
This is the **published version** of the bachelor thesis:

Escobet Busquets, Aina. *Desenvolupament d'una plataforma web serverless per a l'anàlisi genòmica basada en Python amb WebAssembly i Pyodide*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334373>

under the terms of the  license.

Desenvolupament d'una plataforma web serverless per a l'anàlisi genòmica basada en Python amb WebAssembly i Pyodide

Aina Escobet Busquets

30 de juny de 2026

Resum— Les eines bioinformàtiques actuals depenen majoritàriament de la línia de comandes, fet que dificulta l'accés a l'anàlisi genòmica a investigadors sense formació tècnica i limita l'ús de programari especialitzat. Aquest treball presenta el disseny i la implementació d'una plataforma web *serverless* per a l'anàlisi genòmica de llocs d'unió de factors de transcripció (TFBS), executada íntegrament al navegador mitjançant Pyodide i WebAssembly. L'aplicació integra un motor bioinformàtic en Python amb una interfície web desenvolupada amb Vue.js, eliminant la necessitat de backend i garantint que les dades genòmiques romanguin al dispositiu de l'usuari. L'ús d'un Web Worker assegura una experiència fluida durant l'execució. La plataforma ha estat validada amb genomes bacterians reals, demostrant la seva viabilitat i el correcte funcionament del pipeline d'anàlisi en un entorn completament client-side.

Paraules clau— WebAssembly, Pyodide, bioinformàtica, TFBS, factors de transcripció, PSSM, Vue.js, serverless, Web Worker, anàlisi genòmica.

Abstract— Most bioinformatics tools still rely on command-line interfaces, which limits access to genomic analysis for researchers without technical training and reduces the usability of specialized software. This project presents the design and implementation of a serverless web platform for the analysis of transcription factor binding sites (TFBS), executed entirely in the browser using Pyodide and WebAssembly. The application integrates a Python-based bioinformatics engine with a Vue.js web interface, removing the need for a backend and ensuring that genomic data remain on the user's device. A Web Worker is used to maintain a responsive interface during computation. The platform has been validated with real bacterial genomes, demonstrating its viability and the correct functioning of the analysis pipeline in a fully client-side environment.

Keywords— WebAssembly, Pyodide, bioinformatics, TFBS, transcription factors, PSSM, Vue.js, serverless, Web Worker, genomic analysis



1 INTRODUCCIÓ - CONTEXT DEL TREBALL

EN els darrers anys, la bioinformàtica s'ha consolidat com una disciplina fonamental en l'àmbit de les ciències de la vida, impulsada pel creixement exponencial de dades generades per tecnologies com la seqüenciació massiva (*Next-Generation Sequencing*, NGS) [1] de DNA i RNA [2]. Per processar, analitzar i interpretar aquest vo-

lum d'informació, es necessiten eines computacionals especialitzades, habitualment distribuïdes com a aplicacions de línia de comandes configurables mitjançant fitxers estructurats com JSON o YAML. Aquesta modalitat d'ús representa una barrera d'entrada notable per als usuaris finals, principalment biòlegs i professionals de l'àmbit biomèdic, que sovint no disposen de la formació tecnològica necessària [3], limitant així l'impacte real de moltes eines innovadores dins de la comunitat científica. El problema és especialment rellevant en l'ecosistema bioinformàtic actual, basat principalment en Python i en llibreries consolidades com *Biopython* [4], àmpliament utilitzades per al processament i l'anàlisi de dades biològiques.

- E-mail de contacte: ainaescobet@gmail.com
- Menció realitzada: Tecnologies de la Informació
- Treball tutoritzat per: Ivan Erill Sagales i Maria Carmen de Toro Valdivia (Àrea de Ciències de la Computació i Intel·ligència Artificial)
- Curs 2025/26

Per mitigar aquestes limitacions han sorgit plataformes web com Galaxy [5], un entorn en línia que permet executar fluxos d'anàlisi bioinformàtics complexos mitjançant una interfície gràfica, sense necessitat de conèixer la línia de comandes. Galaxy integra un ampli conjunt d'eines, com Shovill, Snippy o SPAdes, que cobreixen tasques habituals en genòmica comparativa, anotació i processament de lectures. Tot i la seva utilitat, Galaxy es basa en una arquitectura client-servidor en què totes les dades carregades per l'usuari s'envien a un servidor remot, cosa que implica dependències d'infraestructura, costos de manteniment i, sobretot, limitacions de privadesa rellevants quan es treballa amb fitxers genòmics sensibles.

En aquest Treball de Fi de Grau s'exploren solucions tecnològiques orientades a millorar l'accessibilitat als programaris d'anàlisi genòmica, amb l'objectiu de reduir la dependència de la línia de comandes i facilitar-ne l'ús a investigadors sense formació informàtica. En particular, es desenvolupa una interfície gràfica web capaç d'executar programari Python dissenyat originalment per funcionar en entorns de terminal, oferint una experiència d'ús més intuïtiva i accessible. Tecnològicament, el projecte explora diferents aproximacions per integrar codi Python dins d'una arquitectura web moderna.

A diferència de les plataformes tradicionals, la solució proposada es basa en una arquitectura sense servidor dedicat (*serverless*), en què l'aplicació s'executa íntegrament al costat del client. D'aquesta manera, els usuaris poden executar l'anàlisi i accedir als resultats de manera directa i amigable des d'una interfície web, sense necessitat d'instal·lar programari addicional ni dependre de serveis externs, garantint alhora que els fitxers genòmics romanguin sempre al dispositiu de l'usuari [6].

Com a cas d'ús, aquest projecte desenvolupa una aplicació biomèdica que consisteix en una eina d'escaneig genòmic per a la identificació de llocs d'unió de factors de transcripció (*transcription factor binding sites*). Els factors de transcripció són proteïnes que s'uneixen a regions reguladores de l'ADN, com la regió promotora d'un gen, i regulen el procés de transcripció mitjançant mecanismes d'activació o repressió [7]. Aquests mecanismes permeten que l'expressió dels gens s'activi o es reprimeixi de manera específica segons les condicions ambientals i els senyals que rep la cèl·lula.

La resta d'aquest document presenta els objectius del projecte, la metodologia, l'arquitectura del sistema, la implementació i validació tecnològica, el funcionament del motor d'anàlisi genòmica, els resultats, les conclusions i, per acabar, l'ús que s'ha fet de la IA.

2 OBJECTIUS DEL PROJECTE

2.1 Objectiu general

L'objectiu principal d'aquest projecte és definir i implementar una solució tecnològica *serverless* que permeti executar, des del navegador, aplicacions biomèdiques desenvolupades originalment com a eines de línia de comandes. Com a cas d'ús, el projecte integra una aplicació Python d'anàlisi de seqüències genòmiques capaç d'identificar llocs d'unió de factors de transcripció mitjançant una interfície gràfica intuïtiva, orientada a facilitar la interacció de l'usuari.

2.2 Objectius específics

- Desenvolupar una aplicació de línia de comandes en Python per a la identificació de patrons de seqüències d'ADN, basada en *Biopython* [4], configurable mitjançant fitxers de configuració en format JSON o YAML i amb generació de fitxers de sortida estructurats.
- Definir un esquema de configuració que permeti desacoblar la lògica de càlcul de l'aplicació Python de la interfície d'usuari, per poder executar l'aplicació biomèdica tant des de la línia de comandes com des del portal web.
- Desenvolupar una interfície gràfica web que permeti generar automàticament els fitxers de configuració necessaris per executar l'aplicació.
- Avaluar la compatibilitat de biblioteques científiques de Python en entorns basats en el navegador.
- Estudiar les limitacions associades a les comunicacions HTTP i a les polítiques de seguretat CORS (Cross Origin Resource Sharing) en aplicacions web que integren codi Python.
- Definir una arquitectura d'integració de software entre la interfície web i l'aplicació Python que permeti executar l'anàlisi genòmic des del navegador.
- Integrar l'execució del codi Python al navegador mitjançant la tecnologia seleccionada.
- Validar la viabilitat de generar, gestionar i descarregar fitxers de resultats des de l'entorn web.
- Desplegar l'aplicació en un entorn web estàtic seguint una arquitectura *serverless* que no requereixi infraestructura de servidor dedicat.

