Sæbio: automatización de un pipeline de aplicaciones para la secuenciación del genoma

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Santiago Martínez Willi

TUTORES:
José Juan Hernández Cabrera
José Évora Gómez

Julio 2021

Agradecimientos

A mi familia, pareja y amigos por apoyarme sin falta.

A José Juan por darme la oportunidad de hacer un TFG con el que poder seguir ampliando mis conocimientos.

A Raúl, del SIANI, por su inestimable ayuda.

A todo el personal sanitario por su encomiable labor.

Resumen

El uso de software de difícil instalación y manejo junto al desorden e ineficiencia en la gestión de datos es causa de que el Hospital Universitario de Gran Canaria Doctor Negrín tenga necesidades bioinformáticas insatisfechas para cumplir con el Plan Nacional de Resistencia a los Antibióticos (PRAN). A raíz de esto surge este proyecto con el objetivo principal de automatizar el proceso de la secuenciación del genoma y brindarlo como un servicio para la generación de informes de vigilancia epidemiológica.

Para el estudio del proceso mencionado se realizan charlas con personal del hospital y se estudian las herramientas bioinformáticas involucradas. El servicio se implementa como un servicio web que sigue la arquitectura REST (Representational State Transfer). El *backend* de este servicio se desarrolla en Java con el micro-framework Spark. Por otro lado, para su *frontend* se hace uso de JavaScript con el framework progresivo Vue.

Se ha logrado desarrollar un servicio web que ofrece al personal del hospital una plataforma de almacenamiento centralizada de los recursos necesarios para el análisis de la secuenciación del genoma de una bacteria, a la vez que proporciona la funcionalidad de solicitar dicho análisis y almacenar su resultado tras su finalización. Este servicio dispone de una sencilla interfaz de usuario que facilita el uso del servicio web, abstrayendo al usuario del uso de línea de comandos.

La dificultad elevada de instalación y uso del software bioinformático, ligada a la problemática que supone un gran volumen de datos cada vez mayor y la falta de integración entre plataformas de datos y dicho

software, deriva en un coste sanitario muy elevado, puesto que el análisis, almacenamiento y manipulación de los datos se acaba haciendo de forma manual. Esta ineficiencia se puede solventar con una integración eficaz entre el software bioinformático y una plataforma de datos a la vez que se facilita y automatiza el uso de dicho software, con lo que se lograría reducir el coste sanitario y aportar un gran valor.

Abstract

The use of software that is difficult to install and manage, together with the disorder and inefficiency in data management, is the reason why the Hospital Universitario de Gran Canaria Doctor Negrín has unmet bioinformatics needs to comply with the National Antibiotic Resistance Plan (PRAN). As a result, this project arises with the main objective of automating the genome sequencing process and providing it as a service for the generation of epidemiological surveillance reports.

For the study of the mentioned process, discussions are held with hospital personnel and the bioinformatics tools involved are studied. The service is implemented as a web service that follows the REST (Representational State Transfer) architecture. The *backend* of this service is developed in Java with the Spark micro-framework. On the other hand, the *frontend* is developed in Javascript with the progressive framework Vue.

We have managed to develop a web service that offers hospital staff a centralized storage platform for the resources needed for the analysis of the genome sequencing of a bacterium, while providing the functionality to request the analysis and store the result after its completion. This service has a simple user interface that facilitates the use of the web service, abstracting the user from the use of the command line.

The high difficulty of installation and use of bioinformatics software, linked to the problem of an ever-increasing volume of data and the lack of integration between data platforms and such software, results in a very high healthcare cost, since the analysis, storage and manipulation

of the data ends up being done manually. This inefficiency can be solved with an efficient integration between bioinformatics software and a data platform while facilitating and automating the use of such software, thus reducing healthcare costs and providing great value.

Índice general

Índice de figuras

Índice de cuadros

1 Introducción

Las técnicas de secuenciación de ADN son de suma importancia en el campo de la biología. Cuando se sospecha de un brote, los servicios de microbiología reciben muestras de diferentes localizaciones y/o pacientes para identificar el microorganismo causante del mismo. A estos aislados se les secuencia el genoma completo (*Whole Genome Sequenced*, *WGS*) y luego, mediante diversas herramientas y técnicas bioinformáticas, se analizan los resultados obtenidos de aplicar WGS.[2] El desarrollo y avance de estas técnicas y herramientas ha dado lugar a un problema: la generación de conjuntos de datos de gran tamaño y de complejo estudio y manipulación. Además, estas herramientas, por lo general, presentan grandes dificultades de uso para sus usuarios al requerirles un conocimiento informático bastante más elevado del que se presupone de un microbiólogo.

La colaboración entre la Universidad de Las Palmas de Gran Canaria y el Hospital Universitario de Gran Canaria Doctor Negrín surge a partir de las dificultades con la instalación y uso de la herramienta bioinformática *Nullarbor*, un *pipeline* cuya finalidad es la generación de informes de microbiología completos a partir de muestras WGS aisladas.[2] A raíz de esto nace este proyecto con el objetivo principal de automatizar el proceso de la secuenciación del genoma y brindarlo como un servicio para la generación de informes de vigilancia epidemiológica.

Este trabajo de fin de título continúa la línea de trabajo iniciada por el Instituto Universitario SIANI con el trabajo de fin de título de Omar Verona Kämpfer.[1]

Se llevan a cabo reuniones con el cliente, el personal del laboratorio de microbiología del hospital, y ante los requisitos fijados se realiza un servicio web que se conforma de una base de datos documental integrada con dos *APIs RESTful*. A lo largo del proyecto se sigue un estricto desarrollo guiado por tests y comportamiento. Este servicio

web cuenta, también, con una interfaz gráfica de sencillo uso en forma de *Single Page Application*.

Finalmente, el resultado es una integración eficaz entre una plataforma de datos sanitarios y la herramienta bioinformática Nullarbor, automatizándose el proceso de secuenciación del genoma y abstrayendo al usuario del uso de dicha herramienta. De esta forma, se le brinda al cliente lo que había solicitado: un servicio para la generación de informes de vigilancia epidemiológica.

2 Antecedentes

2.1. Problemática

El proyecto nace a raíz de las necesidades planteadas por el Hospital Universitario de Gran Canaria Doctor Negrín para cumplir con el Plan Nacional de Resistencia a los Antibióticos (PRAN). El PRAN plantea la creación de una red de laboratorios para la vigilancia de los microorganismos resistentes. Esta red requiere que se puedan compartir los datos de secuenciación genética de los microorganismos responsables de los principales problemas de resistencia.

Los servicios de microbiología, cuando se sospecha de un brote, reciben muestras de diferentes localizaciones y/o pacientes para identificar el microorganismo causante del mismo. A estos aislados se les secuencia el genoma completo (*Whole Genome Sequenced*, *WGS*) y luego, mediante diversas herramientas y técnicas bioinformáticas, se analizan los resultados obtenidos de aplicar WGS.[2] La secuenciación genética incluye el uso de varias herramientas bioinformáticas:

- **Toma de muestras** con un aparato secuenciador.

- **Trimming de las muestras** para eliminar adaptadores genéticos, secuencias identificadoras añadidas a los fragmentos de ADN, usando *Trimmomatic*.

- **Ensamblaje de las muestras** (de *novo assembly*) mediante el uso de un assembler (*spades*, *megahit*, etc.) incluido en *Nullarbor* (*pipeline* para la generación de

informes microbiológicos completos).

- **Anotación del genoma** (identificación de puntos de interés en las secuencias de ADN) mediante el uso de *Prokka*, herramienta incluida en Nullarbor.

- **Identificación de la muestra** (*MLST*, *Multilocus Sequence Typing*) mediante una comparación exhaustiva con secuencias anotadas de bases de datos de referencia (*Genbank*, etc.).

- **Generación de un árbol filogenético** para visualizar la distancia genética entre los distintos individuos secuenciados (Nullarbor).

- **Estudio del resistoma o viruloma** mediante *Abricate*, herramienta incluida en Nullarbor.

- **Estudio de los snips** (polimorfismos de un solo nucleótido, SNP) mediante el uso de *Snippy*, incluido en Nullarbor.

Para la realización de estas tareas, el personal de hospital utiliza Nullarbor, un pipeline de generación de informes microbiológicos a partir de aislados secuenciados. El objetivo final de Nullarbor es aplicar las técnicas de secuenciación del genoma para mediante la caracterización de un microorganismo determinar si tiene relación clonal o epidemiológica con otro. Sin embargo, el uso de Nullarbor requiere un nivel de conocimientos informáticos bastante más elevado del que cabe esperarse de un microbiólogo, ya que presenta grandes dificultades no solo en su manejo mediante línea de comandos, sino también en su instalación, disponible únicamente en plataformas UNIX y con un gran número de dependencias dispersas entre múltiples repositorios.

El uso de Nullabor implica, además, otro problema, el cual es la gestión tanto de los datos de las muestras obtenidas de WGS como de los datos resultantes de Nullarbor. Como no existe ninguna plataforma que realice todas las tareas del pipeline completo, las muestras de WGS son llevadas de manera manual mediante pendrives a la máquina donde está instalado Nullarbor, se ejecuta este programa, cuyo proceso puede tardar

muchas horas, y se espera a sus resultados sin ser notificados una vez acabe. Finalmente, los datos generados por Nullarbor se toman manualmente.

Todo lo anteriormente dicho concluye en que el personal del hospital se encuentra con demasiadas complicaciones: uso de herramientas que requieren conocimientos externos al ámbito del usuario, intervención excesiva por parte del usuario a lo largo del pipeline, transferencia de datos manual mediante pendrives, gestión por parte de los usuarios de dónde se almacenan estos datos, largas esperas por el resultado del proceso sin ser notificado de su finalización, etc. Esta falta de automatización y, por tanto, consecuente excesiva interacción del usuario en el pipeline causa ineficiencia en el proceso y puede dar lugar a errores por parte del usuario y pérdida de datos.

2.2. Objetivos

Este trabajo de fin de título continúa la línea de trabajo iniciada por el Instituto Universitario SIANI con el trabajo de fin de título de Omar Verona Kämpfer.[1] El principal objetivo de este proyecto es satisfacer las demandas del cliente, el laboratorio de microbiología del Hospital Universitario de Gran Canaria Doctor Negrín, siendo su principal necesidad la automatización del proceso de la secuenciación del genoma y brindarlo como un servicio para la generación de informes de vigilancia epidemiológica. Este servicio debe abstraer al usuario de la necesidad de instalación del pipeline Nullarbor y del uso de sus diversas herramientas. También tiene como objetivo que el usuario se desentienda de la transferencia de datos y de la gestión de estos, siendo el servicio un ingestor de datos de secuenciación genética.

Por otra parte, se busca que este servicio sea escalable y accesible a usuarios a través de la red como un *web service*. Este web service, asimismo, debe permitir que, cuando el usuario solicite el análisis del genoma de una bacteria, este pueda suscribirse al evento para que pueda obtener información sobre este proceso en cualquier momento.

Adicionalmente, el servicio web puede disponer de una interfaz gráfica de usuario de fácil uso.

Las necesidades que intenta cubrir este proyecto no son exclusivas del Hospital Universitario de Gran Canaria Doctor Negrín. Actualmente, todos los laboratorios de sanidad pública reciben cada vez más y más cantidades de datos con valor. Es por ello por lo que es imperativo que la gestión de los datos sea eficaz, eficiente y capaz de adaptarse a los tiempos modernos, como la actual pandemia de COVID-19.

Este proyecto, además de todo lo descrito, también busca brindar una solución inmediata para la generación de informes de vigilancia epidemiológica del laboratorio de microbiología.

