

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

---

Tesi di Laurea

**PROGETTAZIONE E SVILUPPO DI  
UN'APPLICAZIONE WEB  
PER L'IDENTIFICAZIONE DI PRIMER OTTIMI  
PER L'AMPLIFICAZIONE**

Laureando:  
Alberto Gastaldello

Relatore:  
Prof. Nicola Ferro

Correlatori:  
Prof. Barbara Di Camillo  
Dr. Giacomo Baruzzo

Padova, 9 dicembre 2019  
Anno Accademico 2018/2019

---



*Alla mia Famiglia*



# Indice

<b>Capitolo 1</b>	<b>Introduzione</b>	<b>1</b>
<b>Capitolo 2</b>	<b>Background</b>	<b>3</b>
2.1.	<i>Tecnologie utilizzate</i>	3
2.1.1.	Endpoint	3
2.1.2.	Servizio web	4
2.1.3.	REST - <i>Representational State Transfer</i>	4
2.1.4.	API - <i>Application Programming Interface</i>	7
2.1.5.	Linguaggi lato server	7
2.1.6.	Linguaggi lato client	9
2.1.7.	Framework web	10
2.1.8.	Django	10
2.1.9.	Server web	13
2.1.10.	Proxy e reverse proxy	13
2.1.11.	nginx	13
2.1.12.	WSGI - <i>Web Server Gateway Interface</i>	14
2.1.13.	Gunicorn	14
2.1.14.	Celery	14
2.1.15.	Flower	15
2.1.16.	DBMS - <i>Database Management System</i>	15
2.1.17.	Schema ER	16
2.1.18.	Schema logico	16
2.1.19.	PostgreSQL	16
2.1.20.	Cache	17
2.1.21.	Redis	17
2.1.22.	Debian	17
2.1.23.	Virtualizzazione a livello di SO	18
2.2.	<i>Metagenomica e sequenziamento</i>	19
2.2.1.	Il microbiota	19
2.2.2.	Sequenziamento	21
2.2.3.	Reazione a catena della polimerasi e primer	22
2.2.4.	Design dei primer	26
2.2.5.	Automatizzazione del design	28
<b>Capitolo 3</b>	<b>mopo16S</b>	<b>29</b>
3.1.	<i>Score di ottimizzazione</i>	29
3.1.1.	Efficienza	29
3.1.2.	Coverage	31
3.1.3.	Matching-bias	31
3.2.	<i>Algoritmo</i>	32
3.3.	<i>Input e output</i>	34
3.3.1.	File di sequenze in input	34
3.3.2.	Parametri	35
3.3.3.	File di output	38
3.4.	<i>Dettagli tecnici</i>	39

<b>Capitolo 4</b>	<b>Analisi dei requisiti</b>	<b>41</b>
4.1.	Obiettivi	41
4.2.	Utenti e stakeholder	42
4.3.	Requisiti funzionali	42
4.3.1.	Gestione utenti	42
4.3.2.	Sequenze in input per mopo16S	42
4.3.3.	Primer in input per mopo16S	43
4.3.4.	Specifiche in comune per l'input	43
4.3.5.	Sottomissione del job	44
4.3.6.	Esecuzioni di mopo16S	44
4.3.7.	File di output da mopo16S	45
4.3.8.	Visualizzazione dei risultati	45
4.3.9.	API	45
4.4.	Requisiti non funzionali	46
<b>Capitolo 5</b>	<b>Progettazione</b>	<b>47</b>
5.1.	Progettazione concettuale	47
5.2.	Progettazione logica	51
5.3.	Architettura	53
<b>Capitolo 6</b>	<b>Implementazione</b>	<b>55</b>
6.1.	Componenti del sistema	55
6.2.	Codice	58
6.2.1.	Struttura modulare	58
6.2.2.	Applicazioni di terze parti	58
6.2.3.	Configurazioni	59
6.2.4.	Mappatura degli oggetti	60
6.2.5.	URL	61
6.2.6.	Template HTML	61
6.2.7.	Gestione utenti	62
6.2.8.	Caricamento delle sequenze e dei primer	62
6.2.9.	Creazione job	62
6.2.10.	Esecuzione job	62
6.2.11.	Visualizzazione delle risorse	63
6.2.12.	Visualizzazione dei risultati	63
6.2.13.	Web API	63
6.2.14.	Sorgenti e licenza	65
6.3.	Singularity e Docker	65
<b>Capitolo 7</b>	<b>Valutazione e collaudo</b>	<b>67</b>
7.1.	Ambiente di test	67
7.2.	File di test	68
7.3.	Demo e guida per l'utente	69
7.3.1.	Registrazione e login	69
7.3.2.	Caricamento sequenze	69

