per fer-la més simple d'utilitzar -tot i que hi ha algunes opcions que són difícils de compatibilitzar. Així, en la configuració inicial els botons amaguen el text "Continuar" en dispositius de pantalla petita per donar més espai als elements d'entrada de dades:



El més complex és configurar les assignatures, perquè és un formulari molt extens i difícil d'encabir, però les accions es poden realitzar igualment tot i ser més complex de seguir els canvis realitzats.

5.6 Càlculs a realitzar

Ara que ja coneixem les dades, cal que tinguem clars els càlculs que necessitarem. Recordem que volem un recomanador de matrícules, que haurà de tenir en compte factors com:

- Quines assignatures ja hem superat o tenim prèviament convalidades
- Quantes hores volem dedicar-hi al semestre
- Quin itinerari hem escollit -si ja el tenim clar.
- Les assignatures que només es fan un semestre l'any

Durant la creació de l'estructura de dades, que he anat realitzant iterativament -primer coneixent les dades preexistents i després afegint-hi camps a omplir per a tenir la informació necessària- ja he pogut cobrir algun dels casos:

- Assignatures ja superades o convalidades, dins de l'estructura assignatures/estat on hi anirem desant aquestes dades per a la consulta posterior.
- Itinerari escollit que guardem a dadesUsuari i el de les assignatures que també queda registrat.

- La semestralització dins d'assignatures/semestres
- A dadesUsuari/semestres també hi tindrem la informació de la previsió de matrícula futura.

Per tant, ens quedaria parametritzar d'alguna manera el càlcul de càrrega semestral. En aquest cas, he buscat la informació primer a Google i he trobat la resposta al web de la UOC (en castellà) [54]:

"En el caso de titulaciones de primer y segundo ciclo, el crédito equivale a 10 horas de dedicación; en el caso de titulaciones de grado del EEES, el crédito ECTS equivale a 25 horas de trabajo."

Per tant, podem assumir una variable que anomenaríem "Dedicació per crèdit", i que establiríem en 25 hores. A partir d'aquí, ja podrem realitzar el càlcul a partir d'una mitjana de setmanes per quadrimestre (que miraré de calcular sobre calendari però no sempre disposarem de tots els festius, tot i que pel què he anat comprovant podríem assumir unes 15-16 setmanes) i de les hores informades per l'usuari. En aquest cas em queda estudiar la viabilitat també de desar aquesta informació al JSON o potser millor en cookie o de deixar-la respondre en diverses ocasions.

A nivell més tècnic, l'estructuració dels càlculs seria la següent:

1. Càlcul dels crèdits a realitzar durant cada semestre a partir de les hores informades per l'usuari. Tot i que partim d'una base de 25 hores per crèdit, aquesta xifra la deixarem fàcilment modificable a l'HTML per mitjà d'una variable Javascript. De moment, però, per il·lustrar els càlculs, emprarem el 25 com a valor fixat.
    a. Els crèdits sempre seran múltiples de 6, de manera que sempre tindrem un mínim de 25 hores*6 crèdits=150 hores, que repartirem en 16 setmanes d'un quadrimestre. Ens quedarien unes 9 hores i mitja a la setmana per assignatura de 6 crèdits (18 tirant cap a 20 per les de 12 crèdits).
    b. Sabent això, doncs, sabem que tindrem una assignatura per aproximadament cada 10 hores setmanals disponibles que indiqui l'usuari.
    c. En cas que l'usuari informi menys de 10 hores, se l'advertirà i es comptarà una sola assignatura per semestre, tant si és de 6 com de 12 crèdits -ja que del contrari no podria acabar.
    d. En cas que les hores informades per l'usuari no siguin múltiples de 10, se li proposarà incloure més o menys hores -i els crèdits que toquen- per tal de poder quadrar la proposta.

2. A partir de les estructures de dades definides anteriorment, engegarem de forma paral·lela el procés per separat per a cada itinerari.

3. Iterarem entre totes les assignatures per a cada itinerari i guardarem informació sobre:

a. Quines assignatures són optatives d'itinerari per a cada cas, per comptar-les com a obligatòries.
b. Quines assignatures optatives que no són de l'itinerari necessitem, si n'hi ha, cursar abans de les optatives d'itinerari.

4. La primera fase del càlcul de la proposta pròpiament dit consistirà en incorporar al calendari de la proposta totes les assignatures convalidades, superades o forçades manualment per l'usuari, així com incloure les referències d'aquestes assignatures en les llistes d'assignatures ja fetes en global i per itinerari i tipus, i retirar les assignatures ja col·locades de la variable on hi desem les que ens falten per incorporar.

5. La segona fase serà recursiva. Llegirà les assignatures pendents segons el tipus (primer bàsiques, després obligatòries, després les assignatures optatives necessàries per a les d'itinerari, després les optatives d'itinerari i finalment la resta d'optatives) i les anirà assignant a la proposta.

   a. Per a col·locar cada assignatura en una subfase també recursiva, primer es mirarà tot el què s'hagi incorporat a la proposta.
   b. Es buscarà quin és el semestre més alt lliure -i si és del passat, es marcarà el semestre vinent com al que s'ha de començar a col·locar. Entenem com a lliure un semestre en què quedin suficients crèdits com per cursar l'assignatura que estem comprovant -o, si l'assignatura supera els crèdits màxims setmanals, es forçarà quan ja no quedin mes opcions disponibles.
   c. Es compararà la informació de requisits de l'assignatura -per saber si es pot matricular o encara no i primer s'han de col·locar altres matèries.
   d. Es compararà la informació sobre les bisemestrals, en cas de poder-la col·locar, per veure si és possible cursar-la en el semestre proposat.
   e. S'anirà repetint el procés mentre quedin assignatures que es puguin col·locar.