3 Estado del arte

Las técnicas de secuenciación de ADN son de suma importancia en el campo de la biología. El desarrollo y progreso de estas técnicas ha dado lugar a un problema, el cual es la generación de conjuntos de datos de gran tamaño y de complejo estudio y manipulación. La creación de nuevas herramientas bioinformáticas dedicadas al análisis y estudio de la secuenciación de ADN más potentes, sumada a la creciente sensorización de los dispositivos, contribuye a este problema.[1]

Tecnologías como *high-throughput sequencing* (HTS), también conocida como *next generation sequencing* (NGS) o WGS, están afectando a la forma en la que se analiza el ADN bacteriano, reemplazando a métodos moleculares para la detección microbiana. Estas nuevas técnicas no generan una sola secuencia del genoma completo, sino que genera múltiples fragmentos de secuencias (contigs), los cuales requieren, además, de un cribado de fragmentos defectuosos, ensamblaje, etc. y conforman un conjunto de datos de gran volumen.[1]

La bioinformática es una disciplina emergente y relativamente joven de investigación que busca abordar esta clase de problemas. Algunas definiciones de bioinformática que encontramos en la literatura son:

"La bioinformática es un área de investigación en la que informáticos, biólogos, físicos, matemáticos, y químicos combinan su habilidad para colaborar en diversas tareas, desde el descubrimiento de nuevos hechos en sistemas biológicos complejos hasta la racionalización de la organización de los sistemas de salud."[3]

"La bioinformática está conceptualizando la biología en términos de macromoléculas

(entendidas desde el punto de vista físico-químico) y aplicando técnicas informáticas (derivadas de disciplinas como las matemáticas, las ciencias de la computación, y la estadística) para entender y organizar la información asociada a estas moléculas en una gran escala."[4]

"La bioinformática es un área de investigación interdisciplinar que aplica metodologías de las ciencias de computación, las matemáticas aplicadas y la estadística al estudio de fenómenos biológicos."[5]

Actualmente existen potentes herramientas bioinformáticas que dan solución a las dificultades del WGS. Estas herramientas se enfocan en distintas partes del pipeline o bien en varias, ya sea para el ensamblado de los contigs, la criba de sus fragmentos defectuosos, para realizar diversos análisis sobre los datos del genoma como distancias filogenéticas entre microorganismos o predicciones de resistencia a antibióticos, etc.[1]

La tabla 3.1 da una visión general de las distintas aplicaciones de las que disponemos hoy día para la secuenciación genómica.[1]

Uso	Software	Descripción	URL
Medidas de calidad y preprocesado en la lectura	FASTQC	Herramientas para mostrar estadísticas de secuenciado y para lecturas NGS	`http://www.bioinformatics.babraham.ac.uk/projects/fastqc`
	TRIMMOMATIC	Aplicación de línea de comandos para el recorte de aquellos datos de lecturas cortas con terminación individual o en pares	`http://www.usadellab.org/cms/?page=trimmomatic`
	FASTX-Toolkit	Una colección de aplicaciones de línea de comandos para el preprocesado de archivos FASTA/FASTQ de lecturas cortas	`http://www.bioinformatics.babraham.ac.uk/projects/fastqc`

	PRINSEQ	Aplicación tanto web como de línea de comandos para el filtrado, reformateo o recorte de datos de secuenciación genómica y metagenómica, es capaz de generar resúmenes estadísticos en formato gráfico o tabular NGS	`http://www.bioinformatics.babraham.ac.uk/projects/fastqc`
Detección de contaminación	Kraken	Clasificación taxonómica de las lecturas, es útil para análisis metagenómicos o para la detección de contaminación en muestras puras de cultivos	`https://ccb.jhu.edu/software/kraken/`
	MIDAS	Clasificación taxonómica de las lecturas, es útil para análisis me- tagenómicos o para la detección de contaminación en muestras puras de cultivos	`https://github.com/snayfach/MIDAS`
Software y *pipelines* de ensamblado	Velvet	Ensamblador *de novo* diseñado para lecturas cortas	`http://github.com/dzerbino/velvet/tree/master`

	SPAdes	Ensamblador *de novo* diseñado para lecturas cortas; también da la posibilidad de ensamblajes híbridos entre lecturas cortas y largas	`http://cab.spbu.ru/software/spades/`
	Canu	Ensamblador *de novo* diseñado para aquellas moléculas indivi- duales con alto ruido como son las lecturas largas	`http://github.com/marbl/canu`
	INNUca	Un *pipeline* estandarizado, completamente automatizado, flexible, portable e independiente del patógeno para el ensamblaje de genoma bacteriano y control de calidad en lecturas cortas	`http://github.com/INNUENDOCON/INNUca`
	shovill	Un *pipeline* para el ensamblaje de genoma bacteriano que mejora la velocidad y precisión de SPAdes	`https://github.com/tseemann/shovill`

Tipado *in silico*	Microbial InSilico Typer (MIST)	Generación rápida *in silico* de datos de tipado (e.g. MLST, MLVA) a partir de borradores de ensamblaje del genoma bacteriano	`http://bitbucket.org/peterk87/microbialinsilicotyper`
	ResFinder	Una herramienta web para la detección de genes de resistencia antimicrobial adquiridos en genoma bacteriano usando lecturas en bruto o borradores de ensamblaje de genoma	`https://cge.cbs.dtu.dk/services/ResFinder/`
	MLST1.8	Una herramienta web para la determinación de tipos MLST a partir del genoma bacteriano usando esquemas MLST públicos	`https://cge.cbs.dtu.dk/services/MLST`
	Mlst2.9	Una aplicación de linea de coman- dos que es capaz de extraer el MLST a partir de genomas bacterianos usando esquemas MLST públicos	`https://github.com/tseemann/mlstCFSANSNP`

	Snippy	Un *pipeline* para la rápida identificación de variantes haploides y la construcción de filogenía usando single nucleotid polymorphism (SNPs) del genoma central	`http://github.com/tseemann/snippy`
Aproximaciones gen a gen	BGSdb	Una base de datos accesible por web diseñada para guardar y analizar información fenotípica y genotípica relacionada, incluyendo el motor de llamada de alelos para una aproximación gen a gen; es la base de datos tanto de PubMLST como de PasteurMLST	`https://github.com/kjolley/BIGSdb`
	Enterobase	Base de datos curada y recurso online para el tipado molecular de *Salmonella*, *Escherichia coli*, *Yersinia spp.* y *Moraxella spp.* usando un acercamiento gen a gen	`http://enterobase.warwick.ac.uk/`

	Genome Profiler	Algoritmo independiente de llamada a alelos gen a gen que usa la vecindad genética conservada para resolver paralogías genéticas	http://sourceforge.net/projects/genomeprofiler/
	chewBBACA	Algoritmo exhaustivo independiente de alta eficacia de llamadas a alelos gen a gen basado en la codificación de secuencias de ADN, incluye un conjunto de herramientas para dar una vista general del rendimiento de los esquemas	https://github.com/B-UMMI/chewBBACA
Anotación genética	Prokka	Anotación funcional rapida de genomas bacterianos produciendo archivos de salida que cumplen los estándares definidos	http://github.com/tseemann/prokka
	RAST	Servicio completamente automatizado para la anotación de genomas bacterianos y de árqueas	http://rast.nmpdr.org/

	MicroScope	Plataforma de análisis exhaustivo para la anotación genética y el análisis de genomas bacterianos	http://www.genoscope.cns.fr/agc/microscope/home/index.php
	NCBI Pathogen Detection	Una plataforma online para compartir y comparar datos de cepas epidémicas/de brotes; actualmente contiene bases de datos para 20 tipos de especies bacterianas, centrándose en patógenos ligados a los alimentos y a infecciones ligadas a la atención sanitaria	https://www.ncbi.nlm.nih.gov/pathogens/
Alineamiento del genoma	Harvest	Un conjunto de herramientas para el alineamiento y visualización del genoma-núcleo para un analisis rápido y de alto rendimiento de genomas bacterianos intraespecíficos	http://harvest.readthedocs.io/en/latest/

	Mauve	Alineador para un analisis comparativo de genomas bacterianos completos	`http://darlinglab.org/mauve/mauve.html`
Agrupamiento por homología y estudios de asociación	Roary	*Pipeline* de pangenomas independiente de alta velocidad para genomas bacterianos	`http://sanger-pathogens.github.io/Roary/`
	Scoary	Estudios de asociación a nivel de pangenoma usando la salida del Roary	`https://github.com/AdmiralenOla/Scoary`
	Neptune	Software diseñado para detectar firmas genomicas dentro de poblaciones bacterianas	`https://github.com/phac-nml/neptune`
Inferencia filogenética	RAxML	Estimación de la máxima verosimilitud de la filogenia tanto paralela como secuencial que trabaja con los alineamientos de secuencias de nucleótidos y proteínas	`https://sco.h-its.org/exelixis/software.html`

	FastTree	Computa áboles filogenéticos de máxima verosimilitud a partir de alineamientos de múltiples secuencias en grandes nucleótidos o proteínas	http://www.microbesonline.org/fasttree/
	Gubbins	Computa la máxima verosimilitud a partir de alineamientos tras eliminar aquellas regiones que contienen densidades elevadas de sustituciones de bases	https://github.com/sangerpathogens/gubbins
	PHYLOViZ 2.0	Software independiente desarrollado en Java para inferencia filogenética, visualización y análisis de métodos de tipado que generan perfiles alélicos y los datos epidemiológicos asociados a partir de secuencias	http://www.phyloviz.net/
Herramientas de visualización de datos	Microreact	Una aplicación web para la visualización e intercambio de datos epidemiológicos genómicos	http://microreact.org/

	Phandango	Aplicación web para una exploración rápida de datos genómicos poblacionales de gran escala combinando los datos de salida de diferentes métodos de análisis genómicos	`https://github.com/jameshadfield/phandango`
	iTOL	CAplicación web para mostar, anotar y gestionar árboles filogenéticos	`http://itol.embl.de/`
	GenGIS 2	Aplicación con gráficos 3-D e interfaces en Python que permiten a los usuarios combinar los datos de mapas digitales y las secuencias	`http://kiwi.cs.dal.ca/GenGIS/Main_Page`
Plataformas y *pipelines* analíticos multipropósito	Centre for Genomic Epidemiology Toolbox	Un conjunto de herramientas web y servicios para el tipado molecular de patógenos, ensamblado genómico, predicción fenotípica (e.g. preddición de la resistencia) y construcción de la filogenia	`http://cge.cbs.dtu.dk/services/`

Integrated Rapid Infectious Disease Analysis (IRIDA) platform	Una plataforma basada en Galaxy para la investigación en tiempo real de brotes infecciosos usando datos genómicos, incluye un módulo para la gestión de datos de secuencia y flujos de trabajo, un *framework* para ontologías (GenEpiO) y herramiemientas de visualización de datos	`https://irida.corefacility.ca/documentation/downloads/index.html`
Integration genomics in surveillance of foodborne pathogens (INNUENDO) platform	Una plataforma para a investigación en tiempo real de brotes infecciosos y la vigilancia de patógenos ligados a alimentos usando datos genómicos, incluye módulos para la gestión de datos de secuencia, ensamblado con medidas QA/QC, un *pipeline* de análisis gen a gen, *framework* para ontologías y herramientas de visualización	`https://github.com/INNUENDOCON/INNUENDO_platform`

	Nullarbor	Un *pipeline* para la generación de informes microbiológicos de salud pública a partir de aislados secuenciados, incluye datos específicos de secuenciación, identificador de especies, subtipos y estudio de los SNPs	`http://github.com/tseemann/nullarbor`

Cuadro 3.1: Ejemplos de software actual para la secuenciación del genoma (Verona Kämpfer et al. 2020, Tabla 3.1)[1]

Como se puede apreciar, existen varias herramientas bioinformáticas capaces de satisfacer varias de las necesidades actuales del análisis y vigilancia epidemiológica. Sin embargo, presentan problemas:

- Muchas herramientas requieren de unos conocimientos informáticos demasiado elevados, por lo que presentan una barrera de entrada muy grande para nuevos usuarios.[1]

- El uso de muchas de ellas no es práctico, involucrando excesivamente al usuario en el proceso.

- Las plataformas de datos de las instituciones sanitarias suelen ser bastante inflexibles y longevas y las herramientas bioinformáticas no presentan facilidades para la integración con ellas, por lo que se hace clara la necesidad de una integración entre herramientas bioinformáticas y plataformas de datos sanitarios.

4 Resultados

Respecto a los objetivos propuestos del trabajo de fin de título, se ha logrado cumplir con todos. Al cliente, el laboratorio de microbiología del Hospital Universitario de Gran Canaria Doctor Negrín, se le ha dado una herramienta bioinformática que automatiza el proceso de secuenciación del genoma de una bacteria y facilita la generación de informes de vigilancia epidemiológica. La aplicación, por consiguiente, hace que el usuario prescinda de la necesidad de instalar Nullarbor y de todas las complicaciones que conlleva su instalación. Asimismo, integra el proceso de secuenciación del genoma con un repositorio centralizado de datos sanitarios, logrando que el usuario se desentienda tanto de la transferencia como de la gestión de los datos.