3 METODOLOGIA

El desenvolupament d'aquest projecte presenta un cert grau d'incertesa tecnològica a causa de la integració de codi Python dins del navegador. Aquesta integració pot implicar certes limitacions en la compatibilitat de llibreries, la gestió de dependències i les restriccions de seguretat dels navegadors web. Per aquest motiu, la metodologia de desenvolupament a seguir és de tipus iteratiu i incremental [8]. Aquest enfocament permet construir el projecte de manera progressiva amb prototips funcionals que es van ampliant i corregint a mesura que es van validant els components. Aquesta metodologia es basa en els següents principis:

- Desenvolupament de prototips funcionals mínims per validar cada component del sistema.
- Validació progressiva de les tecnologies crítiques pel desenvolupament del projecte.
- Evolució incremental de l'arquitectura del sistema.
- Identificació i mitigació primerenca de riscos tècnics.

Aquest model permet una adaptació progressiva entre els components implementats en Python i la interfície web, facilitant un desenvolupament flexible i escalable que s'ajusta a les possibles limitacions identificades durant les diferents fases d'avaluació tecnològica.

3.1 Planificació temporal

El desenvolupament del projecte es planifica al llarg de dinou setmanes, estructurades en diverses fases. La planificació temporal del projecte es pot observar a la Fig. A.9, inclosa en l'Apèndix.

Les fases de planificació són les següents:

Fase 1 – Anàlisi tecnològica inicial (setmanes 1–2)

- Anàlisi del projecte proposat.
- Estudi d'entorns per executar Python al navegador.
- Comparació d'alternatives i selecció final.
- Revisió de la biblioteca *Biopython* i del mòdul Entrez.

Fase 2 – Prototip i detecció de limitacions (setmanes 3–4)

- Desenvolupament d'un prototip simple amb Pyodide.
- Proves d'ús d'Entrez per accedir a API externes.
- Identificació de problemes de compatibilitat amb CORS i *urllib* [9] per accedir a API externes.

Fase 3 – Redisseny de funcionalitats d'Entrez (setmanes 5–6)

- Estudi d'alternatives per fer peticions HTTP a API externes des del navegador.
- Adaptació de les funcionalitats d'Entrez incompatibles amb WebAssembly.
- Implementació de funcions equivalents per a peticions GET i POST.

Fase 4 – Desenvolupament de l'aplicació d'anàlisi genòmica (setmanes 7–11)

- Desenvolupament de l'aplicació de línia de comandes.
- Implementació dels mètodes d'identificació de patrons d'ADN.
- Integració de *Biopython* per al processament de seqüències.
- Generació de fitxers de sortida estructurats.

Fase 5 – Desenvolupament de la interfície web (setmanes 12–15)

- Desenvolupament de la interfície gràfica.
- Generació de fitxers de configuració.
- Integració amb el motor Python executat amb Pyodide.

Fase 6 – Integració del sistema (setmana 16)

- Integració completa dels components del sistema.
- Implementació de la descàrrega dels resultats

Fase 7 – Validació final i finalització (setmanes 17–19)

- Validació funcional amb dades de prova.
- Optimització i resolució d'errors.
- Finalització de la memòria i la defensa del projecte.

4 ARQUITECTURA DEL SISTEMA

4.1 Visió general de l'arquitectura

L'arquitectura del sistema es basa en un model completament client-side, en el qual tota l'aplicació s'executa dins del navegador web. Aquesta aproximació respon al requisit funcional d'una arquitectura *serverless*, eliminant la necessitat de *backend* dedicat i permetent un desplegament basat en contingut estàtic.

El sistema està compost de tres elements principals. En primer lloc, una interfície d'usuari desenvolupada com a *Single-Page Application* (SPA) per configurar i executar una aplicació biomèdica. El model SPA [10] és el més utilitzat quan es tracta d'oferir una experiència d'usuari (UX/UI) similar a la d'una aplicació d'escriptori. En segon lloc, el mòdul Pyodide, que consisteix en un entorn d'execució de Python compilat a WebAssembly i executat directament dins del navegador. Finalment, l'aplicació Python encarregada del processament i l'anàlisi de seqüències genòmiques.

La Figura 1 mostra l'estructura general del sistema i la relació entre els seus components principals.

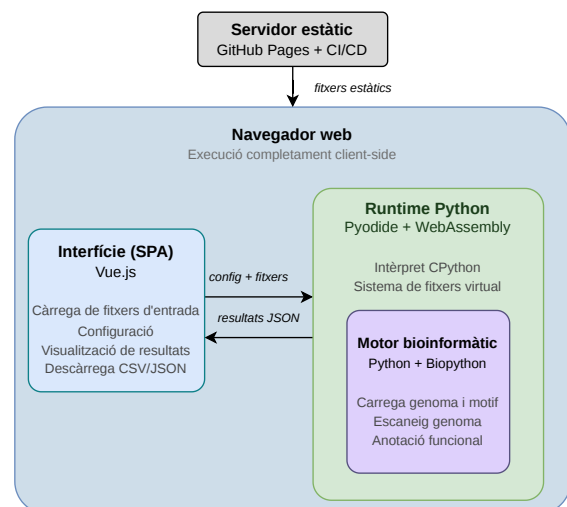


Fig. 1: Arquitectura general del sistema client-side.

El desplegament de l'aplicació es realitza mitjançant GitHub Pages, cosa que permet el desplegament d'una aplicació completament estàtica sense necessitat de servidor backend.

Per automatitzar el procés de construcció i publicació, s'utilitza GitHub Actions com a sistema d'integració i desplegament continu (CI/CD). Com es pot observar a la Figura 2, amb GitHub Actions es defineix un pipeline que s'encarrega d'executar les fases de validació del codi, construcció de la SPA i publicació automàtica en cada actualització del repositori.

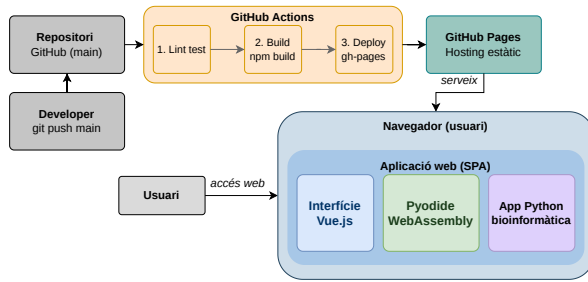


Fig. 2: Pipeline CI/CD amb GitHub Actions: des del *push* al repositori fins al desplegament de la SPA a GitHub Pages.

Aquest enfocament garanteix que cada desplegament es realitzi de manera consistent, reduint els errors humans en el procés i facilitant la iteració contínua sobre l'aplicació.

4.2 Processament bioinformàtic

El motor d'anàlisi està orientat a la identificació computacional de regions reguladores en seqüències genòmiques. Aquest procés es basa en la detecció de *motifs* (patrons) de seqüència [11], representats habitualment mitjançant matrius específiques de posició de freqüència o de puntuació (PSFM/PSSM), que modelen quantitativament la probabilitat que una seqüència d'ADN constitueixi un lloc d'unió per a un factor de transcripció determinat [12].

Aquestes matrius s'obtenen a partir de conjunts de seqüències d'unió coneguts, derivats d'experiments genòmics com ChIP-seq [13] o SELEX [14], i constitueixen la base matemàtica del procés d'escaneig del genoma. La Figura 3 il·lustra aquest procés de manera conceptual.

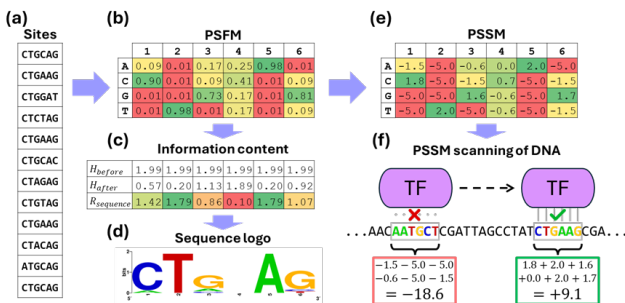


Fig. 3: Procés de construcció de la PSSM i escaneig de seqüències d'ADN per a la detecció de llocs d'unió de factors de transcripció.

A partir d'aquestes representacions, és possible escanejar seqüències genòmiques i predir regions candidates amb potencial funció reguladora sobre l'expressió genètica.