7.3.3. Caricamento primer .....	70
7.3.4. Creazione job .....	70
7.3.5. Visualizzazione e download risultati .....	71
7.3.6. API web .....	72
7.4. Risultati.....	73
<b>Capitolo 8 Conclusioni e lavoro futuro .....</b>	<b>75</b>
8.1. Conclusioni .....	75
8.2. Deployment .....	76
8.3. Possibili sviluppi.....	77
<b>Bibliografia .....</b>	<b>79</b>





# Capitolo 1 Introduzione

Il corpo umano è colonizzato da una grande varietà di microbi che formano comunità di batteri, virus ed eucarioti microbici specifici di ciascun ambiente anatomico. Poiché molti organismi non sono mai stati coltivati in modo indipendente, ogni comunità deve essere studiata nel suo insieme. Grazie a nuove tecniche di sequenziamento del DNA è possibile effettuare analisi e campionamenti sofisticati di questi sistemi complessi con metodi indipendenti dalla coltura.

Nel contesto della metagenomica uno degli strumenti chiave per lo studio della diversità microbica è il sequenziamento mirato di ampliconi del gene ribosomiale 16S mediante la reazione a catena della polimerasi. L'accuratezza di questo approccio dipende fortemente dalla scelta delle coppie di primer e, in particolare, dall'equilibrio tra efficienza, specificità e sensibilità nell'amplificazione delle diverse sequenze batteriche 16S presenti in un campione. Da questo scaturisce la necessità di avere metodi computazionali che permettano di progettare primer batterici 16S ottimali, in grado di tener conto delle conoscenze fornite dalle moderne tecnologie di sequenziamento.

*mopo16S* è un software che propone una soluzione capace di adempiere allo scopo, permette infatti di ottimizzare la scelta dei primer attraverso un nuovo algoritmo che considera diversi criteri per attribuire ad ogni coppia di primer un punteggio. Il software parte da una selezione iniziale di primer fornita in input, li modifica in passi successivi per tentare di migliorare la soluzione, infine restituisce il fronte di Pareto dei primer modificati.

Lo scopo del progetto presentato in questo documento è la realizzazione di un'applicazione web in grado di fornire alla comunità scientifica che studia questo ambito della genomica un'interfaccia di facile utilizzo per accedere alle funzionalità di *mopo16S*. In aggiunta, si propongono strumenti per la visualizzazione e l'analisi dei risultati ottenuti: la rappresentazione grafica consente infatti di interpretarli in modo più immediato ed efficace. All'interfaccia web viene poi affiancata un'interfaccia programmatica per permettere l'utilizzo del servizio con applicazioni di terze parti.

Nel documento viene approfondito il contesto della metagenomica con particolare attenzione sulla PCR e sulla funzione svolta dai primer durante questo processo, segue una descrizione del software mopo16S e del suo funzionamento. Si espongono quindi nel dettaglio funzionalità, progettazione e architettura dell'applicazione web, fornendo contestualmente la lista degli strumenti usati e alcune considerazioni riguardo alle scelte implementative.

Negli ultimi capitoli sono riportate le specifiche tecniche e le modalità di collaudo, con i relativi risultati ottenuti. Si è voluto inserire anche una serie di osservazioni a proposito degli sviluppi futuri e i benefici apportati dal servizio sviluppato, focalizzandosi in particolare modo sulle modifiche da apportare all'applicazione prima di collocarla in un ambiente di produzione.

## Capitolo 2 Background

Questo capitolo è dedicato a raccogliere tutte le informazioni utili alla comprensione dei concetti utilizzati nel resto del documento.