La aplicación se brinda como un servicio web. Este servicio dispone de una interfaz gráfica de usuario de uso sencillo que logra que el personal del hospital se abstraiga del uso de línea de comandos. El *web service*, además, permite al usuario acceder a sus funcionalidades desde cualquier plataforma que disponga de un navegador web, si desea hacer uso de la interfaz gráfica, o de una herramienta de línea de comandos como *curl*.

Con este servicio web el personal del hospital es capaz de almacenar de manera centralizada secuencias *trimmed* (estando ya *trimmed* o solicitando el *trimming* en la subida), referencias y los ficheros resultantes del proceso de análisis de la secuenciación del genoma; entre ellos, el informe de vigilancia epidemiológica. Permite, también, la solicitud de análisis de la secuenciación del genoma de múltiples aislados para la generación del informe.

La herramienta que se brinda al cliente es escalable sin depender de los recursos del usuario, ya que se ejecuta en un servidor HTTP y no en su máquina local. Adicionalmente, no solo consigue que el usuario se desentienda de la gestión física de los datos, sino que se le otorga posibilidades de las que antes no disponía, como filtrarlos por fecha

y/o por género y especie.

Por otro lado, la API de la que consume la interfaz gráfica se desarrolló siguiendo TDD y BDD, con lo que se intenta minimizar el número de errores de la aplicación. Siempre se notifica al usuario cuando se producen errores de cliente o errores del sistema, si estos llegasen a ocurrir. También se informa al usuario cuando un recurso se crea con éxito. Con todo esto se consigue que el usuario sea consciente en todo momento de lo ocurre y sepa que hay respuestas a sus acciones.

La interfaz gráfica, por otra parte, se implementó como una *Single Page Application* (*SPA*). Con esto se alcanza que la interacción del usuario con la aplicación sea fluida y lo más parecida posible a una aplicación de escritorio. Esta interfaz, además de ser sencilla de usar, embebe el informe de vigilancia epidemiológica por lo que para verlo no es necesario descargarlo, sino que tanto visualizarlo como compartirlo con el personal del hospital se realiza a través de una *URL* y no a través del traspaso y manejo manual de ficheros.

Asimismo, cabe destacar que el proyecto *Sæbio: Nullarbor API* se llevó a cabo de tal forma que, en un futuro, su contenerización junto a una instalación de Nullarbor sería categóricamente simple ya que se dejó preparada una arquitectura capaz de ser *Serverless*.

Los tres proyectos se encuentran alojados en *GitHub*:

- **Sæbio: genome sequencing API:** `https://github.com/santiagoWilli/saebio-genome-sequencing-api`

- **Sæbio: genome sequencing GUI:** `https://github.com/santiagoWilli/saebio-gs-gui`

- **Sæbio: Nullarbor API:** `https://github.com/santiagoWilli/saebio-nullarbor-api`

Finalmente, la herramienta bioinformática desarrollada en este trabajo de fin de grado es una aplicación completamente funcional y le da al personal del laboratorio de microbiología una solución para la generación de informes de vigilancia epidemiológica, el cual era el objetivo principal de este proyecto.

5 Metodología

5.1. Estudio previo

La colaboración entre la Universidad de Las Palmas de Gran Canaria y el Hospital Universitario de Gran Canaria Doctor Negrín surge a partir de las dificultades con la instalación y uso de la herramienta bioinformática Nullarbor. Nullarbor es un pipeline cuya finalidad es la generación de informes de microbiología completos a partir de muestras WGS aisladas.[2]

El estudio de las complicaciones derivadas del uso de Nullarbor se hizo en otro proyecto de fin de título llamado Arquitectura de datos para la caracterización y análisis de microorganismos multiresistentes.[1] A lo largo de dicho proyecto se logró instalar el pipeline en una máquina virtual de Linux y el control de la aplicación se abstrajo mediante una serie de *scripts* preconfigurados que le fueron entregados al hospital por otro laboratorio familiarizado con el uso de la herramienta. Mediante el estudio del proyecto arriba mencionado y de reuniones con personal del hospital quedan expuestas las necesidades y dificultades a las que se enfrenta el laboratorio de microbiología.

El laboratorio de microbiología decide utilizar la herramienta Nullarbor debido a la próxima adopción de técnicas de WGS. Estas técnicas generan *contigs*, segmentos de secuencias de ADN superpuestos que reconstruyen a la secuencia original de ADN cuando se realiza su ensamblado. Para realizar este ensamblado y el posterior análisis y estudio de la secuenciación del genoma, se requiere de varias herramientas bioinformáticas especializadas,[1] las cuales están agrupadas por Nullarbor.

Los contigs, por su naturaleza, hacen que el volumen bruto de datos sea muy grande. Además, el pipeline Nullarbor produce otro gran volumen de datos a lo largo de su *workflow*: datos de ensamblado, mapas genómicos anotados, datos de resistencia, distancias filogenéticas, estudio sobre el resistoma o viruloma, estudio sobre los snips, etc.[1] Esta gran cantidad de datos y la gestión manual de estos por parte de los usuarios lleva a la clara necesidad de una arquitectura capaz de dar solución a la gestión de datos y que facilite el uso de las herramientas bioinformáticas.

Se llevó a cabo varias reuniones con personal del hospital y el SIANI para aprender sobre las herramientas que engloba Nullarbor y el uso de este pipeline a través de los scripts *bash* preconfigurados. En estas reuniones quedó claro el workflow básico de Nullarbor a través de estos scripts:

- **trim**: realiza el trimming de los pares de secuencias en el directorio de trabajo actual para eliminar de los fragmentos los datos que no interesan para los pasos posteriores del workflow.

- **generatecsv**: genera el fichero nullarbor.csv, en el cual se asocia a cada aislado su par de secuencias.

- **run <archivo de referencia>**: ejecuta todos los pasos del workflow posteriores al trimming en el directorio de trabajo actual. Hace uso, además del fichero CSV, de un archivo de referencia que es una secuencia del genoma de un microorganismo ya ensamblada. A la hora de ejecutar este script, debemos asegurarnos de que todos los aislados y la referencia son del mismo género y especie y, además, disponer de un mínimo de cuatro aislados diferentes. El resultado final del pipeline es una gran cantidad de datos, entre los que se encuentra un informe HTML que recopila todos estos datos y los presenta de manera organizada y de fácil lectura para el usuario.

Una secuencia está formada por un par de ficheros. Los nombres de estos ficheros siguen la expresión regular **[a-zA-Z]+[0-9]{1,6}_R(1|2).(fq|fastq).gz**. La convención que se tomó con el laboratorio de microbiología para nombrar a los aislados se explica de la siguiente manera, separando el nombre en campos:

- **[a-zA-Z]+** hace referencia al código que identifica al género y especie del aislado. Por ejemplo, kpneu es el código que identifica al género *Klebsiella* y a la especie *pneumoniae*.

- **[0-9]{1,6}** indica el código interno que le asignó el hospital al aislado.

- **_R(1|2)** R1 o R2, precedido de una barrabaja, aclara qué par es.

- **.(fq|fastq).gz** es la extensión, la cual es FASTQ, un formato basado en texto para almacenar tanto una secuencia biológica como sus valores de calidad.[6]

Para el nombre de una referencia se sigue la misma convención en los dos primeros campos y no dispone del tercer campo ya que la forma solo un solo fichero. Su extensión es FASTA, un formato basado en texto para representar secuencias de ácidos nucleicos o de péptido,[7] y no presenta compresión.

A lo largo de estas reuniones, el cliente va elaborando progresivamente los requisitos que debe cumplir el servicio:

- Enviar el par de ficheros de una secuencia y que se almacene el par de ficheros resultante de hacer el trimming.

- Descargar el par de ficheros trimmeados de una secuencia comprimido en un archivo de formato ZIP.

- Añadir el conjunto de género y especie de un microorganismo y sus abreviaciones.

- Eliminar un conjunto de género y especie de un microorganismo si no hay ficheros que apunten a él.

- Modificar las abreviaciones de un conjunto de género y especie de un microorganismo.

- Subir el fichero de una referencia.

- Descargar el fichero de una referencia.

- Iniciar el proceso del pipeline a partir de las secuencias y referencias previamente almacenadas y guardar el informe y referencia resultantes una vez acabe el proceso.

- Descargar el informe de vigilancia epidemiológica resultante de una solicitud previa de realización del proceso del pipeline.

5.2. Desarrollo

5.2.1. Desarrollo del repositorio

Debido a la naturaleza documental del proceso de secuenciación del genoma, se decide implementar el repositorio como una base de datos documental; concretamente, con *MongoDB*. Este tipo de base de datos es uno de los numerosos tipos de bases de datos *NoSQL*. Específicamente, una base de datos documental presenta varias ventajas sobre una base de datos *SQL* para nuestro caso particular. Las bases de datos documentales son mucho más flexibles, ya que la estructura de sus documentos (filas SQL) no viene determinada por la estructura de la colección (tabla SQL), permitiendo almacenar grandes conjuntos de datos no estructurados e ir estructurándolos y sacar información de ellos a medida que se vaya desarrollando la aplicación. Esto, por ejemplo, permite añadir documentos con ciertos campos que no tienen por qué tenerlos los documentos ya existentes en la colección y cambiar, según la versión, la estructura de los documentos, pudiendo convivir documentos con distintas versiones en la misma colección. Otro

aspecto destacable de estas bases de datos es que los propios datos están distribuidos en pocas colecciones, con pocas relaciones, incrementando la eficiencia. Respecto a la escalabilidad, las bases de datos documentales facilitan la escalabilidad horizontal, al permitir distribuir las colecciones entre distintos servidores.[8]

MongoDB almacena los datos en formato *BSON* (*Binary JSON*). BSON añade campos opcionales no nativos de JSON para mejorar la eficiencia, pero es, desde un punto de vista de alta abstracción, idéntico a JSON.[9] Debido a esto, para la estructura de los datos se van decidiendo a lo largo del desarrollo de este proyecto, como primera versión, los esquemas BSON de la figura 5.1.

Posteriormente, si se desea modificar la estructura de los documentos, bastaría con añadir un campo que denote la versión del documento, teniendo como primera versión los que no tienen dicho campo. Finalmente, en la interfaz gráfica, para trabajar con documentos que tienen distintas estructuras entre ellos y que no se produzcan fallos, solo haría falta tratar los documentos según su versión.

5.2.2. Desarrollo del backend

El proceso de secuenciación del genoma de una bacteria se acordó brindarlo como un servicio web. Este servicio web automatiza el proceso y sirve para la generación de informes de vigilancia epidemiológica. Para su backend se decidió implementarlo como una API RESTful. Sin embargo, el desarrollo de esta API necesita del desarrollo de otra API que haga uso de Nullarbor, la herramienta bioinformática elegida por el laboratorio de microbiología, siendo un total de dos APIs las que se desarrollaron. La primera, la que consumirá el laboratorio de microbiología del hospital, se denominó *Sæbio: genome sequencing API* (SGSA) y la segunda, la cual será utilizada exclusivamente por la otra, es *Sæbio: Nullarbor API* (SNA).

```
Secuencia a la espera del resultado del trimming: {
    "strain": ObjectId,
    "code": String,
    "originalFilenames": [String],
    "genomeToolToken": String,
    "trimRequestDate": Date,
    "sequenceDate": Date
},
Secuencia tras resultado del trimming: {
    "strain": ObjectId,
    "code": String,
    "originalFilenames": [String],
    "genomeToolToken": String,
    "trimRequestDate": Date,
    "sequenceDate": Date,
    "trimmedPair": [ObjectId] or Boolean
},
Referencia: {
    "strain": ObjectId,
    "code": String,
    "file": ObjectId
},
Referencia (resultante de un analisis): {
    "strain": ObjectId,
    "reportName": String
    "file": ObjectId
},
Informe a la espera del resultado del pipeline: {
    "name": String,
    "strain": ObjectId,
    "sequences": [ObjectId],
    "reference": ObjectId,
    "genomeToolToken": String,
    "requestDate": Date
},
Informe tras el resultado del pipeline: {
    "name": String,
    "strain": ObjectId,
    "sequences": [ObjectId],
    "reference": ObjectId,
    "genomeToolToken": String,
    "requestDate": Date,
    "files": {
        "report": ObjectId, "reference": ObjectId, files...: ObjectId
    } or Boolean
},
Cepa: {
    "name": String,
    "keys": [String]
}
```