6. El procés s'anirà repetint sense tenir en compte cap límit d'assignatures ni de crèdits totals, sinó fins que ja no quedi cap assignatura pendent de col·locar -superant els crèdits necessaris. Quan s'arribi a aquest punt, es retallarà la proposta:
   a. Treballarem amb una nova proposta, en aquest cas incorporant-hi les assignatures que haguem anat desant. Això es farà fins que detectem una assignatura optativa i el número de crèdits proposats superi els crèdits per superar el grau. D'aquesta manera, ens assegurarem que hem matriculat prou crèdits i que hem complert tots els requisits.
   b. Treballar amb una nova variable de proposta en aquest pas ens facilita el procés respecte a retallar la proposta existent, ja que canviarien els índexs i seria difícil fer correctament la iteració.

7. Les presentarem a l'usuari ordenades segons:
    a. La preferència d'itineraris indicada per l'usuari, amb l'itinerari preferit visible d'entrada.
    b. El format inicial podrà ser el d'una taula amb els detalls de la proposta i assignatures per a cada semestre.
    c. Treballarem també en la presentació gràfica de la proposta.

Per assolir l'objectiu dels càlculs, a la interfície el botó "Calcular proposta" llança primer una crida a una funció CalculaProposta que primer inicialitza l'objecte propostaNova que és el que utilitzarem per mostrar els resultats finals -i així evita duplicar recomanacions si es fa clic al botó més d'un cop-, crida a la creació de les estructures temporals de dades amb la funció creaEstructuraProposta i també s'encarrega d'avaluar les hores de disponibilitat indicades per tal de proposar a l'usuari que n'acabi escollint un número que permeti quadrar la matrícula.

Quan s'obté un múltiple de 10 (en aquest cas, de les hores que guardem com a disponibilitat setmanal mínima), llavors es crida a una nova funció, CalculaProposta2, que és la que inicia els càlculs de recomanació.

Aquest càlcul de proposta comença mirant quin és el semestre actual -en què ens trobem per data a dia d'avui- per tal de poder adaptar els càlculs, mitjançant la funció calculaSemestreActual que retorna una matriu amb el semestre i l'any. A continuació clona la matriu d'assignatures a un objecte temporal (assignaturesItera) per després començar a col·locar les assignatures que l'usuari ha configurat a l'estructura de la proposta.

Després de cara iteració o operació amb la proposta, hi ha crides a la funció actualitzaPropostaISumatoris, que actualitza estructures com els sumatoris de crèdits i assignatures globals, per tipus i per itinerari, en funció de les dades que acabem d'incloure a assignaturesColocades.

Fins aquest moment, s'ha estat treballant en una mateixa sola estructura de dades perquè teníem informació global, però a partir d'aquest moment es clonen les dades per a cada itinerari possible i es bifurca el càlcul per a cada itinerari. Ara, com que ja hem de recomanar -abans d'aquest punt col·locàvem les preferències de l'usuari sense fer comprovacions- utilitzem la funció colocaAssignatura, que primer verifica que l'assignatura no estigui ja posada en algun semestre, per després iterar fent crides a una altra funció, miraProperSemestreLliureProposta, per veure quan es pot ubicar l'assignatura que s'està comprovant. En cas de poder-la posar, inclourà la informació a assignaturesColocades i cridarà la funció actualitzaPropostaISumatoris.

La funció miraProperSemestreLliureProposta és un dels punts més importants perquè fa els càlculs concrets per a cada assignatura. És una funció recursiva, perquè si no troba una resposta es torna a cridar amb paràmetres nous (buscant un semestre més enllà o, si detecta que les

hores de l'assignatura són superiors a la disponibilitat de l'usuari, forçant la col·locació de l'assignatura per evitar un bucle infinit). Així, verificarem aquí si hi ha disponibilitat suficient per col·locar els crèdits, si el semestre actual es cursa l'assignatura en cas de ser bisemestral, i també mirarem si s'han fet les assignatures prèvies amb una altra funció addicional, miraSiPreviesFetes.

En aquest darrer cas, miraSiPreviesFetes mira el camp despresDe de l'assignatura: s'hi pot trobar codis d'assignatura -que simplement verificarà si s'han cursat en un semestre anterior a l'actual-, o alguns casos com un número enter -per verificar si s'han completat més de X crèdits-, el text "Itinerari" que verifica que s'hagin cursat els crèdits d'assignatures optatives d'itinerari, o altres opcions que deixem pendents de desenvolupar perquè no apliquen a la majoria dels casos i implicarien complicacions més grans a nivell temporal.

Tornant al bucle principal, un cop col·locades les assignatures pot ser que en quedin de pendents -perquè hi havia assignatures que s'havien de col·locar abans i estaven en una posició més avançada de la matriu- així que s'anirà iterant mentre no s'hagin col·locat totes les assignatures, i finalment es fa la darrera crida a la funció showProposta que s'encarrega de gestionar la part de mostra de les dades -més relacionada amb la interfície.

Un darrer càlcul, secundari però també a tenir en compte per a descriure'l, és el que realitzem per mostrar el percentatge d'assoliment. En aquest cas, durant la darrera iteració de dibuix sumem els crèdits superats o convalidats i els totals, i en calculem el percentatge amb un dividint crèdits superats per crèdits totals i multiplicant-los per 100. Amb aquest valor, després, mostrem una barra de progrés on arrodonim amb el mètode Math.round de Javascript, que arrodoneix sense decimals, però per a obtenir dos decimals es multiplica el valor que tenim per 100 i es torna a dividir per 100, de manera que l'arrodoniment s'assoleix a dos decimals per mostrar-ho correctament.