Aquest tipus d'anàlisi es desenvolupa habitualment utilitzant eines i llibreries de l'ecosistema Python, especialment dins l'àmbit bioinformàtic, fet que motiva la necessitat d'integrar un entorn d'execució Python directament al navegador.

En la implementació final, aquest pipeline s'ha dissenyat de manera incremental. Cada etapa del processament s'executa de manera independent i només es recalcula quan els paràmetres associats canvien. Aquesta estratègia permet reduir el temps de càlcul i reutilitzar resultats intermedis entre execucions consecutives, fet especialment rellevant en entorns d'execució dins del navegador.

4.3 Execució de Python al navegador

L'execució del codi Python al navegador es basa en la combinació de tres tecnologies complementàries: WebAssembly, Pyodide i JavaScript.

WebAssembly (Wasm) és la base de l'execució. Es tracta d'un format binari de baix nivell capaç d'executar codi dins del navegador web amb un rendiment relativament proper al natiu, proporcionant un espai aïllat i segur per a aplicacions d'alt rendiment [15].

Pyodide és una adaptació de l'interpret estàndard de Python (CPython) compilada a WebAssembly [16]. En carregar Pyodide, el navegador descarrega un entorn complet de Python empaquetat en un mòdul Wasm en el qual s'inclouen llibreries científiques habituals utilitzades en bioinformàtica, com ara *Biopython*.

JavaScript actua com a capa d'orquestració del sistema. És el responsable d'inicialitzar l'entorn Pyodide, enviar-hi el codi Python a executar i recollir-ne els resultats per integrar-los amb la interfície gràfica del navegador (vegeu Fig. 4).

El flux d'execució dins del navegador segueix una seqüència clara. En primer lloc, JavaScript carrega i inicialitza l'entorn Pyodide, que conté una versió de CPython compilada a WebAssembly. Aquest procés implica la descàrrega del mòdul Wasm, que es realitza només una vegada durant la inicialització. Quan l'usuari inicia el procés de cerca de llocs d'unió, JavaScript construeix dinàmicament el bloc de codi Python amb els paràmetres i les dades seleccionades i l'envia a l'interpret de Pyodide. Aquest executa el pipeline d'anàlisi genòmica dins del seu entorn aïllat, utilitzant les llibreries carregades en el runtime. Un cop finalitzat el càlcul, els resultats se serialitzen en format JSON i es retornen a JavaScript, que els processa i els integra a la interfície desenvolupada amb Vue.js.

En una primera iteració, Pyodide s'inicialitzava directament al fil principal del navegador. Tot i ser funcional, aquest enfocament bloquejava la interfície durant l'execució del pipeline, ja que JavaScript utilitza un únic fil d'execució. Per resoldre aquest bloqueig, l'arquitectura final executa Pyodide dins d'un Web Worker [17], un mecanisme que permet executar codi en un fil separat i mantenir la interfície responsiva. La comunicació entre el fil principal i el Worker es realitza mitjançant missatges asíncrons, seguint el model d'esdeveniments de JavaScript. Per evitar la càrrega repetida de l'entorn Python, el Worker s'instància una única vegada durant tota la sessió, seguint un patró *singleton* que garanteix una inicialització eficient i reutilitzable.

La Figura 4 mostra el flux complet d'execució entre la interfície Vue.js (JavaScript), el WebAssembly i l'entorn Pyodide dins del navegador. En iniciar l'aplicació, el Worker carrega Pyodide i les llibreries necessàries dins del navegador, mantenint una única instància activa durant tota la sessió. Quan l'usuari comença l'anàlisi, Vue.js envia un missatge al Worker amb els fitxers i paràmetres, aquest escriu els fitxers al sistema de fitxers virtual de Pyodide, executa el pipeline Python i retorna els resultats en format JSON. Els missatges de progrés i resultats es comuniquen de manera asíncrona, garantint que la interfície es mantingui responsiva durant l'execució.

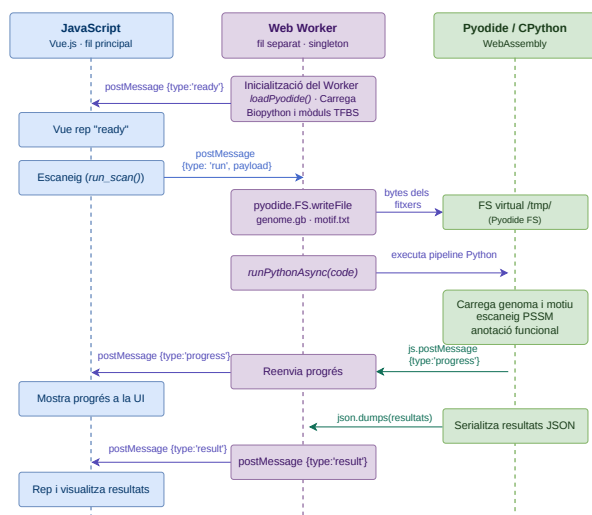


Fig. 4: Interacció entre Vue.js, el Web Worker i Pyodide dins del navegador per a l'execució client-side del pipeline Python.

La gestió de dades es realitza íntegrament al navegador, incloent-hi la càrrega dels fitxers d'entrada a l'aplicació Python, la generació de configuracions i la descàrrega de resultats, evitant així la dependència de serveis externs.

Tot i els avantatges d'aquest enfocament, cal considerar algunes limitacions, com la compatibilitat parcial de les llibreries Python amb WebAssembly, el consum de recursos al dispositiu client i el temps de càrrega inicial de l'entorn d'execució.

4.4 Stack tecnològic

D'acord amb l'arquitectura client-side definida anteriorment, a la secció 4.1, l'aplicació s'implementa com una *Single-Page Application* (SPA). Aquesta aproximació permet gestionar la navegació i l'actualització del contingut directament al navegador, reduint les recàrregues completes de pàgina i millorant la fluïdesa de l'experiència d'usuari.

Per al desenvolupament del frontend s'ha escollit el framework Vue.js [18]. Aquesta elecció es justifica per la seva corba d'aprenentatge reduïda, la simplicitat del seu model reactiu i l'organització clara del codi basada en components, que facilita el desenvolupament incremental de la interfície i el manteniment del codi.

El desenvolupament de la interfície es basa en l'ús de *Single File Components* (SFC), que encapsulen la plantilla HTML, la lògica JavaScript i els estils CSS dins d'un únic fitxer. Aquesta aproximació afavoreix la separació de responsabilitats, la reutilització de components i el manteniment evolutiu de l'aplicació, cosa que permet dividir funcionalitats complexes en unitats independents.

Pel que fa al processament bioinformàtic, l'aplicació utilitza l'entorn descrit prèviament, a la secció 4.3, basat en la combinació de WebAssembly, Pyodide i JavaScript. Aquesta integració permet executar codi Python directament des del frontend, usant JavaScript com a capa d'orquestració que inicialitza Pyodide, carrega dades d'entrada, invoca els processos d'anàlisi genòmica i integra els resultats dins la interfície de la SPA. Gràcies a aquest enfocament, és possible fer servir llibreries científiques com *Biopython* man-

tenint una arquitectura completament *client-side* sense dependència d'un backend tradicional.

4.5 Arquitectura de la interfície web

Un cop desenvolupat i validat el motor d'anàlisi, l'objectiu principal ha estat permetre que l'usuari pugui executar el pipeline sense necessitat de conèixer Python ni d'utilitzar la línia de comandes. Per aquest motiu, el sistema s'ha integrat dins d'una aplicació web implementada amb el framework Vue.js, dissenyada per oferir una experiència d'usuari intuïtiva i eficient (UX/UI).

La integració entre Vue.js i Pyodide permet executar el pipeline bioinformàtic directament des del navegador, integrant la interfície d'usuari amb el motor d'anàlisi. Tal com es mostra a la Figura 1, l'aplicació segueix un model de *Single-Page Application* (SPA) i es divideix en components independents que representen les diferents etapes del flux d'anàlisi: càrrega del genoma, càrrega del motiu, configuració dels paràmetres, execució del pipeline i visualització dels resultats. Aquesta organització facilita el manteniment del codi i permet desacoblar la interfície de la lògica bioinformàtica. Un cop definida l'organització de la interfície, en la següent secció descrivim el mecanisme de comunicació entre els components Vue i l'entorn d'execució Python.