Figura 5.1: Esquemas BSON del repositorio

5.2.2.1. Sæbio: genome sequencing API

Este proyecto provee las funcionalidades necesarias al personal del laboratorio de microbiología para la generación de informes de vigilancia epidemiológica. Esta API consume a su vez de la API *Sæbio: Nullarbor API* y se integra con una base de datos MongoDB. Para el desarrollo de *Sæbio: genome sequencing API* (SGSA) se siguen los principios de una metodología ágil basada en iteraciones para entregar periódicamente valor al cliente. El plan inicial suponía unas iteraciones de dos semanas, con 30 horas por iteración; sin embargo, debido a diversas complicaciones las iteraciones se alargaron o acortaron, pero sin perder el valor entregado al cliente en cada una de ellas. Como herramienta de control de versiones, se hizo uso de Git. La gestión de ramas se hizo siguiendo la escencia del flujo de trabajo de Gitflow, desarrollándose las nuevas funcionalidades de la iteración en la rama *develop* y haciendo un *merge* con los cambios a *main* cuando la iteración estuviese acabada.[10]

Por otro lado, el desarrollo de este proyecto siguió la metodología *Test Driven Development* (TDD) en conjunto con *Behaviour Driven Development* (BDD). Inicialmente solo se previó seguir un desarrollo guiado por tests; sin embargo, TDD hace tests unitarios, o lo que es lo mismo, únicamente testea módulos aislados. Por otra parte, BDD hace tests de de aceptación; es decir, verifica el correcto funcionamiento de la aplicación según la interacción del usuario con ella, con los diferentes módulos interconectados. Como el objetivo principal de este proyecto de fin de título es entregarle valor al cliente, se decidió usar en conjunto TDD con BDD para asegurarnos, en la medida de lo posible, de que el servicio que recibe el cliente es un servicio sin fallos. Por tanto, el desarrollo se llevó a cabo siguiendo una metodología *outside-in testing*.[11] Esto es, se plantea un test de aceptación que falle que guía el desarrollo de la funcionalidad. Tras esto se entra en un bucle de desarrollo guiado por tests unitarios para los módulos implicados en la funcionalidad. Cuando el test de aceptación pase, la funciolidad estará implementada como se desea según el comportamiento descrito por el test. De esta manera, el test unitario (TDD) verifica el correcto funcionamiento de una unidad aislada y el test de aceptación (BDD) verifica que las diferentes unidades se integran de la manera esperada para lograr el comportamiento deseado.[12]

Para el desarrollo de ambas APIs se decide seguir la arquitectura REST, por lo cual hablamos de API *RESTful*. La Transferencia de Estado Representacional (REST) es un estilo de arquitectura software para sistemas hipermedia que hace uso del protocolo HTTP para el transporte y operaciones sobre los datos,[13] los cuales pueden estar en formato JSON, HTML, XLT, Python, PHP o texto sin formato. Una API que se

ajusta a la arquitectura REST es una API RESTful. Para esto, la API debe: tener una arquitectura cliente/servidor sin estado, cada operación HTTP contiene toda la información necesaria para comprender la petición y las operaciones son independientes; tener un conjunto de operaciones bien definidas; una interfaz uniforme en la que los recursos deben ser identificables mediante su URI, la información que se le provee al cliente debe ser la necesaria para manipular los recursos y los mensajes que se envíen al cliente deben tener información suficiente sobre el resultado de la operación.[14]

SGSA, al ser RESTful, se desarrolla haciendo uso del micro-framework *Spark*. Spark está pensado para implementar aplicaciones web con *Java* o *Kotlin* de manera sencilla y con el mínimo *boilerplate code*[15] (fragmentos de código estándar repetitivo que se usa en los proyectos), a diferencia de frameworks grandes como *Spring*. A raíz de esto, el lenguaje de programación con el que se implementa la API es Java. Java es un lenguaje de alto nivel de abstracción, seguro, eficiente y robusto, entre otras características. Esto hace que sea un lenguaje ideal para desarrollar una API RESTful.

Para la gestión de dependencias, la compilación, la gestión y construcción del proyecto se hace uso de *Maven*. Maven es una herramienta para la gestión y construcción de proyectos en Java y cuyos objetivos principales son facilitar un proceso de construcción sencillo y uniforme, proveer de información de calidad sobre el proyecto y alentar a realizar mejores prácticas de desarrollo.[16]

Por otra parte, se separó el código de la lógica de la aplicación del código del funcionamiento de Spark para que no haya acoplamiento. Al no haber acoplamiento, en un futuro, si fuera necesario, se podría cambiar el framework (Spark) por otro sin necesidad de modificar la lógica de la aplicación ni su código. Este desacople, además, permite hacer tests unitarios sobre los distintos módulos sin necesidad de incluir cógido de test para nada exterior a la lógica de la propia aplicación; en este caso, Spark.[17] Para los tests se hace uso de las librerías *JUnit 5* y *AssertJ*, una librería *fluent API* que provee de un gran conjunto de aserciones para mejorar la legibilidad del código de test.[18]

Respecto a los tests de aceptación, estos se implementaron en Java con ayuda de la librería *REST-assured*.[19] Esta librería está hecha para simplificar la implementación de tests en APIs RESTful. Para la realización de estos tests, se siguieron ciertas pautas descritas en un tutorial de *functional testing* de Spark,[20] como el desarrollo de *bash scripts* para el arranque y terminación del servicio y de la base de datos de manera automatizada. También se hace uso de la librería *WireMock*, un simulador de APIs

basadas en HTTP,[21] para emular el comportamiento de Sæbio: Nullarbor API.

La aplicación necesita que se le indique en el momento de ejecución ciertos parámetros para poder funcionar correctamente. Algunos de estos parámetros son opcionales ya que tienen valores por defecto. Estos parámetros se indican mediante línea de comandos como argumentos y bridan una forma sencilla de cambiar de repositorio y de herramienta de secuenciación del genoma (Sæbio: Nullarbor API). La siguiente lista indica cuáles son:

- -gt, --genome-tol <saebio-nullarbor-api URL>
- -p, --port <puerto de la aplicación> (4567 por defecto)
- -dbh, --db-host <dominio del repositorio, sin protocolo ni puerto>
- -dbp, --db-port <puerto del repositorio> (27017 por defecto)
- -db, --database <nombre de la BD a usar> (saebio por defecto)
- --help

El desarrollo del proyecto sigue los principios *SOLID*. SOLID es un acrónimo ideado por Robert C. Martin que representa cinco principios para la elaboración de programas legibles y mantenibles. Las ventajas de seguir los principios SOLID son un mantenimiento del código más fácil y rápido, la capacidad de poder añadir nuevas funcionalidades de forma sencilla y un código de mayor calidad y reusabilidad.[22] La lógica de la aplicación, como se comentó antes, está separada de la lógica de Spark. La clase principal de la aplicación, *Application*, relaciona cada ruta sobre la que se puede operar de la API con una instancia de distinta clase para cada ruta. Estas clases tienen un único propósito bien definido de acuerdo con el principio de responsabilidad única. Además, una instancia de estas clases en caso de requerir acceder a la base de datos o hacer uso de la herramienta de secuenciación del genoma lo hará a través de instancias abstractas

inyectadas en su creación y no será ella misma la que se encargue de realizar nuevas instancias, dependiendo, así, de abstracciones y siguiendo el principio de inversión de dependencias. Estas clases, a su vez, implementan diversas interfaces, con propósito concreto y no general, según la necesidad, cumpliendo con el principio de segregación de interfaz. Con todo lo anterior, también se sigue el principio de sustitución de Liskov y el principio de abierto/cerrado.

El proyecto tiene una arquitectura *publisher/subscriber* en cierta medida, sin canales de mensajería asíncronos. Cuando se solicita un análisis del genoma de varias secuencias o se solicita el trimming de una secuencia, se genera un recurso con un *token* único. Este recurso está suscrito a la herramienta de secuenciación del genoma mediante dicho token; de manera que, cuando la herramienta finaliza la operación solicitada, esta notifica al recurso suscrito del resultado.

La aplicación dispone de varias operaciones sobre HTTP:

- **GET:** para obtener todas las secuencias, referencias, conjuntos género y especie e informes. En el caso de añadir a la URI un parámetro de ruta, se lo trataría como el ID del recurso solicitado. También para obtener los ficheros asociados a ciertos recursos.

- **PATCH:** para modificar las abreviaciones de los conjuntos género y especie.

- **POST:** para crear nuevos recursos o modificar aquellos suscritos a la herramienta de secuenciación del genoma.

- **DELETE:** para eliminar un conjunto género y especie si este no está referenciado por ningún recurso.

- **OPTIONS:** para habilitar el *Cross-origin resource sharing* (CORS),[23] necesario para que peticiones de origen cruzado, como las realizadas desde un *frontend*, sean permitidas.

Finalmente, la aplicación dispone de autenticación y autorización. La autenticación se realiza mediante usuario y contraseña, existiendo actualmente un único usuario, ya que no existen funcionalidades relacionadas con la gestión de usuarios. La encriptación de la contraseña se implementó con *PBKDF2*.[24] Por otra parte, la autenticación se lleva a cabo mediante *JWT*. *JSON Web Token* (JWT) es un estándar abierto (RFC 7519) que define una forma de transmitir información de forma segura entre las partes en formato JSON. Esta información es confiable y puede ser verificada porque está firmada digitalmente. Los JWTs pueden ser firmados usando una clave secreta o un par de claves pública/privada.[25] En nuestro caso, el JWT se firma y verifica con una clave privada haciendo uso de *HMAC*, incluyendo los campos *issued at* y *expires at*.

5.2.2.2. Sæbio: Nullarbor API

En el proyecto de fin de título llamado «Arquitectura de datos para la caracterización y análisis de microorganismos multiresistentes»[1] se logró instalar el pipeline Nullarbor en una máquina virtual de Linux en un servidor del SIANI y el control de la aplicación se abstrajo mediante una serie de scripts preconfigurados que le fueron entregados al Hospital Universitario por otro laboratorio familiarizado con el uso de la herramienta. El servicio web objetivo de este proyecto de fin de título necesita hacer uso de Nullarbor para la secuenciación del genoma de un microorganismo y la generación de informes de vigilancia epidemiológica. En las metodologías descritas en el TFT01 se planteó seguir una arquitectura *serverless*, de manera que inicialmente se decidió contenerizar con *Proxmox* a SNA (Sæbio: Nullarbor API) junto a Nullarbor para levantar instancias de Nullarbor según la solicitud y terminarlas una vez acabase el proceso asociado a la solicitud. La computación sin servidor (serverless) es un modelo de ejecución en el que el proveedor en la nube se encarga de ejecutar un fragmento de código mediante la asignación dinámica de los recursos, cobrando únicamente por los recursos utilizados durante la ejecución. El código, generalmente, se ejecuta dentro de contenedores sin estado que pueden ser activados por diversos eventos que incluyen solicitudes HTTP, entre otros.[26] A lo largo del desarrollo del proyecto de fin de título no se pudo contar con una infraestructura basada en Proxmox; sin embargo, se realizó una arquitectura preparada para ser serverless al haberse implementado una segunda API separada de SGSA. Esta segunda API está lista para ser contenerizada junto a una instalación de Nullarbor como proyecto futuro.

Dicha API es Sæbio: Nullarbor API, que ofrece la ejecución de las distintas partes del pipeline Nullarbor según la solicitud que recibe. Su desarrollo se lleva a cabo, también,

con el micro-framework Spark y, por tanto, el lenguaje de programación elegido es Java. Asimismo se emplea Maven y se siguen los principios SOLID, además de separar la lógica de la aplicación de la lógica de Spark como en SGSA.

SNA no sigue TDD ni BDD, ya que Sæbio: genome sequencing API sí lo sigue y, al ser su único cliente, en ella se hace todas las comprobaciones necesarias para evitar fallos en el uso de Nullarbor.

La aplicación dispone de cuatro operaciones sobre HTTP:

- **POST /trim:** realiza el trimming sobre la secuencia recibida.
- **POST /analysis:** crea el recurso solicitud de análisis.
- **PATCH /analysis/{token}:** añade a la solicitud de análisis indicada el fichero adjunto.
- **POST /analysis/{token}:** realiza el proceso de análisis del genoma de la solicitud de análisis indicada.