5.7 Ús en un ordinador local

Tot i que la intenció inicial era que el mateix paquet es pogués executar des d'un servidor web o des d'un ordinador local amb un simple doble clic a l'arxiu index.html, el cert és que les restriccions actuals de seguretat dels navegadors ho fan impossible.

En concret, les crides de qualsevol tipus a arxius que es trobin al disc dur estan prohibides -només es poden fer via http o https, o sigui, mitjançant un servidor web.

Per provar de solventar-ho, hem intentat executar require.js [55] (que té una extensió per llegir arxius JSON), així com l'API Fetch [56] del navegador, però en tots els casos actualment els navegadors ja no admeten l'ús d'arxius del disc dur per evitar problemes de seguretat que se'n derivarien.

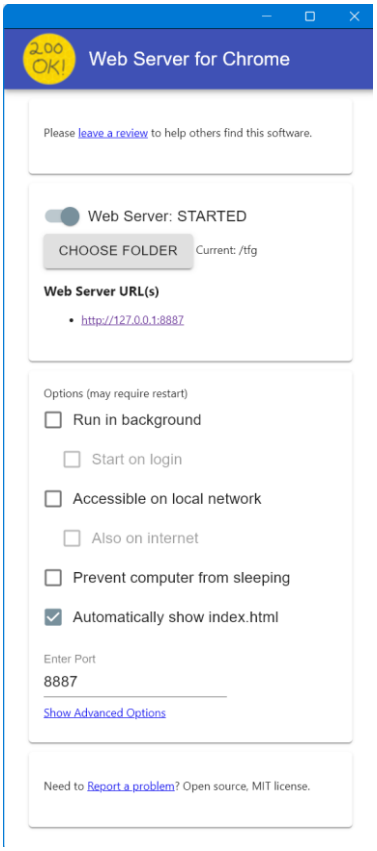
Això implica, doncs, que només tindríem dues opcions:

1. Passar l'arxiu JSON a una variable Javascript dins de l'arxiu index.html (o de l'scripts.js). Això faria que qui l'utilitzés necessités tenir coneixements d'HTML i Javascript.
2. Intentar executar el navegador amb alguna instrucció que eliminés les restriccions -però actualment ja no és possible, tot i que fa uns mesos encara funcionava algun truc.
3. Utilitzar un servidor web local.

La solució 3, tot i semblar complexa, semblava la més viable i finalment he trobat una opció que crec que és molt més efectiva, basada en això: l'ús d'una extensió de Chrome, anomenava "Web Server for Chrome" [27], i que activa un servidor web local -accessible a priori només des de la mateixa màquina- a partir d'una carpeta de l'ordinador.

Visitant la web de l'extensió des de Chrome la podrem instal·lar, i un cop instal·lada, la trobarem a l'espai d'aplicacions instal·lades dins de Chrome, a la que es pot accedir ràpidament escrivint chrome://apps/ a la barra d'adreces del navegador.

Aquí buscarem la icona de l'extensió:



I al fer-hi clic se'ns presentarà la configuració. Aquí només haurem d'escollir la carpeta:

Farem clic inicialment al botó Choose Folder, i escollirem la carpeta on ho hem desat. Un cop fet, ens assegurarem Web Server indica "Started" i ja podrem fer clic a l'opció "Web Server URL(s)", que obrirà l'adreça on podem veure ja els continguts correctament.

L'extensió dóna encara algunes altres opcions -com ara obrir la connexió a altres ordinadors de la xarxa local o funcionar en segon pla-, però per a l'execució no farien falta. També es podria canviar el port, tot i que el que duu per defecte és prou concret.

6. Proves

Un cop desenvolupada l'eina de recomanació de matrícula, ara hem de verificar que hem assolit els objectius mitjançant una sèrie de proves. Per fer-ho, hem establert un pla de proves que hem dividit en quatre tipologies:

- **Proves de compatibilitat** [2]: comprovació del funcionament de la interfície en diferents entorns -navegadors diferents, mides de pantalla i dispositius diferents.
- **Proves funcionals** [3]: on s'han fet proves de les funcionalitats per verificar que els requisits inicials estan coberts.
- **Proves destructives** [4]: on provem d'executar tot allò que pugui fer fallar l'eina. En el nostre cas, com que no hi ha un servidor per atacar (és tot *client-side*), obviarem atacs distribuïts o de denegació de servei perquè no serien responsabilitat de l'eina sinó del servidor on s'allotja.
- **Proves d'usabilitat** [5]: Valoració de l'eina en funció del seu grau d'usabilitat i propostes que es podrien fer per a millorar-la.

Per a les **proves de compatibilitat**, doncs, estudiem el comportament tècnic de l'eina en diferents entorns de programari per validar que funciona com s'espera. No entrarem a valorar funcionalitats concretes o requisits -aixo és per les proves funcionals- sinó que l'eina es carregui i s'hi pugui interactuar -o fins a quin nivell es pot amb un seguit de configuracions.

Per escollir les configuracions, tindrem en compte:

1. Dispositiu: farem proves amb ordinador, tauleta digital i telèfon mòbil.
2. Sistemes operatius: ho provarem en un Windows, un iOS i un Android.
3. Navegadors: Les proves es faran amb Chromium (Edge i Chrome), Firefox, IE11 (tot i que aquest no el prendrem en consideració a l'hora de fer modificacions), Safari i Chrome per Android.

Aquests diferents casos s'han escollit perquè, per una part, i com havíem vist a la descripció de les PWA, estem cobrint un percentatge molt alt de la quota de mercat de les diferents configuracions, i a més prenem exemples on ens trobarem amb diferents cobertures de compatibilitat, per poder documentar, en cada cas, fins a quin punt podem fer ús de totes les capacitats que hem programat o no.

I per tal de llistar correctament aquestes capacitats, les hem dividit en preguntes concretes:

- Es pot accedir a l'eina? (Serà No si no es pot utilitzar o dóna errors que no permeten fer-ne cap ús.)