4.6 Arquitectura del Web Worker

La comunicació entre la interfície Vue.js i el motor bioinformàtic es realitza mitjançant un Web Worker compartit que encapsula l'execució de Pyodide. Aquest Worker s'inicialitza una única vegada durant la càrrega de l'aplicació i es reutilitza durant tota la sessió, actuant com a capa intermèdia entre la interfície i l'entorn Python.

Els components Vue no interactuen directament amb Pyodide, sinó que utilitzen un servei JavaScript centralitzat que encapsula tota la comunicació amb el Worker. Aquest servei exposa operacions d'alt nivell, com la validació de genomes, la validació d'accions, la càrrega de motius i l'execució del pipeline d'anàlisi.

El Worker rep missatges que especifiquen l'operació a executar. Segons el tipus de petició, escriu els fitxers d'entrada al sistema de fitxers virtual de Pyodide, executa el codi Python corresponent i retorna els resultats a la interfície.

Aquesta arquitectura permet reutilitzar una única instància de Pyodide per a totes les operacions, evitant la seva reinicialització i reduint significativament el temps d'execució. A més, l'ús d'un Web Worker garanteix que les tasques computacionalment intenses es realitzin en un fil d'execució independent, evitant bloquejos de la interfície i mantenint una experiència d'usuari fluida.

4.7 Integració entre Vue i Pyodide

La comunicació entre Vue i Pyodide s'ofereix mitjançant un servei JavaScript centralitzat encarregat d'inicialitzar l'entorn Python, gestionar els fitxers d'entrada i executar el pipeline incremental. Aquest servei encapsula les funcionalitats principals de Pyodide i exposa una interfície reutilitzable per a la resta de components de l'aplicació Vue, tal com s'ha introduït a la Figura 4.

La inicialització de Pyodide es duu a terme de manera asíncrona durant la càrrega inicial de l'aplicació, moment en què es descarreguen l'entorn WebAssembly i les dependències necessàries. Un cop inicialitzat, l'entorn Python es manté persistent durant tota la sessió de l'usuari. La interfície Vue.js centralitza la comunicació amb Pyodide: recull les accions de l'usuari, prepara les dades necessàries i construeix la configuració requerida per executar el pipeline. Els resultats generats se serialitzen en format JSON i es transfereixen de nou a l'entorn JavaScript perquè Vue els pugui processar i representar. El model reactiu de Vue permet que qualsevol modificació de l'estat de l'aplicació actualitzi automàticament els components visuals corresponents. Això facilita la sincronització entre els resultats retornats per Pyodide i la seva representació a la interfície, sense necessitat de manipulació manual del DOM.

Per permetre que el pipeline treballi amb fitxers carregats per l'usuari, s'utilitza el sistema de fitxers virtual de Pyodide. Aquest sistema és temporal, resideix completament en memòria i desapareix quan es recarrega la pàgina. Els fitxers seleccionats des de la interfície es processen mitjançant l'API `FileReader` de JavaScript i s'escriuen dins del sistema de fitxers virtuals de Pyodide, de manera que Python els pot utilitzar com si fossin fitxers locals.

Aquesta estratègia permet mantenir intacta la lògica interna del pipeline desenvolupat originalment en Python. Gràcies al fet que Pyodide proporciona un entorn compatible amb el sistema de fitxers habitual de Python, el pipeline pot operar sobre els fitxers carregats des del navegador com si fossin fitxers locals convencionals. Això evita la necessitat de reescriure la lògica de processament per adaptar-la al navegador i facilita la reutilització directa de llibreries bioinformàtiques, com ara *Biopython*, sense modificacions específiques. En conseqüència, la implementació executada al navegador manté el mateix comportament i estructura que la implementació original desenvolupada per a la línia de comandes, fet que simplifica el manteniment del codi i redueix la complexitat associada a la integració entre la interfície web i el motor d'anàlisi. Addicionalment, el processament local de totes les dades, sense necessitat d'enviar informació a servidors externs, contribueix a reforçar la privacitat i la confidencialitat de les dades de l'usuari.

Tot i els avantatges del model client-side, Pyodide presenta diverses limitacions derivades de l'execució dins del navegador, com ara restriccions de memòria, absència de *multithreading* real i un temps d'inicialització superior al d'una aplicació nativa. Aquestes limitacions han condicionat diverses decisions d'optimització del projecte, especialment la implementació del pipeline incremental i la reutilització d'estat entre execucions.

Per tal d'evitar re-computacions innecessàries, el pipeline manté un estat persistent dins de l'entorn Pyodide durant tota la sessió. Aquest estat inclou objectes ja carregats, com el genoma analitzat, el motiu processat o els líndars calculats. Quan l'usuari modifica només alguns paràmetres, el sistema reutilitza les etapes ja computades i només recalcula les parts afectades. Aquesta estratègia és essencial per minimitzar el temps d'execució en un entorn amb recursos limitats com el navegador.

5 IMPLEMENTACIÓ I VALIDACIÓ TECNOLÒGICA

Un dels principals reptes tecnològics del projecte consisteix en la integració de biblioteques bioinformàtiques Python dins d'un entorn d'execució basat en navegador. Tot i que Pyodide permet executar Python mitjançant WebAssembly, no totes les funcionalitats disponibles en entorns tradicionals de Python són directament compatibles amb el model d'execució web.

Durant les primeres fases del desenvolupament es va analitzar la viabilitat d'utilitzar el mòdul *Entrez* de *Biopython* [19] per obtenir genomes bacterians a partir de consultes a les API de repositoris públics com National Center for Biotechnology Information (NCBI) [20].

A continuació es descriuen els principals problemes trobats en la utilització de la llibreria *Entrez* dins de Pyodide i el procés d'avaluació per trobar una solució final.

5.1 Limitacions detectades

El mòdul *Entrez* utilitza internament mecanismes de comunicació HTTPS basats en la llibreria *urllib* [9]. Aquesta llibreria depèn de l'ús directe de sockets de xarxa i de l'accés directe a la pila TCP/IP del sistema operatiu, un model propi dels entorns d'execució nadius de Python.

Aquesta arquitectura no és compatible amb l'entorn on s'executen les aplicacions web modernes, el qual està altament restringit per motius de seguretat. En primer lloc, els navegadors imposen el model de seguretat *Same-Origin Policy* (SOP), implementat mitjançant polítiques CORS (*Cross-Origin Resource Sharing*). Aquest mecanisme estableix que una aplicació web només pot realitzar peticions HTTP cap al mateix domini des del qual s'ha carregat la pàgina. El navegador únicament permet comunicacions amb dominis externs si aquests inclouen capçaleres CORS específiques, com *Access-Control-Allow-Origin*, indicant explícitament que accepten peticions provinents d'altres dominis. En segon lloc, WebAssembly executa el codi dins d'un entorn aïllat (*sandboxed*), és a dir, completament aïllat del sistema operatiu per motius de seguretat. Aquest model impedeix l'accés directe a recursos de baix nivell com sockets de xarxa, fitxers locals o la pila TCP/IP del sistema. Com a conseqüència, totes les operacions d'entrada/sortida han de ser gestionades a través de les API del navegador, com *fetch* [21] o *XMLHttpRequest* [22], que estan sotmeses a les mateixes restriccions de seguretat (SOP i CORS). Aquest conjunt de restriccions implica que llibreries com *urllib*, que assumeixen accés directe a sockets de xarxa, no poden funcionar correctament dins de Pyodide.

Finalment, cal considerar el model d'execució de JavaScript utilitzat pel navegador. Aquest es basa en un únic fil d'execució (*single-thread*) i un mecanisme d'orquestració anomenat *event loop* [23]. Les operacions potencialment bloquejants, com les peticions de xarxa, s'executen de manera asíncrona per evitar bloquejar la interfície d'usuari. En canvi, moltes biblioteques científiques de Python, incloent-hi *Biopython*, es basen en un flux d'execució síncron i seqüencial. Aquesta diferència entre el model asíncron del navegador i el model síncron esperat per *Biopython* provoca incompatibilitats directes, especialment

en les consultes a serveis externs. A conseqüència d'aquestes limitacions, les peticions HTTPS realitzades mitjançant el mòdul Entrez no es poden executar correctament dins de l'entorn Pyodide.