La ejecución de Nullarbor se hace a través de bash scripts. Los procesos de Nullarbor suelen durar un largo periodo de tiempo: el análisis puede durar varias horas hasta que finalice. Cuando el servidor *Jetty* embebido de Spark responde a la solicitud, los procesos iniciados por la *Java Virtual Machine* (JVM) para dicha solicitud son terminados. Para evitar esto, se hace uso de la utilidad *nohup*.[27]

5.2.3. Desarrollo del frontend

Una vez estuvieron las APIs implementadas y funcionales, cubriendo las funcionalidades requeridas por el cliente, se decidió proceder con el desarrollo de la interfaz gráfica que en un primer momento se había planteado como opcional. Para brindar de gran fluidez al usuario, la *GUI* (*Graphical User Interface*) se lleva a cabo como una *Single Page Application* (*SPA*). En una SPA la página solo se carga una vez, modificándose dinámicamente los recursos cuando se requiera; normalmente, por la interacción del usuario. Esto le da al usuario la sensación de estar utilizando una aplicación de escritorio.[28]

La implementación se hizo con *Vue 3*, un framework progresivo de *JavaScript*, y la gestión de paquetes y del desarrollo se llevó a cabo con *npm*. Asimismo, se hace uso de diversos paquetes: *Vue Router* [29] para el control de las rutas y facilitar la implementación como una SPA, *Vuex* [30] para el control de estados con una *store* para todos los componentes de la aplicación, Axios [31] para las llamadas a la API, *Font Awesome* [32] para los iconos, *Bootstrap* para facilitar el diseño, entre otros paquetes.

La GUI permite al usuario hacer uso de las funcionalidades de Sæbio: genome sequencing API de manera sencilla, cómoda y visual, aliviándole del uso de CLI, uno de los problemas con los que se encontraba el personal del hospital. También, siempre que se produzca un error, se le informa al usuario para que sepa qué ocurre en todo momento.

La aplicación se divide en cuatro apartados: secuencias, referencias, definiciones (género y especie) e informes. Cada una de estas secciones le muestra al usuario una lista de todos los recursos de dicha sección paginados por mes y año, excepto en definiciones que no hay paginación. Además, las secuencias, referencias e informes se pueden filtrar por género y especie. Por otra parte, cada apartado da al usuario la posibilidad de crear un nuevo recurso: subir una secuencia, subir una referencia, crear una nueva definición o solicitar un análisis del genoma. También se puede modificar las palabras clave de una definición o eliminar una definición. Todas estas acciones, exceptuando la solicitud de un análisis, se realizan en la misma ruta en la que se encuentra el usuario para una mayor fluidez y evitar una navegación excesiva.

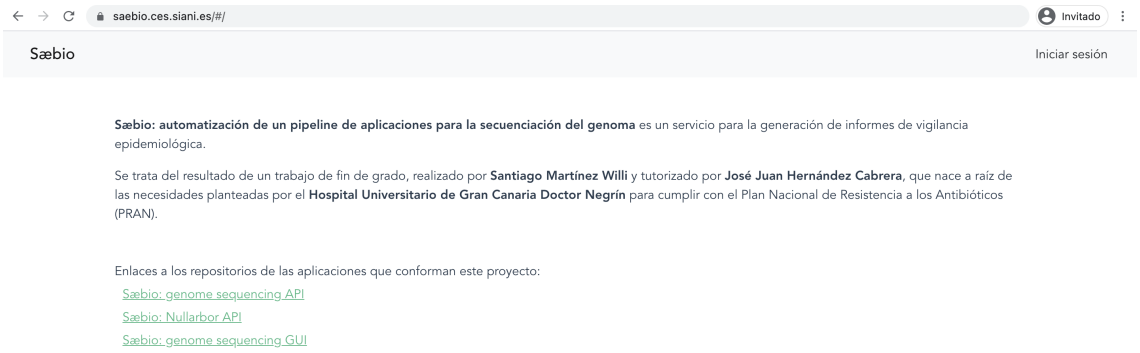


Figura 5.2: Vista del *home*

El apartado *home* es el punto de aterrizaje en la aplicación y muestra una breve información sobre el proyecto.

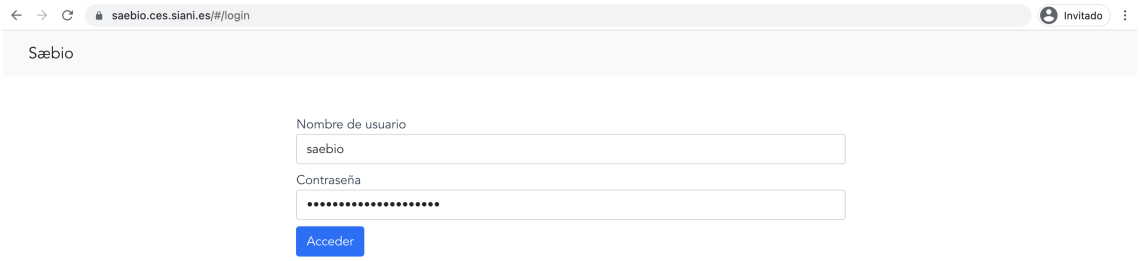


Figura 5.3: Inicio de sesión en la aplicación

Figura 5.4: Secuencias filtradas según su fecha de secuencia

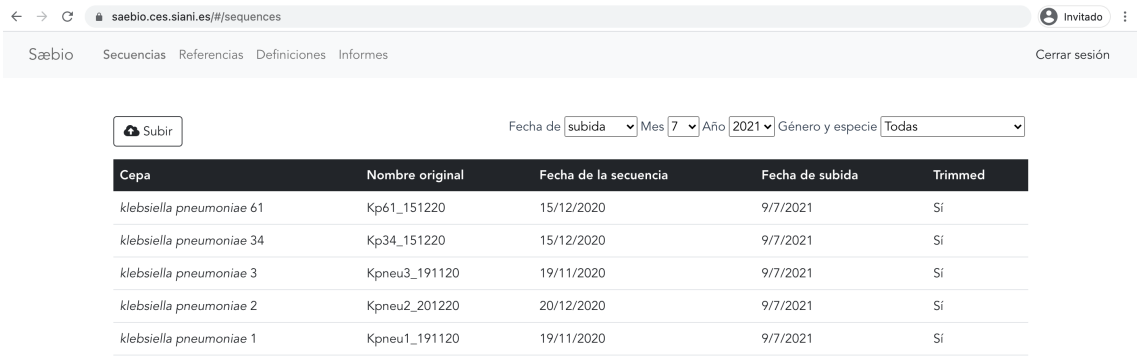


Figura 5.5: Secuencias filtradas según su fecha de subida

Las secuencias pueden filtrarse tanto por su fecha de secuencia como por su fecha de subida indicándolo con el *select Fecha de*, siendo por defecto el de secuencia.

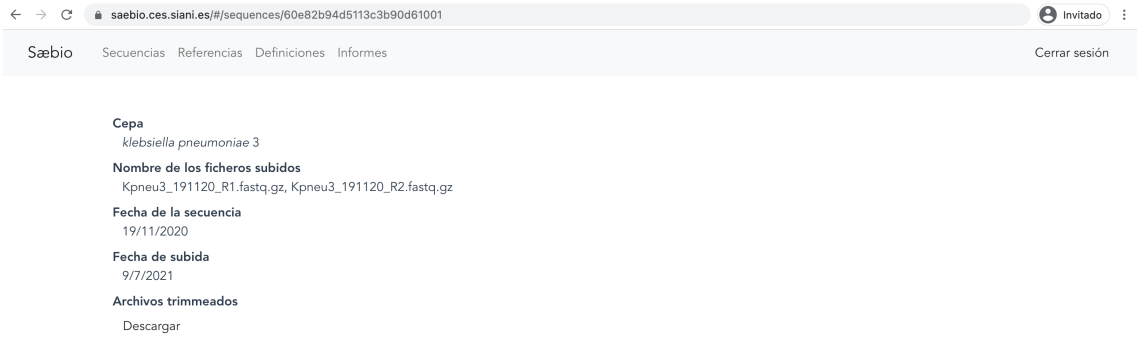


Figura 5.6: Vista de una secuencia específica

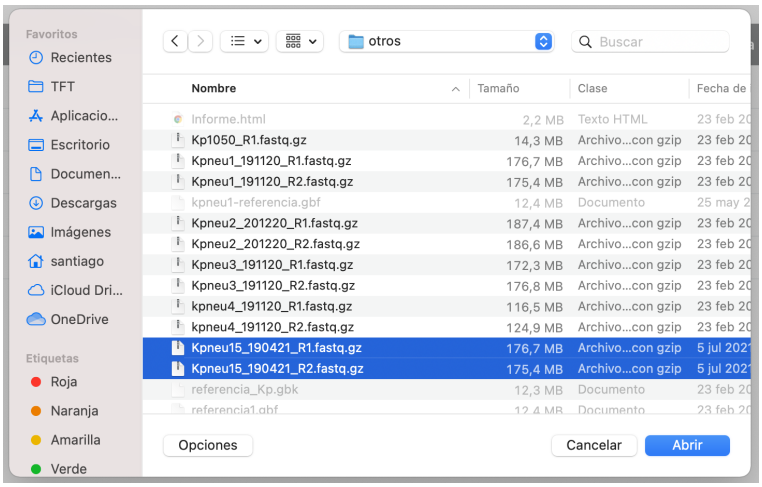


Figura 5.7: Vista de selección de ficheros para subir una secuencia

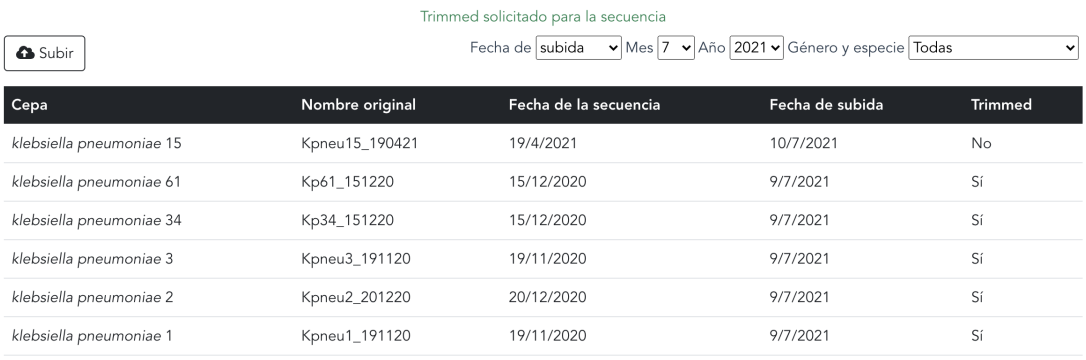


Figura 5.8: Vista tras subir una secuencia

Mientras la secuencia está subiéndose al servidor, el cursor cambia su estilo a *progress* para indicar al usuario que se está llevando a cabo el proceso. Una vez se termina de subir, se notifica al usuario. Si la secuencia no está *trimmed* y, por tanto, se solicita el *trimming*, al usuario le aparecerá que dicha secuencia no está trimmed por el momento.