- Es poden visualitzar tots els continguts? (Serà No en cas que alguna part dels continguts no es pugui veure en pantalla)

- Es pot instal·lar al dispositiu?

- Es poden ordenar correctament els itineraris?

- Observacions: explicació de les limitacions concretes trobades a l'eina per aquella configuració.

Fitxes de proves

Configuració: PC Windows 11 + Google Chrome 96	
Es pot accedir a l'eina?	Si
Es veuen tots els continguts?	Si
Es pot instal·lar al dispositiu?	Si
Es poden ordenar correctament els itineraris?	Si
Observacions: Totes les proves funcionen correctament, ja que s'ha desenvolupat inicialment amb aquest entorn en ment.	

Configuració: PC Windows 11 + Microsoft Edge 96	
Es pot accedir a l'eina?	Si
Es veuen tots els continguts?	Si
Es pot instal·lar al dispositiu?	Si
Es poden ordenar correctament els itineraris?	Si
Observacions: Totes les proves funcionen correctament, al tenir el mateix motor que Chrome (Chromium).	

Configuració: PC Windows 11 + Firefox 94	
Es pot accedir a l'eina?	Si
Es veuen tots els continguts?	Si
Es pot instal·lar al dispositiu?	No
Es poden ordenar correctament els itineraris?	Si
Observacions: El funcionament global és correcte, però ja sabíem prèviament que Firefox no permet instal·lar PWAs com aplicacions.	

Configuració: iPad iOS + Safari	
Es pot accedir a l'eina?	Si
Es veuen tots els continguts?	Si
Es pot instal·lar al dispositiu?	No
Es poden ordenar correctament els itineraris?	Fase 1: No. Fase 2: Si
Observacions: A la primera fase no funcionava l'ús de JQuery UI per a la ordenació. Gràcies a una recerca a Google, hem trobat a Github una ampliació de la funcionalitat per permetre que funcioni a iPad i Android [23]. Un cop aplicat, tot funciona correctament.	

Configuració: Mòbil Xiaomi Android + Chrome for Android	
Es pot accedir a l'eina?	Si
Es veuen tots els continguts?	Si
Es pot instal·lar al dispositiu?	No

Es poden ordenar correctament els itineraris?	Fase 1: No. Fase 2: Si
Observacions: La pantalla és petita per la gràfica, però és difícil escalar una mida així, i en tot cas ho deixarem per la part d'usabilitat, ja que tècnicament ha funcionat tot. A la primera fase no funcionava l'ús de JQuery UI per a la ordenació. Gràcies a una recerca a Google, hem trobat a Github una ampliació de la funcionalitat per permetre que funcioni a iPad i Android [23]. Un cop aplicat, tot funciona correctament.	

Per a les **proves funcionals** verificarem els requisits de funcionament que hem marcat en aquesta memòria per tal de validar que es compleixen:

- Accedir i indicar el meu nom i que es desi **– Es compleix**
- Escollir els meus estudis si estan entrats a l'eina **– Es compleix**
- Introduir les hores setmanals de disponibilitat per estudiar **– Es compleix**
- Ordenar els itineraris per ordre de preferència **– Es compleix**
- Indicar si he cursat alguns estudis previs que puguin convalidar assignatures, i quins. **– Es compleix**
    - I, si no els trobo, poder seguir sense indicar-ne cap. **– Es compleix**
- Indicar si ja he cursat o convalidat assignatures, i quines **– Es compleix**
- Poder forçar la matrícula d'alguna assignatura en un semestre determinat. **– Es compleix**
- Modificar el meu nom i que es desi **– Es compleix**
- Rebre recomanacions sobre les hores de disponibilitat per optimitzar els meus recursos, i poder-les modificar **– Es compleix**
- Poder afegir, modificar o eliminar assignatures ja fetes, convalidades o matrícules que vulgui forçar. **– Es compleix**

- Obtenir una recomanació de matrícula segons les dades que hagi indicat **– Es compleix**
    - En format gràfic **– Es compleix**
    - En format taula **– Es compleix**
- Poder instal·lar l'eina com si fos una aplicació en múltiples dispositius -Progressive Web App. **– Es compleix**
- Poder utilitzar l'eina en qualsevol lloc
    - En diferents dispositius **– Es compleix**
    - En diferents sistemes operatius **– Es compleix**
    - Amb diferents navegadors **– Es compleix**
    - Amb un disseny que s'adapti als diferents tamanys de pantalla **– Es compleix**

Així doncs, mitjançant la realització d'una prova complerta de funcionament, hem anat verificant cadascun dels requisits tot seguint els passos mentre feiem una prova. Amb aquesta verificació, podem passar a la següent fase de proves, les destructives.

Per a les **proves destructives** en aquest cas hem realitzat alguns intents per forçar errors en els resultats. En aquest sentit, hem de tenir en compte algunes limitacions respecte a les proves que podríem fer:

- Descartem les proves que siguin relatives a la seguretat del servidor, perquè estan fora de l'abast del projecte
- Com que és una eina en Javascript i depèn exclusivament del client, no farem proves de seguretat sinó per verificar que es tracta correctament l'entrada de dades on correspongui perquè l'eina no falli.

Proves realitzades:

Acció	No introduir nom i continuar
Resultat	Rebem un error indicant que cal que indiquem un nom.