5.2 Estratègies explorades

Per resoldre aquestes limitacions, es van estudiar diversos mecanismes de comunicació HTTP compatibles amb el navegador. Una comparació detallada de les alternatives analitzades es mostra a la Taula 1, inclosa en l'Apèndix. A continuació, enumerem els mecanismes de comunicació analitzats:

- **open_url**: llibreria inicial proporcionada per Pyodide que permet realitzar peticions HTTP GET utilitzant el sistema de xarxa del navegador. Tot i funcionar en casos simples, presenta limitacions importants en peticions POST i en el processament de respostes XML [24].
- **pyfetch**: llibreria moderna que proporciona una interfície basada en l'API *fetch()* del navegador. Ofereix suport complet per als mètodes HTTP (GET i POST), però introdueix un model d'execució asíncron que no és compatible amb l'estructura seqüencial del pipeline bioinformàtic, el qual espera poder realitzar les peticions dins d'un flux estrictament síncron [16].
- **Wrapper asíncron**: mecanisme d'encapsulament d'una funció asíncrona dins d'una interfície aparentment síncrona per permetre la comunicació amb les API del navegador. Tot i simplificar parcialment la integració, la funció continua essent asíncrona, fet que obliga a transformar tot el flux d'execució de l'aplicació cap a un model asíncron. Aquest enfocament és incompatible amb el flux d'execució síncron que requereix la utilització de la llibreria de *Biopython*.
- **run_sync (solució adoptada)**: funcionalitat experimental *run_sync* de Pyodide, que permet executar operacions asíncrones del navegador mantenint una interfície aparentment síncrona des del punt de vista del codi Python. Aquest mecanisme permet mantenir la compatibilitat amb la funcionalitat síncrona tradicional de *Biopython*, executar peticions GET i POST des del navegador i evitar la reescriptura completa de l'aplicació en un model asíncron. Tot i ser una funcionalitat encara en evolució, Pyodide la preveu consolidar i estabilitzar en versions futures, fet que reforça la seva viabilitat a llarg termini [25].

En conjunt, l'ús de *run_sync* es va demostrar com l'única estratègia capaç de satisfer simultàniament els requisits de compatibilitat amb *Biopython*, la necessitat d'un flux d'execució síncron i les limitacions inherents a l'entorn d'execució del navegador.

A partir d'aquesta adaptació, es va implementar una llibreria síncrona compatible amb el model asíncron de JavaScript que reproduïx les funcionalitats principals d'Entrez utilitzades per *Biopython*. Això permet l'accés a API externes per a la recuperació automàtica de genomes bacterians dins d'una arquitectura completament *serverless*.

6 FUNCIONAMENT DEL MOTOR D'ANÀLISI GENÒMICA

El motor d'anàlisi genòmica desenvolupat en Python consisteix en el nucli computacional del projecte i és l'encarregat d'executar totes les operacions bioinformàtiques necessàries per identificar i anotar llocs d'unió de factors de transcripció. El seu funcionament es basa en un pipeline seqüencial que reproduïx el flux de treball habitual en eines bioinformàtiques d'escaneig genòmic, però adaptat a les particularitats d'un entorn d'execució completament client-side com Pyodide. El pipeline s'organitza en quatre etapes principals: la càrrega del genoma, la construcció del motiu, l'escaneig del genoma i l'anotació funcional dels resultats (vegeu Fig. A.10 i Fig. A.11).

A més, el sistema incorpora un model incremental que evita re-computacions innecessàries (secció 6.5 i millora significativament el rendiment en un entorn d'execució limitat com el navegador. Es pot veure el funcionament del sistema en la Figura A.11.

6.1 Càrrega i representació del genoma

El sistema admet tres fonts principals de dades genòmiques:

- Fitxers GenBank [26] proporcionats per l'usuari.
- Identificadors d'accés NCBI, que permeten recuperar automàticament els genomes amb les funcionalitats adaptades d'Entrez (vegeu Secció 5.1).
- Cerca per nom d'espècie mitjançant l'API REST de NCBI Datasets, que permet a l'usuari cercar assemblatges genòmics de referència a partir del nom d'un organisme, seleccionar l'assemblatge desitjat d'una llista i descarregar automàticament tots els fitxers associats a les seqüències nucleotídiques.

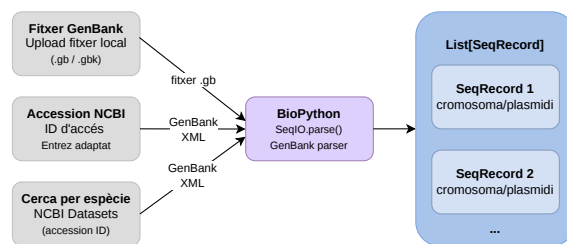


Fig. 5: Fonts de dades genòmiques admeses pel sistema i procés de càrrega.

Com es mostra en la Figura 5, el contingut es processa amb *Biopython* i es converteix en una llista d'objectes *SeqRecord*, cadascun corresponent a un cromosoma o un plasmidi. Aquesta estructura conté la seqüència completa i les anotacions disponibles (gens, CDS, coordenades i orientació). Aquestes anotacions constitueixen la base de totes les etapes posteriors del pipeline, garantint la coherència amb les dades de les fonts oficials.

6.2 Construcció del motiu i la PSSM

El motiu representa el patró de seqüència característic del lloc d'unió d'un factor de transcripció. El sistema admet diversos formats d'entrada (FASTA, TXT, JASPAR) i els

converteix en una representació unificada que permet generar una matriu de puntuació específica de posició (PSSM), com s'explica a continuació. A partir de les seqüències del motiu, es construeix una matriu de freqüències per posició i s'hi apliquen *pseudocounts* per evitar valors nuls [27]. Opcionalment, es poden incorporar probabilitats de fons per ajustar el model a la composició nucleotídica del genoma. Finalment, aquestes freqüències es transformen en puntuacions logarítmiques, obtenint una PSSM que quantifica la probabilitat relativa que una subseqüència del genoma correspongui a un lloc d'unió. La Figura 3 mostra el procés complet de construcció de la PSSM i la seva aplicació mitjançant una tècnica de *sliding window* sobre seqüències d'ADN.

6.3 Escaneig del genoma

L'escaneig del genoma és la fase central del *pipeline*. Consisteix a recórrer la seqüència de nucleòtids utilitzant una finestra lliscant (*sliding window*) de la mateixa longitud que el motiu. Per a cada posició, s'extreu la subseqüència corresponent, se'n calcula la puntuació amb la PSSM i es repeteix el procés per la cadena complementària inversa del genoma.

La puntuació obtinguda es compara amb un llindar (*threshold*), que pot ser numèric, estadístic o derivat de mètodes clàssics com Patser [28]. Quan la puntuació supera el llindar, el sistema registra un *hit* (coincidència potencial amb el motiu biològic) amb la informació necessària perquè aquest lloc s'anoti com a lloc d'unió. Aquest procés permet identificar totes les regions del genoma que tenen una probabilitat elevada de correspondre a un lloc d'unió.

6.4 Anotació funcional dels *hits*

Un cop identificats els llocs potencials d'unió (*hits*), el sistema determina quin gen o quina regió reguladora podria estar afectada. En aquest context, un *hit* correspon a una subseqüència del genoma que supera el llindar de puntuació establert i que, per tant, constitueix una possible regió d'unió del factor de transcripció. Com que un lloc d'unió pot regular gens situats en qualsevol de les dues cadenes del genoma, el sistema avalua la posició de cada hit tant respecte als gens orientats en direcció *forward* com als orientats en direcció *reverse*. Per fer-ho, es calcula la distància entre el hit i el lloc d'inici de traducció (TLS) del gen més proper en cada direcció. A partir d'aquesta informació, cada hit es classifica en una de les tres categories representades a la Figura 6:

1. **Operator:** El lloc d'inici de transcripció del gen es troba dins de la finestra reguladora definida.
2. **Intragenic:** el hit es troba dins de la regió de codificació d'un gen.
3. **Intergenic:** el hit es troba fora de les regions de codificació i fora de la finestra reguladora.

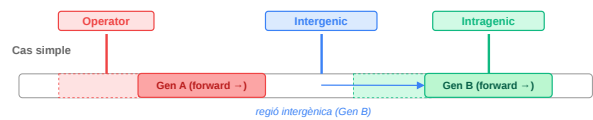


Fig. 6: Classificació d'un hit segons la seva posició.