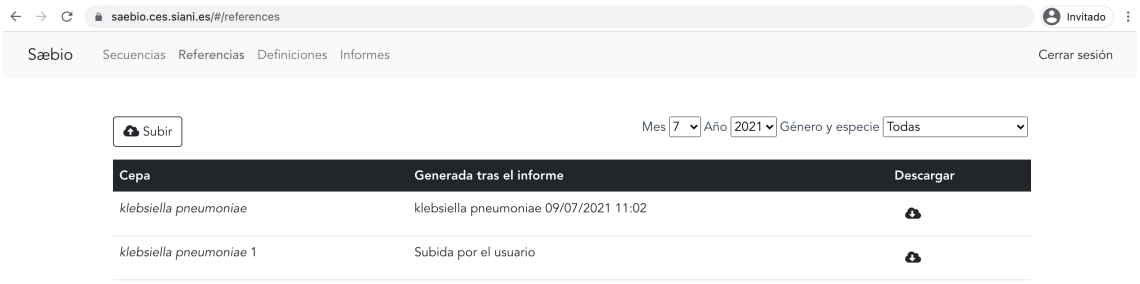


Figura 5.9: Vista de la sección de referencias

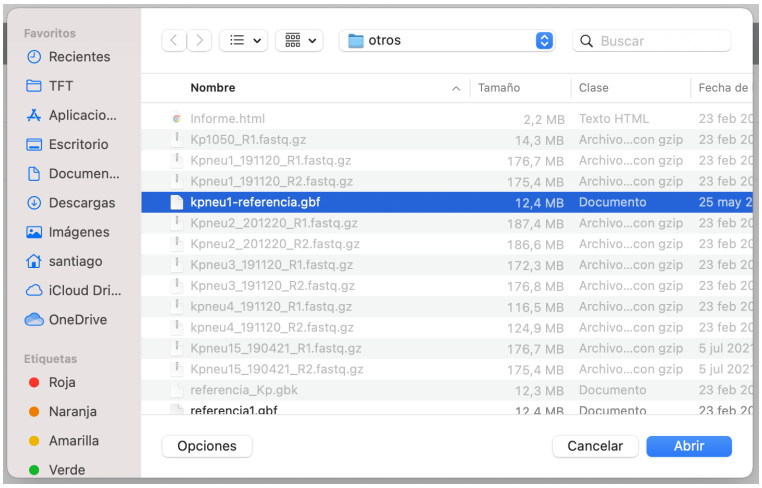


Figura 5.10: Vista de selección de un fichero para subir una referencia

Figura 5.11: Vista de la sección de informes

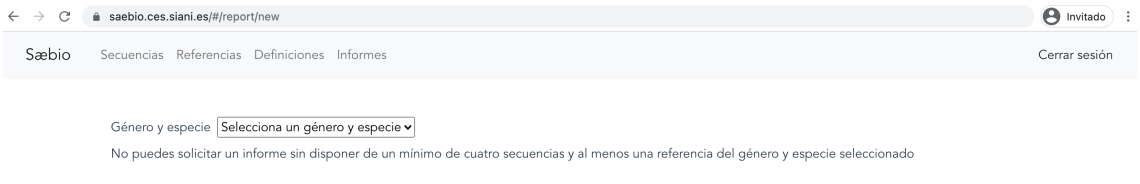


Figura 5.12: Vista de la sección de solicitud de informe

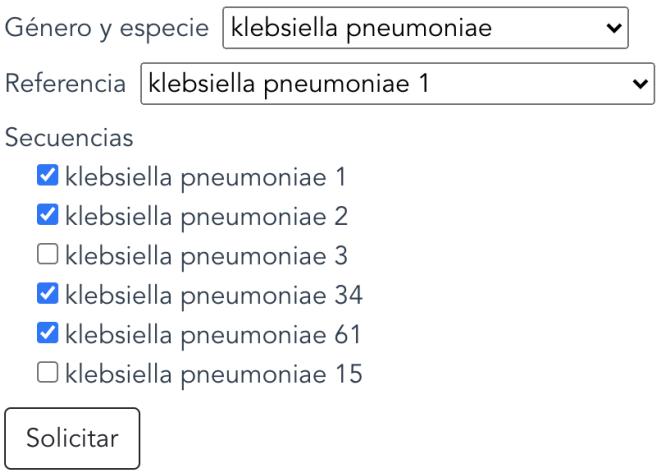


Figura 5.13: Selección de la referencia y secuencias para el análisis

La referencia y secuencias que se muestran son las que su género y especie coincide con el seleccionado. Se debe seleccionar una referencia del *select* y, al menos, cuatro secuencias.

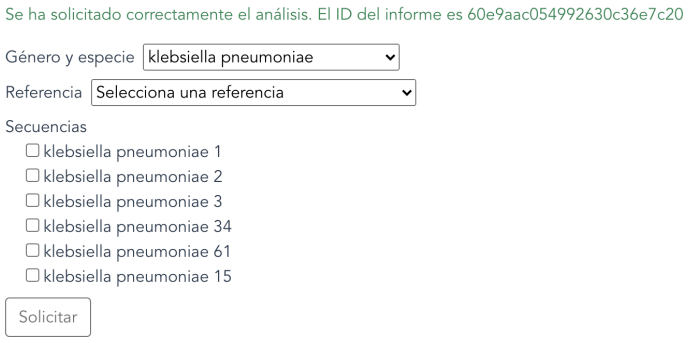


Figura 5.14: Informe solicitado

Desde que se solicita hasta que la solicitud es aprobada, el cursor cambia su estilo a *progress* para indicar al usuario que se está llevando a cabo el proceso. Una vez se inicia el proceso de análisis, se notifica al usuario un identificador con el que localizar

al informe.

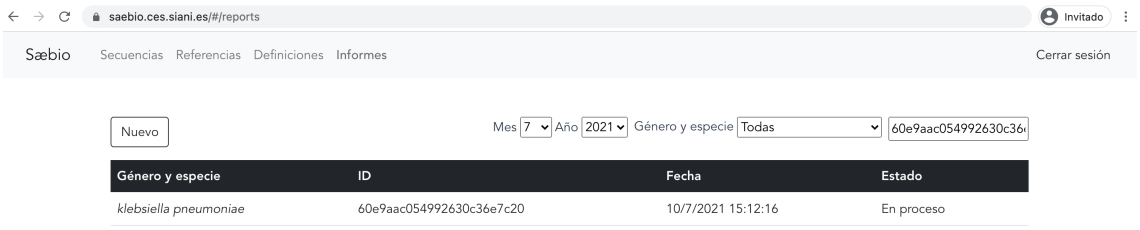


Figura 5.15: Vista de la sección de informes con filtro por ID

En la sección de informes se puede filtrar los informes por el ID en el buscador de la esquina superior derecha.

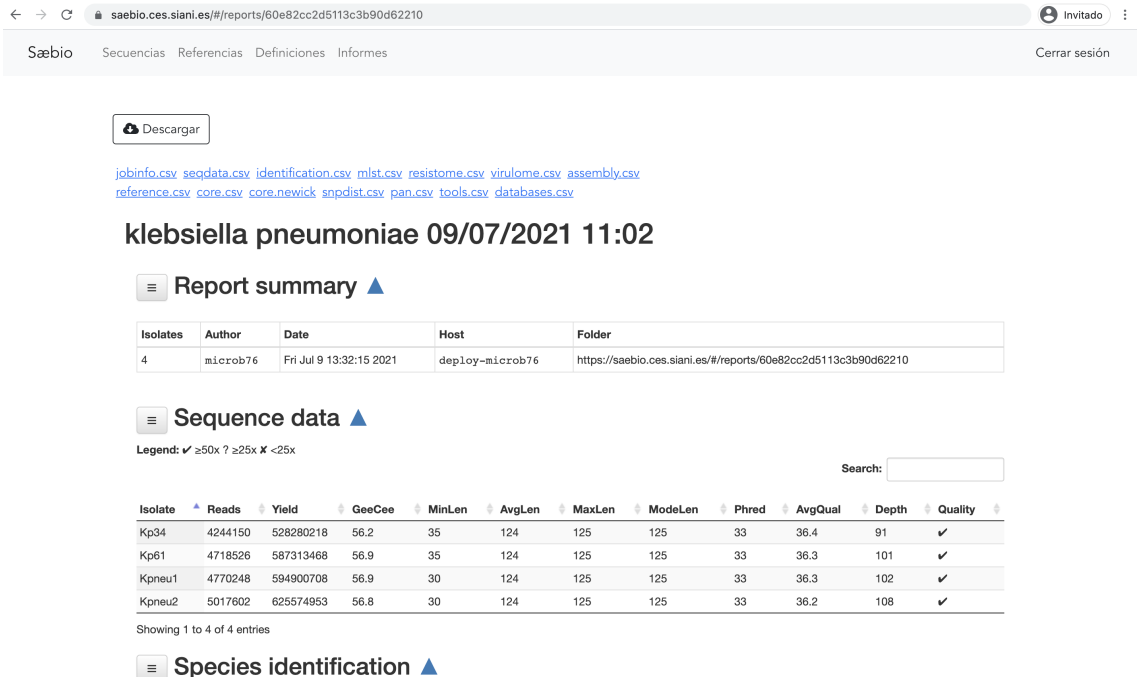


Figura 5.16: Vista de un informe

Una vez finalizado con éxito el análisis del genoma de los aislados seleccionados, se

puede acceder a su informe. Este informe es un *HTML* generado por la herramienta de secuenciación del genoma, Nullarbor. Dicho HTML se muestra embebido dentro de un elemento HTML *iframe* (*Inline Frame*), al que, para que se muestre y funcione correctamente, se le hacen ciertas modificaciones como eliminar sus enlaces a ficheros asociados y añadirlos fuera del iframe. El anexo A.1 muestra un informe extraído de la aplicación.

En la sección de informes, si el análisis falló, en vez de darle al usuario la opción de ver el informe, se le muestra un botón para descargar el archivo *log* del proceso de análisis fallido.

Figura 5.17: Vista de la sección de definiciones (género y especie)

Figura 5.18: Vista de la creación de una nueva definición

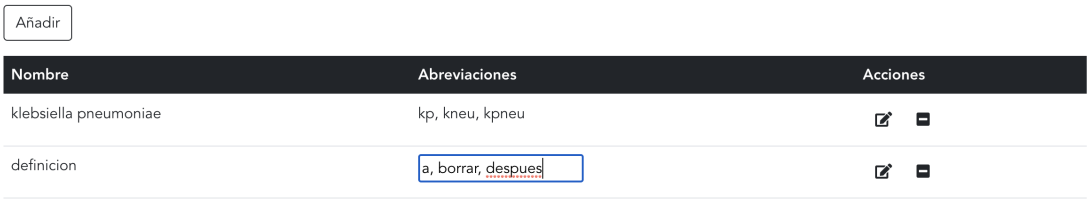


Figura 5.19: Vista de la edición de una definición

5.3. Despliegue

Los tres proyectos desarrollados en este trabajo de fin de título se encuentran actualmente desplegados en servidores del SIANI.

Ambas APIs se compilan con Maven haciendo uso del *plugin Assembly*. Este plugin permite compilar el código en un *JAR* que contiene todas las dependencias.[33] La aplicación Sæbio: Nullarbor API está desplegada actualmente en un servidor del SIANI donde está instalado Nullarbor, con una configuración que trata de imitar la del servidor en el que el hospital tiene instalado dicho pipeline. Por otra parte, la aplicación Sæbio: genome sequencing API está desplegada en otro servidor del SIANI. En este servidor está desplegada también la aplicación desarrollada en Vue, que recibe el nombre de *Sæbio: genome sequencing GUI*. Para el despliegue de esta aplicación, primero se compiló con npm y luego se sirvieron los archivos estáticos de la carpeta *dist* resultante en el servidor, ya que es un servidor HTTP.

La API de Sæbio: Nullarbor API es accesible solamente desde los propios servidores del SIANI, por lo que su único cliente es la aplicación Sæbio: genome sequencing API. Esta segunda API es accesible a partir de la ruta `https://saebio.ces.siani.es/api` y necesita de autorización para hacer uso de sus operaciones. Para recibir la autorización necesaria es requisito autenticarse con usuario y contraseña. Actualmente solo existe un usuario, ya que la aplicación no dispone de funcionalidades relacionadas con la gestión de usuarios. Finalmente, la GUI es accesible desde `https://saebio.ces.siani.es`.

6 Conclusiones

La herramienta bioinformática resultante de este trabajo de fin de grado cumple con los objetivos planteados al inicio del proyecto. Se ha entregado al personal del hospital una herramienta bioinformática que automatiza el proceso de secuenciación del genoma de una bacteria y facilita la generación de informes de vigilancia epidemiológica.

Mientras que el *pipeline* de generación de informes de microbiología completos elegido por el Hospital Universitario de Gran Canaria Doctor Negrín, Nullarbor, presenta un uso e instalación de elevada dificultad, la aplicación desarrollada ofrece un uso sencillo que abstrae de la utilización de línea de comandos y de conocimientos informáticos y no requiere de instalación alguna, ya que se ejecuta en un servidor HTTP y se puede acceder a ella desde cualquier dispositivo que cuente con un navegador web, a excepción de *Internet Explorer* (IE). Esta falta de compatibilidad de *Vue* con IE no se considera un problema, puesto que antes la herramienta de secuenciación del genoma solo estaba disponible en plataformas *UNIX* y ahora es *cross-platform*. Además, al ejecutarse en un servidor HTTP, la aplicación no depende de los recursos del usuario.

La interfaz gráfica, además de ser sencilla de usar, embebe el informe de vigilancia epidemiológica por lo que para verlo no es necesario estar descargándolo, sino que tanto visualizarlo como compartirlo con el personal del hospital se realiza a través de una *URL* y no a través del traspaso y manejo manual de ficheros. La interfaz permite filtrar los recursos por fecha y/o género y especie, lo cual aporta valor al microbiólogo. Por otro lado, esta *GUI* se implementó como una *Single Page Application* (*SPA*), dándole al usuario una interacción más fluida y lo más parecida posible a una aplicación de escritorio.