Acció	Introduir codi HTML al nom i continuar
Resultat	S'executava el codi HTML (*<script>alert('prova')</script>*)
Solució	Substituïm tots els caràcters < i > per les seves versions segures en HTML (< i >) abans de desar el nom
Resultat final	Deixa d'executar-se el codi HTML i es mostra el text tal i com s'ha escrit

Acció	No escollim estudis
Resultat	Rebem un error indicant que cal que n'escollim uns de vàlids

Acció	Escollim uns estudis pels quals no hi ha arxiu desat
Resultat	Rebem un error indicant que cal que n'escollim uns de vàlids

Acció	Indiquem un text enlloc d'un número on ens demana hores setmanals
Resultat	Al ser un objecte d'entrada de tipus "number" no deixa escriure lletres, només números

Acció	No indiquem disponibilitat en hores (posem 0 o ho deixem buit)
Resultat	Rebem un error indicant que cal escriure un número major que 0.

Acció	Indiquem que tenim estudis previs però no en triem cap.
Resultat	Segueix el procés sense incloure'n (podríem haver accedit a aquest pas per error, o no trobar els estudis prèviament entrats)

Acció	Introduïm una assignatura a l'espai de prèviament fetes, convalidades o per forçar, però no indiquem estat
Resultat	La col·loca fora dels semestres, però amb estat *undefined*
Solució	Si no s'ha indicat l'estat d'alguna de les assignatures, no deixa continuar advertint que queda alguna assignatura per acabar de configura
Resultat final	Ja no es poden forçar assignatures sense indicar-ne el motiu.

Arribats a aquest pas, ja no hi ha més opcions d'entrar text. Sí que, al ser codi obert (i Javascript visible per l'usuari), es podria modificar el codi i fer-li fer qualsevol altra cosa. Més enllà que aquest sigui precisament l'objectiu d'aquest projecte -fer-lo personalitzable al màxim-, també pot tenir implicacions de seguretat, però que surten també de l'abast d'aquest projecte perquè no seria possible analitzar tot el què es podria fer -i dependria ja no de criteris tècnics sinó de la confiança de l'usuari que ho executés.

I que inicialment volia fer una sèrie de **proves d'usabilitat** orientades principalment a usuaris que puguin necessitar programari específic tant per motius tècnics com per a discapacitats o altres col·lectius, al documentar-me sobre aquestes proves he trobat molta informació relativa a la necessitat de fer les proves amb equips concrets i amb participació de terceres persones que realitzéssin tots els processos de l'eina per analitzar-ne el seu comportament i rebre'n retroalimentació sobre la que treballar.

Tot això implicaria una quantitat de temps i l'ús d'unes eines de les que no disposo, de manera que no realitzaré aquestes proves però si que en documentaré quin seria un camí a seguir en aquest sentit:

- En primer lloc, com que ja hem fet proves de compatibilitat, sabem que el comportament de l'eina és correcte en diferents configuracions.
- Faltaria realitzar les proves amb programari com lectors de pantalla o similars
- Caldria fer proves amb els colors utilitzats per tal de verificar-ne el contrast en pantalla per a persones amb problemes visuals.
- Un cop verificats aquests passos, es podria procedir a l'estudi amb usuaris.
    - Un primer pas seria escollir una sèrie de *personas*, o perfils d'usuari que puguin estar interessats en l'eina. En aquest cas, a falta de definir-los més, podrien ser altres estudiants de la UOC -buscant diferents estudis, alguns més tècnics que d'altres, i de diferents edats.
    - Per a cadascun d'ells, hauríem de facilitar-los un accés a l'eina i acompanyar-los en el seu ús -sense explicar-los com funciona- i, a poder ser, enregistrar la sessió.
    - Un cop finalitzat, es podria realitzar una sessió de debat amb tots ells perquè cadascú valorés la seva experiència i poguéssin també proposar canvis.
    - Amb tot això, es podria elaborar un informe per a noves modificacions basades en el comportament dels usuaris i els seus comentaris.

7. Conclusions

Voldria dividir les conclusions en diversos subapartats perquè crec que és important detallar per a diferents àmbits els aprenentatges que n'he extret, i que són molts tant a nivell personal com tècnic, així com la revisió crítica de l'assoliment d'objectius i el seguiment de la planificació, per acabar amb les línies de futur que crec que podria ser convenient seguir.

7.1 Lliçons apreses

El què més m'ha costat -de fet, m'està costant mentre ho escric- és la documentació. Sovint ens centrem només en aspectes tècnics, però també és important tenir en compte que cal saber explicar què s'ha elaborat, amb quin objectiu, i quin és el seu funcionament, amb un llenguatge prou planer perquè sigui comprensible per tot tipus de perfils.

També és complex presentar gràfiques que puguin ser útils i estructurades, però resulta interessant realitzar-les per l'exercici de resum i definició que suposa; com també ho és preparar un guió per a la presentació de la memòria.

En un àmbit més pràctic, també és cert que cal parar molta més atenció del què sembla en la planificació temporal -i, per tant, també en estructurar els objectius en tasques que puguin comptar-se d'alguna manera. Durant el treball, amb la coincidència de dies festius, una malaltia -gens greu, però que ha provocat una certa descompensació del càlcul inicial- he vist la importància de fer els càlculs tenint en compte els imprevistos que puguin passar, però tenint en compte que tampoc podem ser massa conservadors donant estimacions temporals.

Finalment, voldria també posar èmfasi en els temes més tècnics, però que també m'han suposat un mal de cap en diverses ocasions. Per una banda, un dels meus objectius: la compatibilitat entre diversos dispositius i plataformes, així com el fet que el programari s'actualitzi, de vegades amb restriccions noves que no havíem tingut en compte inicialment. Així doncs, la meva intenció era poder disposar d'un entregable que funcionés en local només executant un arxiu html, a més de funcionar en un servidor web, però el fet que la parametrització faci servir arxius externs al propi HTML, i les noves directives de seguretat dels navegadors, han fet que això no fos possible. Per sort, he pogut trobar alternatives, com una extensió per a Google Chrome que permet aixecar un servidor local en minuts i sense complicacions, precisament per subsanar aquest problema.