En alguns casos poden aparèixer ambigüitats, com ara regions divergents, on dos gens adjacents es transcriuen en direccions oposades i comparteixen una mateixa regió reguladora, o situacions en què un mateix *hit* és operador d'un gen però intragènic d'un altre. Aquests casos es mostren esquemàticament a la Figura 7.

Davant d'aquests casos, el sistema pot resoldre l'ambigüitat de dues maneres: seleccionar el candidat més probable o bé retornar el candidat més probable per a cadascuna de les dues cadenes (forward i reverse), segons la configuració escollida per l'usuari.

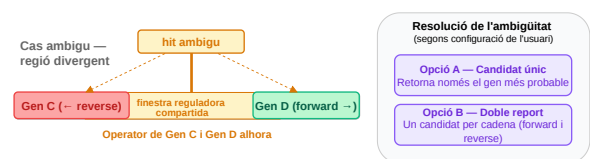


Fig. 7: Classificació d'un hit ambigu segons la seva posició.

Dins del genoma, un operó és un conjunt de gens adjacents que es transcriuen de manera conjunta sota el control d'un mateix promotor. Quan l'usuari ho requereix, el sistema és capaç d'inferir possibles operons mitjançant l'agrupació de gens adjacents que comparteixen la mateixa orientació i es troben separats per distàncies reduïdes, la qual cosa suggereix que poden formar part d'una mateixa unitat transcripcional. La Figura 8 il·lustra aquest procés d'inferència i mostra com es determinen els límits d'un operó a partir de la distància i l'orientació dels gens.



Fig. 8: Esquema d'inferència d'operons a partir de l'orientació i la distància entre gens adjacents.

6.5 Funcionament intern del *pipeline*

El motor de processament manté un estat intern persistent que evita recalculer etapes que no han canviat. Aquest estat inclou el genoma carregat, el motiu i la seva PSSM, els *hits* detectats, l'anotació funcional i els paràmetres utilitzats.

Quan l'usuari modifica algun paràmetre, només es recalculen les etapes estrictament necessàries, tal com es mostra a la Taula 2 de l'Apèndix. Aquest model és especialment adequat per a entorns com Pyodide, on les operacions costoses poden afectar l'experiència d'usuari.

El flux general del pipeline es resumeix en el diagrama de les Figures A.10 A.11, que mostren com s'encadenen les diferents etapes del procés. Així mateix, es mostra com el disseny d'execució incremental permet reutilitzar els resultats previs en els càlculs on no s'ha modificat cap paràmetre.

Aquesta estructura garanteix un funcionament fluid i estable, i afavoreix la integració amb la interfície Vue.js, aportant una bona experiència d'usuari.

7 RESULTATS

7.1 Validació del pipeline amb dades reals

Per validar el funcionament del pipeline, es va dur a terme una avaluació utilitzant genomes i conjunts de llocs d'unió per als quals es disposava de resultats de referència. Aquest enfocament va permetre comparar les prediccions generades pel sistema amb dades prèviament validades i verificar que la detecció de hits, les puntuacions PSSM i l'atribució funcional coincidien amb el comportament esperat.

Es van definir dues configuracions representatives. La primera utilitzava el genoma *Escherichia coli K-12 MG1655* (*E. Coli*) i el motiu regulador LexA específic d'aquesta espècie. Els llocs d'unió coneguts per a aquest regulador en *E. coli* estan recollits a la base de dades CollecTF [29] i serveixen com a referència per validar la qualitat del motiu i dels resultats obtinguts. Aquest escenari representa un cas amb un motiu altament conservat i àmpliament caracteritzat. La segona configuració es va aplicar al plasmidi C de *Cereibacter sphaeroides* 2.4.1 utilitzant un conjunt de motius LexA d'Alphaproteobacteria. Aquest és un cas més heterogeni que requereix paràmetres més permissius.

En ambdós casos es van activar la inferència d'operons i els marges reguladors estàndard ($-250/+50$), i es van generar resultats en formats CSV i JSON. Els resultats obtinguts es van comparar amb els resultats de referència descrits anteriorment per verificar que els hits detectats, les puntuacions PSSM, i l'anotació funcional coincidien amb el comportament esperat. Aquesta comparació va confirmar que el pipeline funciona correctament tant en contextos de motius molt conservats com en escenaris més diversos i biològicament complexos.

7.2 Flux d'execució i gestió de resultats

El flux d'ús de l'aplicació segueix el funcionament del pipeline bioinformàtic (vegeu Fig. A.10 i Fig. A.11). L'usuari carrega un genoma, selecciona un motiu, configura els paràmetres de l'anàlisi i inicia l'execució del pipeline. Els resultats es generen íntegrament al navegador i es poden descarregar en formats estàndard com CSV o JSON o es poden visualitzar directament a la pàgina web.

La interfície reutilitza la mateixa instància de Pyodide durant tota la sessió de l'usuari, evitant la reinicialització completa de l'entorn Python en cada execució. Això permet mantenir l'estat intern del pipeline i reduir el temps de resposta durant les execucions successives.

Aquesta estratègia manté una arquitectura completament *serverless* i garanteix que totes les dades es processen localment, sense dependències externes ni comunicacions de xarxa durant l'anàlisi, fet que contribueix a preservar la privacitat de l'usuari.

7.3 Resultats de la interfície web

Les pantalles principals de la interfície web permeten seguir tot el flux d'anàlisi de manera intuïtiva. La Figura A.12 mostra els diferents tipus d'entrada disponibles tant per carregar el genoma com per introduir el motiu, incloent-hi la selecció de fitxers GenBank, la recuperació automàtica via NCBI i la càrrega manual de seqüències de motiu. Un cop definides les dades d'entrada, l'usuari pot configurar

els paràmetres de l'anàlisi, lindars, *pseudocounts*, marges reguladors i opcions d'atribució tal com es mostra a la Figura A.13. Finalment, la Figura A.14 presenta la visualització dels resultats generats pel pipeline, amb una taula interactiva que permet filtrar, explorar i descarregar la informació obtinguda.

8 CONCLUSIONS

El projecte ha permès validar que és viable portar al navegador eines bioinformàtiques tradicionalment dependents d'un entorn Python natiu. La combinació de Pyodide, WebAssembly i Vue ha permès construir una aplicació completament *serverless* capaç d'executar un pipeline d'anàlisi genòmic real sense necessitat de backend.

La integració del motor d'anàlisi genòmic amb la interfície web ha estat possible sense modificar la lògica bioinformàtica implementada durant el desenvolupament del motor d'anàlisi, gràcies a l'ús del sistema de fitxers virtual i a la conversió transparent de dades entre JavaScript i Python. El model incremental i reutilitzable implementat al motor d'anàlisi genòmic, juntament amb l'execució en Web Workers, ha contribuït significativament a millorar el rendiment i l'experiència d'usuari, especialment en un entorn on els recursos són limitats i l'execució es realitza íntegrament al client.

Podem concloure que l'enfocament adoptat és sòlid i escalable, i que pot servir de base per a futures eines bioinformàtiques accessibles des del navegador. Tot i això, caldrà continuar treballant en aspectes com la compatibilitat de llibreries científiques, l'optimització del temps de càrrega inicial de Pyodide i la gestió de genomes de mida superior a la dels genomes utilitzats durant la validació.

L'aplicació desenvolupada en aquest treball es troba disponible públicament a <https://erilllab.github.io/TFBS-search/>.

9 ÚS DE LA IA GENERATIVA

Durant el desenvolupament del projecte i de la memòria, s'ha fet servir intel·ligència artificial generativa com a eina de suport. El seu ús s'ha centrat principalment en tasques de redacció, revisió estilística, estructuració de continguts i resolució de dubtes puntuals relacionats amb tecnologies específiques. La IA ha servit com a assistent per accelerar processos de documentació i per generar explicacions preliminars que, posteriorment, han estat revisades, corregides i adaptades manualment.

REFERÈNCIES

- [1] H. Satam, K. Joshi, U. Mangrolia, S. Waghoo, G. Zaidi, S. Rawool, R. P. Thakare, S. Bandy, A. K. Mishra, G. Das, and S. K. Malonia, "Next-Generation Sequencing Technology: Current Trends and Advancements," *Biology*, vol. 12, no. 7, p. 997, Jul. 2023.
- [2] S. Rubio, R. A. Pacheco-Orozco, A. Milena Gómez, S. Perdomo, and R. García-Robles, "Secuenciación de nueva generación (NGS) de ADN: presente y futuro en la práctica clínica," *Universitas Medica*, vol. 61, no. 2, pp. 49–63, Jun. 2020.