La aplicación integra el pipeline de secuenciación del genoma con una plataforma de datos sanitarios; siendo la aplicación, entonces, un ingestor de datos de secuenciación

genética. Con esto se logra que el personal del laboratorio no tenga que preocuparse de la gestión y transporte de los datos. Esta plataforma de datos es una base de datos *NoSQL* implementada con *MongoDB*, en vez de ser la de otro proyecto de fin de grado titulado «Saebio: Arquitectura de datos para la caracterización y análisis de microorganismos multirresistentes»,[1] que inicialmente se había planteado como una posibilidad. La principal razón de esta implementación con una base de datos documental se debe a la propia naturaleza documental del proceso de análisis de secuenciación del genoma. Además, el repositorio de datos es lo suficientemente flexible para ser modificados sus documentos según las necesidades futuras del hospital. Lo único que haría falta incluir en los documentos es un campo *versión* que indique en cuál se hayan los datos.

Por tanto, con este servicio web el personal del hospital no solo puede solicitar el análisis de la secuenciación del genoma de múltiples aislados, sino que también es capaz de almacenar de manera centralizada secuencias *trimmed* (estando ya *trimmed* o solicitando el *trimming* en la subida), referencias y los ficheros resultantes del proceso de análisis; entre ellos, el informe de vigilancia epidemiológica.

En los objetivos iniciales se plantea que el servicio web debe permitir que, cuando el usuario solicite el análisis del genoma de una bacteria, este pueda suscribirse al evento para que pueda obtener información sobre este proceso en cualquier momento. También se planteó en las metodologías que se llevaría a cabo siguiendo una arquitectura *publisher/subscriber*. La metodología que se siguió finalmente no es estrictamente *publisher/subscriber*, puesto que no dispone de canales de mensajería asíncronos. Sin embargo, el cliente de la herramienta de secuenciación del genoma se suscribe al proceso cuando lo solicita y, una vez este finaliza, la herramienta notifica al cliente del resultado. Por tanto, el personal del hospital es capaz de ver en todo momento si el proceso solicitado continúa o finalizó, ya sea de manera exitosa o fallida. Como trabajo futuro, sería interesante que el usuario sea capaz de ver el progreso del proceso. Para esto podría hacerse un estudio del *log* de Nullarbor con el fin de elaborar una estimación a partir de dicho fichero.

En las metodologías se propuso seguir una arquitectura *Serverless* y, aunque finalmente no se pudo contar con una infraestructura basada en *Proxmox*, en vías futuras se puede implementar esta infraestructura. Esto es posible gracias a que contenerizar a *Sæbio: Nullarbor API* junto a una instalación de Nullarbor es categóricamente simple, ya que se dejó preparada una arquitectura para ser *Serverless* al haberse implementado esta API separada de *Sæbio: genome sequencing API*.

Por otra parte, inicialmente se planteó seguir una arquitectura *Lambda*. Esta arquitectura permite una baja latencia de lectura y actualización ante grandes cargas de trabajo al contar con las modalidades de procesamiento *batch* y *stream*.[34] No obstante, una vez se inició el proyecto no se consideró necesario contar con una modalidad *stream*, por lo que no se siguió esta arquitectura.

Un objetivo futuro que podría llevarse a cabo es la integración entre los múltiples trabajos de fin de grado *Saebio*. Esta integración daría un gran valor el hospital porque centralizaría el acceso a todas las aplicaciones desarrolladas.

Este trabajo de fin de grado ha continuado la linea de trabajo iniciada por el Instituto Universitario SIANI con el trabajo de fin de título de Omar Verona Kämpfer.[1] Este proyecto ha sido un trabajo multidisciplinar, al ser una colaboración entre la Universidad de Las Palmas de Gran Canaria con el Hospital Universitario de Gran Canaria Doctor Negrín. A nivel personal, he aprendido mucho sobre diversas áreas. He estado en contacto con personal del laboratorio de microbiología del hospital, interiorizándome en la secuenciación del genoma de una bacteria y en las herramientas bioinformáticas implicadas en el proceso. Profundicé en lo aprendido en la carrera sobre los *web services*; concretamente, en la arquitectura *REST*. Aprendí sobre bases de datos documentales con MongoDB y las ventajas y desventajas del uso de bases de datos SQL frente a NoSQL. También me familiaricé con las arquitecturas Serverless y publisher/subscriber.

Por otra parte, aprendí a desarrollar interfaces gráficas SPA con *Vue*, *Vuex*, *Vue Router* y *Axios*, para lo cual tuve que profundizar en *JavaScript*.

Con el proyecto ya finalizado, me siento mucho más experimentado con las metodologías *Test Driven Development* (TDD) y *Behaviour Driven Development* (BDD), ya que todo el desarrollo de Sæbio: genome sequencing API siguió estrictamente estas dos metodologías, con lo que se intenta minimizar el número de errores de la aplicación.

Además, tuve que repasar *Bash* para la realización de diversos *scripts* con los que ejecutar Nullarbor y automatizar la ejecución de los tests funcionales.

También me familiaricé con la compilación y generación de artefactos *JAR* y la compilación para producción de una aplicación Vue.

A nivel personal, sentí que lo que se enseña en la universidad, al menos en la especialización *Ingeniería del Software*, no es suficiente para el desarrollo de este proyecto. Opino que debería darse más importancia y enseñarse más sobre TDD, ya que es una metodología que, si bien no siempre es aplicable, minimiza la aparición de errores en el software. También considero esencial el aprendizaje sobre bases de datos NoSQL, tecnologías muy utilizadas en el sector de la ingeniería del software que presentan ventajas sobre las bases de datos SQL según el tipo de proyecto a desarrollar.

Las necesidades que este proyecto ha intentado cubrir no son exclusivas del Hospital Universitario de Gran Canaria Doctor Negrín. Actualmente, todos los laboratorios de sanidad pública reciben cada vez más y más cantidades de datos con valor. Sumado a esto, nos encontramos actualmente en la situación de pandemia de COVID-19. Ahora, más que nunca, se hace imperativo la automatización de los procesos bioinformáticos y una integración eficaz entre las plataformas de datos sanitarios y las herramientas bioinformáticas, con el requisito de adaptarse a las situaciones futuras.

La aplicación desarrollada en este trabajo de fin de grado sería capaz de satisfacer estas demandas para la generación de informes de vigilancia epidemiológica. Para adaptar la aplicación a cambios futuros en los datos, solo haría falta actualizar la versión y estructura de los documentos a almacenar en el repositorio. Incluso si se desease cambiar la integración con MongoDB por otra, solo sería necesario cambiar el módulo que se encarga del acceso a los datos. Como se comentó en el capítulo de metodología, este proyecto se adecúa a los principios SOLID y, como tal, es muy modular logrando un bajo acoplamiento.

En relación a esto, si en un futuro se desea cambiar la herramienta Nullarbor por otra, bastaría con reemplazar a Sæbio: Nullarbor API por otra API o sustituir los *scripts* de dicha API por otros que ejecuten la herramienta bioinformática deseada.

Actualmente, la aplicación se encuentra desplegada en los servidores del SIANI y es completamente funcional, con lo que se ha logrado brindar una herramienta bioinformática que da solución para la generación de informes de vigilancia epidemiológica, entregándole valor al personal del laboratorio de microbiología.

Anexos

A Informes

A.1. Informe de vigilancia epidemiológica

klebsiella pneumoniae 10/7/2021 15:12:16

≡ Report summary ▲

Isolates	Author	Date	Host	Folder
4	microb76	Sat Jul 10 16:42:49 2021	deploy-microb76	https://saebio.ces.siani.es/#/reports/60e9aac054992630c36e7c20

≡ Sequence data ▲

Legend: ✔ ≥50x ? ≥25x ✘ <25x

Search:

Isolate	Reads	Yield	GeeCee	MinLen	AvgLen	MaxLen	ModeLen	Phred	AvgQual	Depth	Quality
Kp34	4244150	528280218	56.2	35	124	125	125	33	36.4	91	✔
Kp61	4718526	587313468	56.9	35	124	125	125	33	36.3	101	✔
Kpneu1	4770248	594900708	56.9	30	124	125	125	33	36.3	102	✔
Kpneu2	5017602	625574953	56.8	30	124	125	125	33	36.2	108	✔

Showing 1 to 4 of 4 entries

≡ Species identification ▲

Legend: ✔ ≥80% ? ≥65% ✘ <65%

Search:

Isolate	#1 Match	%	#2 Match	%	#3 Match	%	#4 Match	%	Quality
Kp34	**Klebsiella pneumoniae**	**71.67**	unclassified	0.15	Escherichia coli	0.08	Salmonella enterica	0.03	?
Kp61	**Klebsiella pneumoniae**	**78.42**	unclassified	0.06	Escherichia coli	0.03	Salmonella enterica	0.01	?
Kpneu1	**Klebsiella pneumoniae**	**78.48**	unclassified	0.04	Escherichia coli	0.03	Salmonella enterica	0.01	?
Kpneu2	**Klebsiella pneumoniae**	**76.33**	Escherichia coli	0.05	unclassified	0.04	Salmonella enterica	0.01	?

Showing 1 to 4 of 4 entries

≡ MLST ▲

Search:

Isolate	Scheme	Sequence Type	Allele	Allele	Allele	Allele	Allele	Allele	Allele	Quality
Kp34	kpneumoniae	11	gapA(3)	infB(3)	mdh(1)	pgi(1)	phoE(1)	rpoB(1)	tonB(4)	✔
Kp61	kpneumoniae	11	gapA(3)	infB(3)	mdh(1)	pgi(1)	phoE(1)	rpoB(1)	tonB(4)	✔
Kpneu1	kpneumoniae	11	gapA(3)	infB(3)	mdh(1)	pgi(1)	phoE(1)	rpoB(1)	tonB(4)	✔
Kpneu2	kpneumoniae	11	gapA(3)	infB(3)	mdh(1)	pgi(1)	phoE(1)	rpoB(1)	tonB(4)	✔
Reference	kpneumoniae	11	gapA(3)	infB(3)	mdh(1)	pgi(1)	phoE(1)	rpoB(1)	tonB(4)	✔

Showing 1 to 5 of 5 entries

≡ Resistome ▲

Legend: ✔ ≥95% coverage ? <95% coverage ✘ absent

Search:

Isolate	Found	A7J11_02581	aac(3)-IIa	aadA2	aph(3'')-Ib	aph(3')-Ia	aph(6)-Id	arsB-mob	blaCTX-M-15	blaOXA-1	blaOXA-48	blaSHV-158	b
Kp34	11	?	.	.	.	.	.	✔	✔	✔	✔	✔	.
Kp61	21	?	✔	✔	✔	.	✔	✔	✔	✔	✔	✔	✔
Kpneu1	21	?	✔	✔	✔	.	✔	✔	✔	✔	✔	✔	✔
Kpneu2	23	?	✔	✔	✔	✔	✔	?	✔	✔	✔	✔	✔

Showing 1 to 4 of 4 entries

≡ Virulome ▲

Legend: ✔ ≥95% coverage ? <95% coverage ✘ . absent

Search:

Isolate	Found	entA	entB	entE	entS	fepA	fepB	fepC	fepD	fepG	fimA	fimE	fyuA	irp1	irp2	mgtB	mgtC
Kp34	33	✔	✔	✔	?	✔	?	?	?	?	?	✔	✔	✔	✔	?	?
Kp61	31	✔	✔	✔	?	✔	?	?	?	?	.	.	✔	✔	✔	?	?
Kpneu1	31	✔	✔	✔	?	✔	?	?	?	?	.	.	✔	✔	✔	?	?
Kpneu2	33	✔	✔	✔	?	✔	?	?	?	?	?	✔	✔	✔	✔	?	?