7.2 Assoliment dels objectius

Considero que l'acompliment dels objectius marcats inicialment, a nivell global, ha estat satisfactori. He aconseguit la part que més em feia dubtar -que era disposar d'un entregable realment funcional, que pogués fer una proposta de matrícula vàlida i seguint totes les normes indicades.

Sí que és cert que durant el procés he deixat de banda alguns detalls concrets, per tal de centrar-me també en la interfície, i perquè a l'analitzar-los en detall, suposaven una quantitat de feina que no era assumible dins dels límits temporals del TFG.

Un exemple d'això seria els requisits d'algunes assignatures que anaven més enllà d'haver-ne fet una altra, i que per exemple, depenien d'un cert número de crèdits o d'assignatures només en el cas d'algun itinerari en particular. Aquests casos han quedat pendents i, tot i que per norma apareixen correctament recomanades, també s'ha afegit un text d'avís per recordar que cal revisar-ho sempre manualment, encara.

Finalment, també he tingut algunes limitacions tècniques, com comentava fa uns paràgrafs: el fet de voler facilitar la parametrització en arxius individuals fa que el navegador hagi de fer crides a arxius externs. I això els navegadors només ho permeten si l'arxiu es visita en un navegador web, no si s'obre directament des d'una carpeta del disc dur. Això és una restricció de seguretat, i per tant no es pot saltar fàcilment, de manera que he renunciat a que el "doble clic" a l'arxiu index.html sigui una manera vàlida de funcionar, però si que he documentat opcions simples per a poder-ho fer funcionar en local sense massa problemes.

7.3 Revisió de la planificació

A nivell de planificació temporal, ja des de l'inici he tingut la sort -o l'encert- de poder anar seguint les tasques segons la temporalització prevista inicialment.

Tot i això, és cert que en alguns casos he arribat a la data límit -o l'he passat, recuperant-ho en algun altre moment-, sobretot degut a la quantitat de dies festius intersetmanals, ponts i en algun cas, malalties que m'han fet reduir el ritme de treball.

En tot cas, l'ús de metodologia àgil amb targetes m'ha fet molt fàcil la feina, que ja havia començat a repartir a base de tasques a la PAC2, i m'ha permès poder-me "connectar" ràpidament a la feina tot i que hi hagués dies de desconnexió pel mig, així com a tenir la sensació que avançava, al poder anar marcant coses que ja estaven fetes -i que eren, a nivell temporal, assumibles totes elles en unes hores seguides. Finalment, la iteració de la metodologia àgil m'ha anat molt bé en el meu cas, ja que com que volia fer-ho per aprendre i trobar-me reptes, "tornar al principi" m'ajudava a reubicar-me quan necessitava fer algun canvi de previsió.

7.4 Línies de treball futur

Tot i que aquesta eina, tal i com està, compliria amb el meu objectiu inicial de ser una guia personalitzable de matriculació, és evident que té molt marge de millora a partir d'incloure altres funcionalitats. Així, les més evidents són aquelles que ja he deixat preparades, i per les quals quedaria només fer part de la feina:

- El treball que descrivia a l'apartat Estat de l'art, sobre un recomanador d'assignatures basat en l'opinió d'altres usuaris [12] també podria ser interessant, sobretot per a contactar amb els seus creadors i estudiar una possible integració entre els dos desenvolupaments, de cara a poder oferir una recomanació tècnicament correcta, però que també pogués ponderar-se a partir de les valoracions obtingudes a partir de les opinions d'altres alumnes.

- Actualment el TFG es matricula tant aviat com és possible, de manera que l'eina el sol incloure abans de matricular les possibles optatives que ja queden fora d'itinerari. Seria interessant estudiar una fórmula per poder col·locar sempre el TFG al final -com a opció configurable, o potser com a botó que obrís el configurador d'assignatures i inclogués automàticament el TFG al semestre més alt que es detectés a la proposta, per exemple.

- Una altra funcionalitat interessant, però paral·lela, seria preparar un assistent de creació d'arxius JSON, que oferís un formulari per a crear els arxius parametritzats que he descrit en aquest treball. La seva creació fins ara ha estat manual -i no és excessivament complex-, però també seria viable disposar d'una eina per facilitar-ho encara més a perfils menys tècnics. També es podria estudiar substituir l'opció de triar els estudis d'un desplegable per penjar un arxiu de paràmetres, per exemple.

- Una altra funcionalitat que podria ser interessant seria la d'exportar la recomanació -o la configuració completa- i poder-la també importar, per moure-la entre dispositius o compartir-la, i que pogués ser fàcilment modificable per una tercera persona pel seu propi cas basat en el nostre.

- Finalment, com a opció ja més complexa, però també viable, podríem ampliar l'àmbit dels arxius de parametrització per incloure-hi, d'alguna manera, els preus dels crèdits -i les diferents opcions que hi ha, com ara repeticions, només drets d'exàmens o descomptes per beques-, de manera que poguéssim calcular i veure també el preu de cada semestre -com a mínim a la data en què ho generem, perquè en un futur els preus podrien anar variant.

8. Glossari

API Espai automatitzat de consulta o intercanvi de dades entre dues màquines

Client-side / Server-side: Diferenciació entre tecnologies que s'executen a l'ordinador local o les que necessiten un ordinador extern -servidor- amb un programari específic per a processar-se.

CSS: De "Cascading Style Sheets", definicions d'estil que s'apliquen a documents HTML per a dotar-los de disseny.

DOM: De "Document Object Model", es refereix als diferents elements d'un document estructurat (com HTML)

Framework: Entorn de treball, a nivell tecnològic es refereix a llibreries que faciliten una sèrie de tasques.

FTP: De "File Transfer Protocol", el protocol de transferència de fitxers per excel·lència quan es parla de servidors web, per tal de publicar arxius que seran mostrats pel servidor.