- [3] D. Bolchini, A. Finkelstein, V. Perrone, and S. Nagl, "Better bioinformatics through usability analysis," *Bioinformatics*, vol. 25, no. 3, pp. 406–412, Feb. 2009.
- [4] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. L. de Hoon, "Biopython: Freely available Python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, Jun. 2009.
- [5] E. Afgan, D. Baker, B. Batut, M. van den Beek, D. Bouvier, M. Čech, J. Chilton, D. Clements, N. Coraor, B. A. Grüning, A. Guerler, J. Hillman-Jackson, S. Hiltemann, V. Jalili, H. Rasche, N. Soranzo, J. Goecks, J. Taylor, A. Nekrutenko, and D. Blankenberg, "The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update," vol. 46, no. W1, pp. W537–W544. [Online]. Available: <https://doi.org/10.1093/nar/gky379>
- [6] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich, "Identifying Personal Genomes by Surname Inference," *Science*, vol. 339, no. 6117, pp. 321–324, Jan. 2013.
- [7] P. K. O'Neill, R. Forder, and I. Erill, "Informational Requirements for Transcriptional Regulation," *Journal of Computational Biology*, vol. 21, no. 5, pp. 373–384, May 2014.
- [8] DR. Graham, "Incremental development: Review of nonmonolithic life-cycle development models," *Information and Software Technology*, vol. 31, no. 1, pp. 7–20, Jan. 1989.
- [9] "urllib.request - Extensible library for opening URLs." [Online]. Available: <https://docs.python.org/3/library/urllib.request.html>
- [10] E. A. S. Jr, *SPA Design and Architecture: Understanding Single-Page Web Applications*. Simon and Schuster, Nov. 2015.
- [11] P. D'haeseleer, "What are DNA sequence motifs?" *Nature Biotechnology*, vol. 24, no. 4, pp. 423–425, Apr. 2006.
- [12] I. Erill, "A gentle introduction to... information content in transcription factor binding sites," Apr. 2012. [Online]. Available: <https://erilllab.umbc.edu/wp-content/uploads/sites/41/2016/04/Introduction.Information.Theory.pdf>
- [13] T. S. Furey, "ChIP-seq and beyond: New and improved methodologies to detect and characterize protein–DNA interactions," *Nature Reviews Genetics*, vol. 13, no. 12, pp. 840–852, Dec. 2012.
- [14] M. Djordjevic, "SELEX experiments: New prospects, applications and data analysis in inferring regulatory pathways," *Biomolecular Engineering*, vol. 24, no. 2, pp. 179–189, Jun. 2007.
- [15] Y. Zhang, M. Liu, H. Wang, Y. Ma, G. Huang, and X. Liu, "Research on WebAssembly Runtimes: A Survey," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 8, pp. 239:1–239:47, Oct. 2025.
- [16] "Pyodide.http — Version 0.29.3." [Online]. Available: <https://pyodide.org/en/stable/usage/api/python-api/http.html>
- [17] J. Verdú and A. Pajuelo, "Performance Scalability Analysis of JavaScript Applications with Web Workers," vol. 15, no. 2, pp. 105–108. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/7307120>
- [18] "Vue.js." [Online]. Available: <https://vuejs.org/>
- [19] E. Sayers, "A General Introduction to the E-utilities," in *Entrez® Programming Utilities Help [Internet]*. National Center for Biotechnology Information (US), Nov. 2022.
- [20] "National Center for Biotechnology Information." [Online]. Available: <https://www.ncbi.nlm.nih.gov/>
- [21] "Fetch API - Web APIs | MDN," Jun. 2026. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- [22] "XMLHttpRequest - Web APIs | MDN," May 2026. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>
- [23] "JavaScript execution model - JavaScript | MDN," Apr. 2026. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Execution_model
- [24] "Xml.etree.ElementTree — The ElementTree XML API." [Online]. Available: <https://docs.python.org/3/library/xml.etree.elementtree.html>
- [25] H. Chatham, "JavaScript Promise Integration in Pyodide," Jun. 2025. [Online]. Available: <https://blog.pyodide.org/posts/jsapi/>
- [26] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, B. A. Rapp, and D. L. Wheeler, "GenBank," vol. 28, no. 1, pp. 15–18. [Online]. Available: <https://doi.org/10.1093/nar/28.1.15>
- [27] "Sequence motif analysis using Bio.motifs — Biopython 1.87 documentation." [Online]. Available: https://biopython.org/docs/latest/Tutorial/chapter_motifs.html
- [28] E. Benítez-Bellón, G. Moreno-Hagelsieb, and J. Collado-Vides, "Evaluation of thresholds for the detection of binding sites for regulatory proteins in Escherichia coli K12 DNA," *Genome Biology*, vol. 3, no. 3, pp. research0013.1–research0013.16, 2002.
- [29] S. Kiliç, E. R. White, D. M. Sagitova, J. P. Cornish, and I. Erill, "CollecTF: A database of experimentally validated transcription factor-binding sites in Bacteria," *Nucleic Acids Research*, vol. 42, no. Database issue, pp. D156–160, Jan. 2014.

A APÈNDIX

A.1 Planificació i comparatives

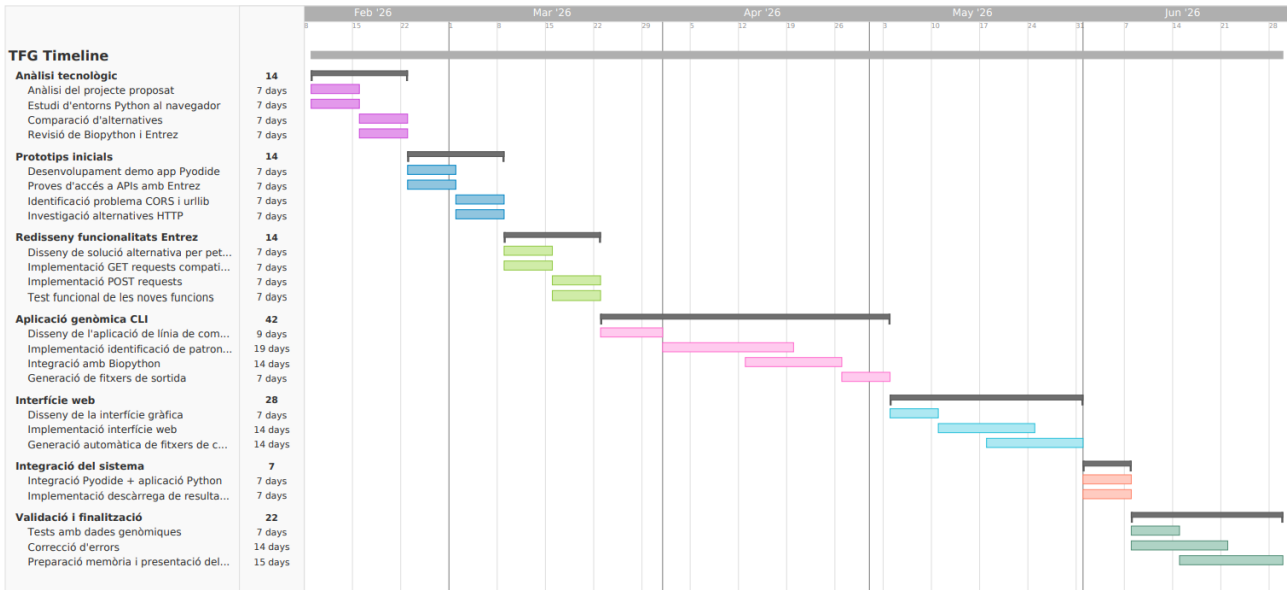


Fig. A.9: Planificació del projecte

TAULA 1: COMPARATIVA DE SOLUCIONS PER A PETICIONS HTTP EN PYODIDE

Solució	GET	POST	Compatible WebAssembly	API síncrona	Resultat
open_url	Si	No	Si	Si	Limitada
pyfetch	Si	Si	Si	No	Canvis significatius
Wrapper asíncron	Si	Si	Conflictiu	No	No és viable
run_sync	Si	Si	Si	Si	Solució adoptada