Showing 1 to 4 of 4 entries

≡ Assembly and annotation ▲

Legend: ✔ Typical ? $\mu_{Contigs}$=84 ∧ μ_{bp}=5534409 ✘ Contigs > 1.5μ ∨ bp ± 0.2μ

Search:

Isolate	Contigs	bp	ok	Ns	gaps	min	avg	max	N50	CDS	rRNA	tRNA	tmRNA	Quality
Kp34	74	5380999	5380999	0	0	615	72716	506442	195492	5083	9	0	0	✔
Kp61	86	5570564	5570564	0	0	592	64774	617066	221587	5279	8	0	0	✔
Kpneu1	88	5570005	5570005	0	0	598	63295	617066	219951	5277	8	0	0	✔
Kpneu2	88	5616071	5616071	0	0	588	63818	360263	203667	5319	8	0	0	✔

Showing 1 to 4 of 4 entries

≡ Reference genome ▲

Search:

Sequence	Length	Description
NC_016838	122799	NC_016838 Klebsiella pneumoniae subsp. pneumoniae HS11286 plasmid pKPHS1, complete sequence.
NC_016839	105974	NC_016839 Klebsiella pneumoniae subsp. pneumoniae HS11286 plasmid pKPHS3, complete sequence.
NC_016840	3751	NC_016840 Klebsiella pneumoniae subsp. pneumoniae HS11286 plasmid pKPHS4, complete sequence.
NC_016841	1308	NC_016841 Klebsiella pneumoniae subsp. pneumoniae HS11286 plasmid pKPHS6, complete sequence.
NC_016845	5333942	NC_016845 Klebsiella pneumoniae subsp. pneumoniae HS11286 chromosome, complete genome.
NC_016846	111195	NC_016846 Klebsiella pneumoniae subsp. pneumoniae HS11286 plasmid pKPHS2, complete sequence.
NC_016847	3353	NC_016847 Klebsiella pneumoniae subsp. pneumoniae HS11286 plasmid pKPHS5, complete sequence.
TOTAL	5682322	Total reference size in bp

Showing 1 to 8 of 8 entries

≡ Core genome ▲

Legend: ✔ ≥90% used ? ≥70% used ✘ <70% used

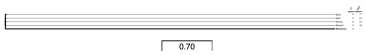
Search:

Isolate	LENGTH	ALIGNED	UNALIGNED	VARIANT	HET	MASKED	LOWCOV	%Used	Quality
Kp34	5682322	5153452	520253	6820	643	0	7974	90.69	✔
Kp61	5682322	5173025	496360	6730	843	0	12094	91.04	✔
Kpneu1	5682322	5173064	495665	6730	848	0	12745	91.04	✔
Kpneu2	5682322	5212435	464476	6731	861	0	4550	91.73	✔
Reference	5682322	5682321	0	0	0	0	1	100.00	✔

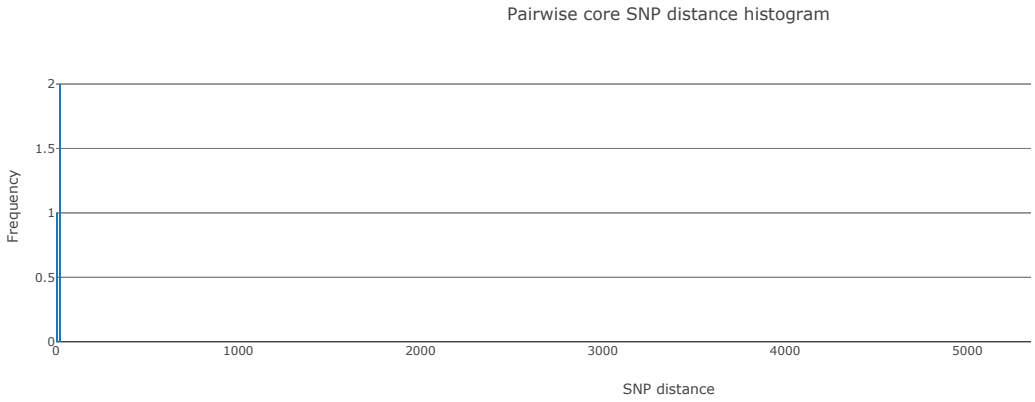
Showing 1 to 5 of 5 entries

≡ Core SNP Phylogeny ▲

4 taxa, 6763 SNPs



≡ Pairwise core SNP distances ▲

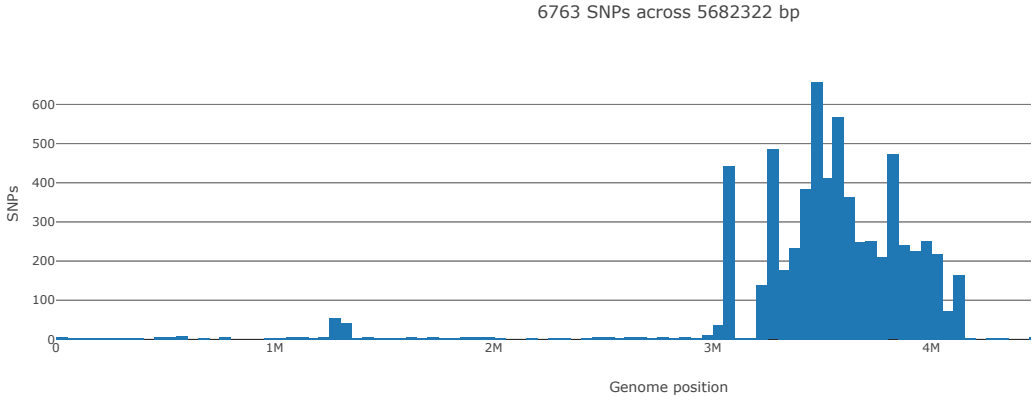


Search:

	Kp34	Kp61	Kpneu1	Kpneu2	Reference
Kp34	0	52	52	55	6728
Kp61	52	0	0	21	6722
Kpneu1	52	0	0	21	6722
Kpneu2	55	21	21	0	6731
Reference	6728	6722	6722	6731	0

Showing 1 to 5 of 5 entries

≡ Core SNP density ▲



≡ Pan genome ▲

Ortholog class	Definition	Count
Core genes	(99% <= strains <= 100%)	5028
Soft core genes	(95% <= strains < 99%)	0
Shell genes	(15% <= strains < 95%)	318
Cloud genes	(0% <= strains < 15%)	0
Total genes	(0% <= strains <= 100%)	5346

≡ Software versions ▲

Tool	Version
Abricate	0.8.13
BWA MEM	0.7.17-r1188
FastTree	2.1.10 Double precision (No SSE3)
FreeBayes	1.3.2-dirty
IQtree	1.6.12 for Linux 64-bit built Jul 19 2020
Kraken	1.0
MLST	2.19.0
MegaHit	1.2.6
Newick-Utils	1.6
Nullarbor	2.0.20181010
Prokka	1.14.6
Roary	3.12.0
SAMtools	1.7
SKESA	2.3.0
SPAdes	3.13.0
Shovill	1.0.4
Snippy	4.6.0
Trimmomatic	0.36
centrifuge	1.0.3-beta
seqret	6.6.0.0
seqtk	1.2-r94
snp-dists	0.6.3

≡ Databases ▲

Database	Name	Date
Kraken	/home/microb76/databases/minikraken_20171101_8GB_dustmasked	2019-07-19
mlst	/home/microb76/.linuxbrew/Cellar/mlst/2.19.0/libexec/db/blast/mlst.fa	2020-02-24

≡ About ▲

This primary author of Nullarbor is Torsten Seemann (http://tseemann.github.io/).
You can download the software from Github (https://github.com/tseemann/nullarbor).
Please report bugs at the Nullarbor Issue Tracker (https://github.com/tseemann/nullarbor/issues).
If you use Nullarbor please use the latest citation (https://github.com/tseemann/nullarbor/#citation).

Bibliografía

[1] Omar Verona Kämpfer. Arquitectura de datos para la caracterización y análisis de microorganismos multirresistentes. B.S. thesis, Universidad de Las Palmas de Gran Canaria, 2020.

[2] Seemann T, Goncalves da Silva A, Bulach DM, Schultz MB, Kwong JC, and Howden BP. Nullarbor. `https://github.com/tseemann/nullarbor`. (último acceso: 4.07.2021).

[3] Mariaconcetta Bilotta, Giuseppe Tradigo, and Pierangelo Veltri. *Bioinformatics Data Models, Representation and Storage*, volume 1, pages 110–116. Elsevier, 2019.

[4] Nicholas M Luscombe, Dov Greenbaum, and M Gerstein. What is bioinformatics? a proposed definition and overview of the field. *Methods of Information in Medicine*, 40(4):346–58, 2001.

[5] João Carriço, Mirko Rossi, Jacob Moran-Gilad, Gary Domselaar, and Mário Ramirez. A primer on microbial bioinformatics for non-bioinformaticians. *Clinical Microbiology and Infection*, 24(4), 2018.

[6] Wikipedia. FASTQ format. `https://en.wikipedia.org/wiki/FASTQ_format`. (último acceso: 30.06.2021).

[7] Wikipedia. Formato FASTA. `https://es.wikipedia.org/wiki/Formato_FASTA`. (último acceso: 30.06.2021).

[8] Academind. SQL vs NoSQL or MySQL vs MongoDB. `https://www.youtube.com/watch?v=ZS_kXvOeQ5Y&ab_channel=Academind`, Jul 2018.

[9] MongoDB Inc. JSON and BSON. `https://www.mongodb.com/json-and-bson`. (último acceso: 30.06.2021).

[10] Atlassian. Flujo de trabajo de Gitflow. `https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow`. (último acceso: 30.06.2021).

[11] Xolv.io. Should TDD and BDD be used in conjunction? https://stackoverflow.com/a/33775671. (último acceso: 30.06.2021).

[12] Sergey Berezovskiy and MattyMatt. Does TDD include integration tests? https://stackoverflow.com/a/18998744. (último acceso: 30.06.2021).

[13] Wikipedia. Transferencia de Estado Representacional. https://es.wikipedia.org/w/index.php?title=Transferencia_de_Estado_Representacional&oldid=136565584. (último acceso: 3.07.2021).

[14] Red Hat. ¿Qué es una API de REST? https://www.redhat.com/es/topics/api/what-is-a-rest-api. (último acceso: 3.07.2021).

[15] Spark Framework: an expressive web framework for Kotlin and Java. https://sparkjava.com/. (último acceso: 3.07.2021).

[16] Apache Software Foundation. What is Maven? https://maven.apache.org/what-is-maven.html. (último acceso: 3.07.2021).

[17] Federico Tomassetti. Unit Testing in Spark. https://sparkjava.com/tutorials/unit-testing. (último acceso: 3.07.2021).

[18] AssertJ - fluent assertions java library. https://assertj.github.io/doc/. (último acceso: 4.07.2021).

[19] Johan Haleby. REST-assured. https://github.com/rest-assured/rest-assured. (último acceso: 3.07.2021).

[20] Federico Tomassetti. Functional tests in Spark. https://sparkjava.com/tutorials/functional-testing. (último acceso: 4.07.2021).

[21] Tom Akehurst. WireMock. http://wiremock.org/. (último acceso: 4.07.2021).

[22] Carlos Macías Martín. Principios SOLID. https://enmilocalfunciona.io/principios-solid/. (último acceso: 3.07.2021).

[23] Franklin M. Gauer III. Implement CORS in Spark. https://sparkjava.com/tutorials/cors. (último acceso: 4.07.2021).

[24] Baeldung. Hashing a Password in Java. https://www.baeldung.com/java-password-hashing. (último acceso: 4.07.2021).

[25] Introduction to JSON Web Tokens. https://jwt.io/introduction. (último acceso: 4.07.2021).

[26] Serverless Stack. ¿Qué es Serverless? https://serverless-stack.com/chapters/es/what-is-serverless.html. (último acceso: 4.07.2021).

[27] Michael Diamond. Creating a Nohup Process in Java. https://stackoverflow.com/a/52395098. (último acceso: 4.07.2021).

[28] Wikipedia. Single-page application. https://es.wikipedia.org/wiki/Single-page_application. (último acceso: 6.07.2021).

[29] Vue Router. https://next.router.vuejs.org/. (último acceso: 6.07.2021).

[30] Vuex. https://next.vuex.vuejs.org/. (último acceso: 6.07.2021).

[31] Axios. https://axios-http.com/. (último acceso: 6.07.2021).

[32] Font Awesome Team. Font Awesome. https://fontawesome.com/. (último acceso: 6.07.2021).

[33] Apache Software Foundation. Apache Maven Assembly Plugin. https://maven.apache.org/plugins/maven-assembly-plugin/index.html. (último acceso: 6.07.2021).

[34] Benjamín Vera-Tudela. Arquitectura Lambda: Combinando lo mejor de dos mundos. https://sg.com.mx/revista/52/arquitectura-lambda-combinando-lo-mejor-dos-mundos. (último acceso: 11.07.2021).