HTML/HTML5: De "HyperText Markup Language", llenguatge de marques que utilitzen les pàgines web. La versió 5 és la més recent.

JS/Javascript: Llenguatge de programació *client-side*, molt extès i el que he utilitzat principalment en el meu treball.

JSON: Format de dades pensat per a Javascript, basat en objectes.

LAMP: Acrònim d'una combinació específica de programari instal·lat en un servidor: Linux com a sistema operatiu, Apache com a servidor web, MySQL com a servidor de bases de dades i PHP com a intèrpret de codi.

localStorage: Funcionalitat dels navegadors moderns per desar dades en local seguint unes restriccions de seguretat.

MySQL: Servidor de bases de dades relacionals molt extès per a l'ús de pàgines web

Navegador: Programa que s'utilitza per visualitzar documents HTML de pàgines web.

PHP: Llenguatge *server-side* molt extès, que s'utilitza, per exemple, a Wordpress o Laravel.

PNG: Format d'imatges informàtiques.
PWA: De "Progressive Web App", un tipus de pàgina web amb funcionalitats addicionals que s'ofereixen només als clients compatibles.

Service Worker: Arxiu que executen les PWA per tal de gestionar les funcionalitats addicionals (memòria cau, notificacions push, instal·lacions..)

Scrapping: Tècnica de lectura de pàgines web per a extreure'n dades de forma automatitzada.

SSL: Protocol de seguretat, quan en faig referència al treball és per definir certificats SSL que permeten als webs intercanviar informació de forma segura amb l'usuari.

SVG: Format d'imatges informàtiques basat en llenguatge de marques.

UML: De "Unified Modeling Language", llenguatge gràfic per documentar un sistema a partir d'un estàndard.

URL: De "Uniform resource locator", defineix l'adreça amb la que es pot accedir a una pàgina determinada d'un web.

VPN: De "Virtual Private Network", xarxa privada virtual per augmentar la seguretat o donar accés a determinats recursos.

9. Bibliografia

[1] Reconeixement 3.0 Espanya de Creative Commons. (11 / 11 / 2021). Recollit de https://creativecommons.org/licenses/by/3.0/es/

[2] Documentació de Google sobre la compatibilitat entre navegadors. (11 / 11 / 2021). Recollit de https://developers.google.com/search/docs/advanced/guidelines/browser-compatibility

[3] Wikipedia: Functional Testing. (11 / 11 / 2021). Recollit de https://en.wikipedia.org/wiki/Functional_testing

[4] Wikipedia: Destructive Testing. (11 / 11 / 2021). Recollit de https://en.wikipedia.org/wiki/Destructive_testing

[5] Usability Testing. (24 / 11 / 2021). Recollit de Usability.gov, U.S. General Services Administration: https://www.usability.gov/how-to-and-tools/methods/usability-testing.html

[6] Wikipedia: Kanban (Desarrollo). (13 / 12 / 2021). Recollit de https://es.wikipedia.org/wiki/Kanban_(desarrollo)

[7] Wikipedia: Metodologia Àgil. (13 / 12 / 2021). Recollit de https://ca.wikipedia.org/wiki/Metodologia_%C3%A0gil

[8] Aplicación móvil para la recomendación de asignaturas en los estudios de grado universitarios. A. Zafra, E. G. (14 / 12 / 2021). Recollit de https://www.uco.es/ucopress/ojs/index.php/ripadoc/article/view/10976/10118

[9] Wikipedia: Multiplataforma. (14 / 12 / 2021). Recollit de https://es.wikipedia.org/wiki/Multiplataforma

[10] Wikipedia: Disseny web responsiu. (14 / 12 / 2021). Recollit de https://ca.wikipedia.org/wiki/Disseny_web_responsiu

[11] Wikipedia: Escalabilitat. (14 / 12 / 2021). Recollit de https://ca.wikipedia.org/wiki/Escalabilitat

[12] What are Progressive Web Apps? Sam Richard, P. L. (14 / 12 / 2021). Recollit de https://web.dev/what-are-pwas/

[13] Wikipedia: Hyper Text Markup Language (català). (14 / 12 / 2021). Recollit de https://ca.wikipedia.org/wiki/Hyper_Text_Markup_Language

[14] Wikipedia: Cascading Style Sheets (en català). (14 / 12 / 2021). Recollit de https://ca.wikipedia.org/wiki/Cascading_Style_Sheets

[15] Wikipedia: JavaScript (en català). (14 / 12 / 2021). Recollit de https://ca.wikipedia.org/wiki/JavaScript

[16] Wikipedia: JSON (en català). (14 / 12 / 2021). Recollit de https://ca.wikipedia.org/wiki/JSON

[17] Cerca a Google: JSON Validators. (14 / 12 / 2021). Recollit de https://www.google.com/search?q=json+validators

[18] Mozilla Developer Network (MDN): Window.localStorage - Referencia de la API Web. (14 / 12 / 2021). Recollit de https://developer.mozilla.org/es/docs/Web/API/Window/localStorage

[19] Mozilla Developer Network (MDN): StorageManager.estimate() - Referencia de la API Web. (14 / 12 / 2021). Recollit de https://developer.mozilla.org/es/docs/Web/API/StorageManager/estimate

[20] Bootstrap · The most popular HTML, CSS, and JS library in the world. (14 / 12 / 2021). Recollit de https://getbootstrap.com

[21] JQuery. (14 / 12 / 2021). Recollit de https://jquery.com/

[22] JQuery UI. (14 / 12 / 2021). Recollit de https://jqueryui.com/

[23] jQuery UI Touch Punch. (14 / 12 / 2021). Recollit de https://github.com/furf/jquery-ui-touch-punch