TAULA 2: DEPENDÈNCIA ENTRE PARÀMETRES DEL PIPELINE I ETAPES DE RE-EXECUCIÓ

Paràmetre modificat	Etapa a recalcular
Fitxer o accesió del genome	Tot el pipeline
Motiu, pseudocount, background	Construcció del motiu + escaneig + anotació
Threshold, integration log	Escaneig + anotació
Marges, operons, double report	Només anotació

A.2 Diagrames del pipeline

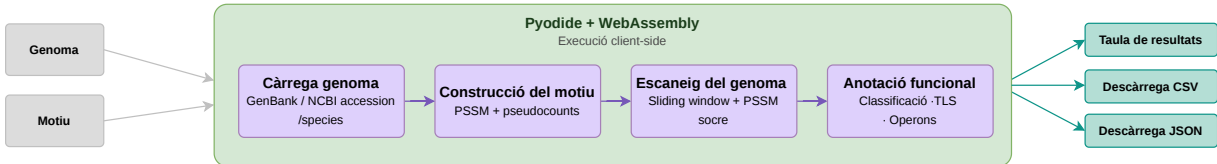


Fig. A.10: Pipeline d'anàlisi genòmica.

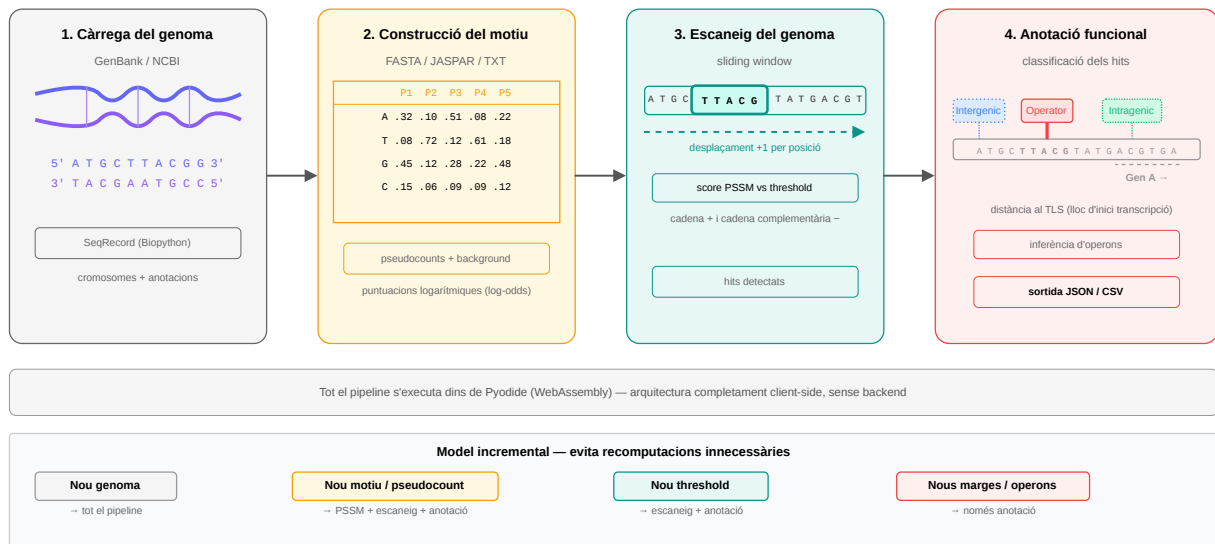


Fig. A.11: Pipeline d'anàlisi genòmica executat íntegrament al navegador mitjançant Pyodide.

A.3 Captures de la interfície web

Genome

Upload file Accession Species

Click to upload genome files
.gb · .gbk · .genbank · .gbff

Upload file Accession Species

SPECIES NAME
Escherichia coli Search

Searches NCBI reference genomes (RefSeq)

Upload file Accession Species

NCBI ACCESSION NUMBER
NC_000913.3 Load

Multiple accessions separated by commas

(a) Tipus d'entrada per carregar el genoma.

Motif

Upload file Paste sequences

Click to upload a motif file
.fasta · .fa · .txt · .jaspar

Upload file Paste sequences

SEQUENCES (ONE PER LINE)

ACGTACGT
TTGTACGT
ACGTACGT

Load sequences

(b) Tipus d'entrada per carregar el motiu.

Fig. A.12: Tipus d'entrada disponibles per al genoma i el motiu.

Pipeline Parameters direct · ±250/50 bp

SCORING

PSEUDOCOUNT 0.01

THRESHOLD METHOD Direct (raw score)

THRESHOLD VALUE 1

STRAND SCORE INTEGRATION Max Log-sum

REGULATORY WINDOW

REGION AROUND TLS

5' UPSTREAM TSS 3' DOWNSTREAM
-250 bp +50 bp

ANNOTATION

OPERON INFERENCE Enabled

MAX OPERON DISTANCE 100 bp

GENE ATTRIBUTION All associations

BACKGROUND MODEL Uniform Custom

Fig. A.13: Pantalla de configuració dels paràmetres de l'anàlisi.

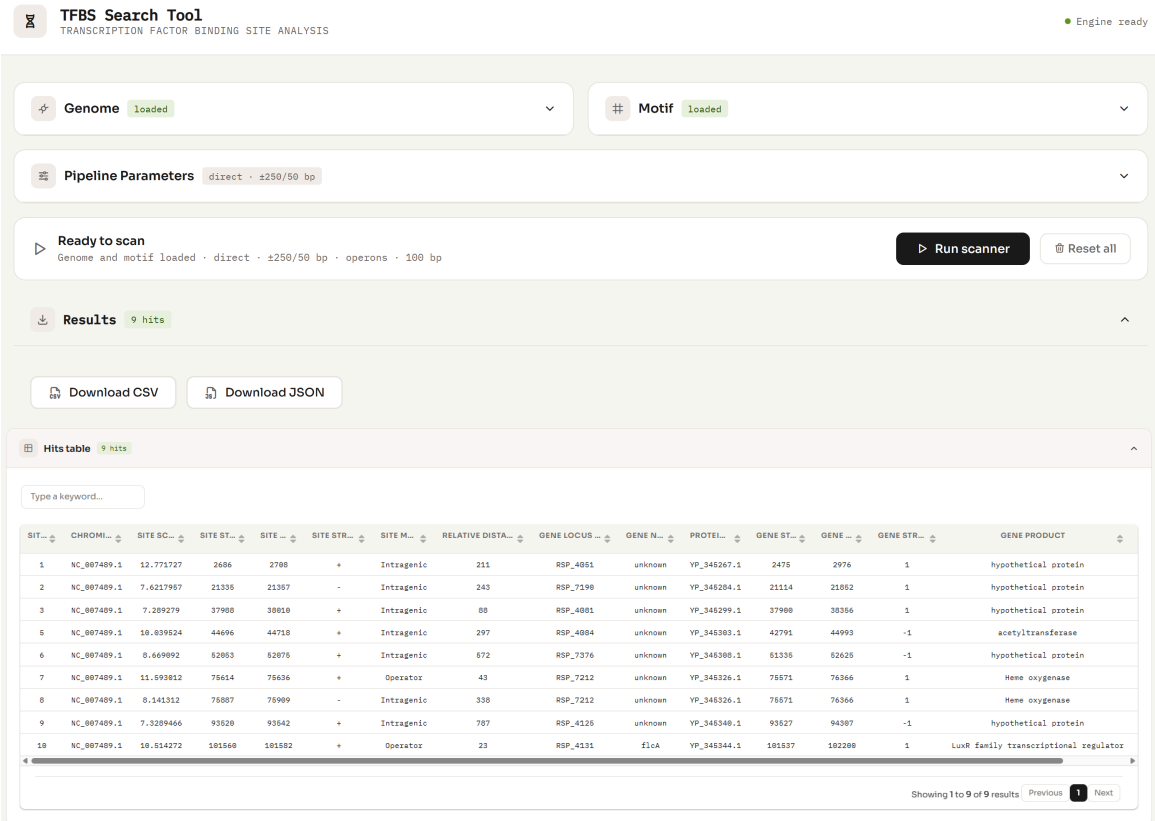


Fig. A.14: Visualització dels resultats generats pel pipeline.