[24] Lodash. (14 / 12 / 2021). Recollit de https://lodash.com/

[25] Backbone.js. (14 / 12 / 2021). Recollit de https://backbonejs.org/

[26] JointJS OpenSource. (14 / 12 / 2021). Recollit de https://www.jointjs.com/opensource

[27] Extensió de Google Chrome: Web Server for Chrome. (14 / 12 / 2021). Recollit de https://chrome.google.com/webstore/detail/web-server-for-chrome/ofhbbkphhbklhfoeikjpcbhemlocgigb/

[28] Arsys. (14 / 12 / 2021). Recollit de https://www.arsys.es

[29] Sobrevia.net Taller de projectes digitals. (14 / 12 / 2021). Recollit de https://www.sobrevia.net

[30] Cloudflare. (14 / 12 / 2021). Recollit de https://www.cloudflare.com/

[31] Visual Studio Code. (14 / 12 / 2021). Recollit de https://code.visualstudio.com/

[32] FileZilla project. (14 / 12 / 2021). Recollit de https://filezilla-project.org/

[33] Let's Encrypt. (14 / 12 / 2021). Recollit de https://letsencrypt.org/

[34] Google Chrome. (14 / 12 / 2021). Recollit de https://www.google.com/intl/ca_ES/chrome/

[35] Mozilla Firefox. (14 / 12 / 2021). Recollit de https://www.mozilla.org/ca/firefox/

[36] Microsoft Edge. (14 / 12 / 2021). Recollit de https://www.microsoft.com/es-es/edge

[37] Adobe Photoshop. (14 / 12 / 2021). Recollit de https://www.adobe.com/es/products/photoshop.html

[38] Trello. (14 / 12 / 2021). Recollit de https://trello.com/home

[39] Atlassian. (14 / 12 / 2021). Recollit de https://www.atlassian.com/

[40] PlantUML Online Server. (14 / 12 / 2021). Recollit de https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa70000

[41] LucidCharts. (14 / 12 / 2021). Recollit de https://lucid.app/

[42] Wikipedia: Transport Layer Security (en català). (17 / 12 / 2021). Recollit de https://ca.wikipedia.org/wiki/Transport_Layer_Security

[43] Wikipedia: Progressive enhancement. (17 / 12 / 2021). Recollit de https://en.wikipedia.org/wiki/Progressive_enhancement

[44] Wikipedia: Fault tolerance (Graceful degradation). (17 / 12 / 2021). Recollit de https://en.wikipedia.org/wiki/Graceful_degradation

[45] Progressive Web Apps: Escaping Tabs Without Losing Our Soul. Russell, A. (17 / 12 / 2021). Recollit de https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/

[46] Xataka: El juicio de Apple contra Epic Games ya tiene sentencia: un "empate" con indemnización millonaria y una histórica obligación de permitir otros sistemas de pagos. @Lyzanor, E. P. (17 / 12 / 2021). Recollit de https://www.xataka.com/legislacion-y-derechos/juicio-apple-epic-games-tiene-sentencia-empate-indemnizacion-millonaria-obligacion-permitir-otros-sistemas-pagos

[47] Xataka Android: El malware Joker no se va ni a tiros: encuentran más de diez apps infectadas en Google Play. @ivan_r, I. R. (17 / 12 / 2021). Recollit de https://www.xatakandroid.com/seguridad/malware-joker-no-se-va-a-tiros-encuentran-diez-apps-infectadas-google-play

[48] Ars Technica: Three iOS 0-days revealed by researcher frustrated with Apple's bug bounty. Salter, J. (17 / 12 / 2021). Recollit de https://arstechnica.com/information-technology/2021/09/three-ios-0-days-revealed-by-researcher-frustrated-with-apples-bug-bounty/

[49] Net Market Share: Browser Market Share. (17 / 12 / 2021). Recollit de https://netmarketshare.com/browser-market-share.aspx?options=%7B%22filter%22%3A%7B%7D%2C%22dateLabel%22%3A%22Trend%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22browser%22%2C%22sort%22%3A%7B%22share%22%3A-1%7D%2C%22id%22%3A%22browsersDesktop%22%2

[50] Web Fundamentals a Google Developers: Introducción a los service workers. Gaunt, M. (17 / 12 / 2021). Recollit de https://developers.google.com/web/fundamentals/primers/service-workers?hl=es

[51] Basic Service Worker Sample. (17 / 12 / 2021). Recollit de https://googlechrome.github.io/samples/service-worker/basic/

[52] JointJS Demos: Organizational Charts. (17 / 12 / 2021). Recollit de https://resources.jointjs.com/demos/org

[53] Stack Overflow: JS: Splitting a long string into strings with char limit while avoiding splitting words. (17 / 12 / 2021). Recollit de https://stackoverflow.com/questions/7624713/js-splitting-a-long-string-into-strings-with-char-limit-while-avoiding-splittin

[54] UOC: Espacio de acogida para nuevos estudiantes. (17 / 12 / 2021). Recollit de http://cv.uoc.edu/webapps/campus/estudiant/estudiant/acollida/es/grau/index.html

[55] Require.js. (17 / 12 / 2021). Recollit de https://requirejs.org/

[56] Mozilla Developer Network: Fetch API. (17 / 12 / 2021). Recollit de https://developer.mozilla.org/es/docs/Web/API/Fetch_API

[57] Las 10 principales autoridades de certificación: los mejores sitios para comprar certificados SSL (17/12/2021). Recollit de https://www.webhostingsecretrevealed.net/es/blog/ecommerce/buy-ssl-certificate/

[58] PWA — How to create a PWA React app using typescript (17/12/2021). Recollit de https://krishankantsinghal.medium.com/pwa-how-to-create-a-pwa-react-app-using-typescript-95564ab6390c