La sezione 2.1 presenta le tecnologie utilizzate nel progetto; espone in dettaglio ogni concetto, sistema o software coinvolto, dallo sviluppo dell'applicazione al deployment.

Nella sezione 2.2 vengono analizzati a fondo i concetti di base relativi alla metagenomica, utili a comprendere il campo di applicazione del software mopo16S (descritto nel capitolo 3) ed il funzionamento dell'applicazione web.

### 2.1. Tecnologie utilizzate

Questa sezione contiene tutte le informazioni necessarie a comprendere a fondo l'architettura del sistema progettato ed esposto nel presente documento.

Le sottosezioni definiscono concetti e principi seguiti nella progettazione dell'applicazione, oltre a protocolli, software e sistemi utilizzati per semplificare lo sviluppo e indispensabili per la messa in produzione del sistema completo.

#### 2.1.1. Endpoint

In ambito reti e comunicazioni il termine endpoint fa riferimento ad un dispositivo o servizio, anche denominato nodo, che accetta ed invia comunicazioni attraverso la rete LAN o WAN a cui è connesso.

Nel caso in esame useremo questo termine per riferirci al punto di connessione in cui sono esposte le risorse servite da un determinato web server in esecuzione su una macchina. L'endpoint di un servizio fornisce quindi le informazioni necessarie ad un client per raggiungere le risorse.

### 2.1.2. Servizio web

Il termine servizio web ha due significati principali [1, 2], uno generico e uno più specifico, di interesse per l'ambito del progetto.

La prima definizione lo descrive come un servizio offerto da un dispositivo elettronico ad un altro con comunicazione tramite *World Wide Web*. Nel nostro caso si parla di server in esecuzione su un dispositivo informatico che ascolta le richieste in una determinata porta su una rete, serve documenti web (ad esempio HTML, JSON, XML, immagini) ed espone tramite interfacce web servizi utili a risolvere specifici problemi.

### 2.1.3. REST - *Representational State Transfer*

REST definisce un insieme di principi architetturali per la progettazione di servizi web, permettendo di creare sistemi distribuiti facilmente scalabili e riutilizzabili. Esso descrive uno stile architetturale con delle linee guida, non rappresenta un sistema concreto e nemmeno uno standard, ma dei concetti che delineano l'organizzazione fondamentale di un sistema: le relazioni tra i componenti e con l'ambiente e le regole che ne governano la progettazione e l'evoluzione [1].

Le regole dello stile architetturale REST riducono i modi in cui il server può processare e rispondere alle richieste del client e sono tali per cui, se rispettate, conferiscono al sistema proprietà non funzionali come miglioramento di performance, scalabilità<sup>1</sup>, semplicità, modificabilità, visibilità, portabilità e affidabilità.

A partire dagli anni 2000, quando è stato introdotto nella tesi di dottorato di Roy Thomas Fielding [3, 4], l'approccio REST è stato ampiamente adottato dagli sviluppatori per la realizzazione di servizi web. Questo perché il web ha tutte le caratteristiche necessarie

---

<sup>1</sup> Scalabilità: proprietà di un sistema di gestire un numero crescente di lavoro aggiungendovi risorse

per essere considerato una piattaforma di elaborazione distribuita secondo i principi REST, senza la necessità di altre sovrastrutture, bastano infatti i suoi concetti basilari:

- URI, meccanismo per individuare risorse in una rete;
- HTTP, protocollo per richiedere una risorsa ad una macchina;
- un linguaggio per la rappresentazione dei contenuti, come ad esempio HTML, XML o JSON.

Questa visione si contrappone all'approccio dei servizi web basati su SOAP<sup>2</sup> che è un protocollo per lo scambio di dati strutturati basato sul concetto di chiamata remota.

I principi che rendono un sistema RESTful sono i seguenti:

- Il concetto fondamentale sul quale si basa un servizio web RESTful è l'esistenza delle risorse. Per risorsa si intende un qualsiasi elemento oggetto di elaborazione e fonte di informazioni come ad esempio un libro, un articolo, un file o un qualsiasi oggetto su cui è possibile effettuare operazioni. Ciascuna risorsa deve essere identificata univocamente, ed essendo in ambito web, il meccanismo più naturale per individuare una risorsa è dato dall'URI. Nell'applicazione realizzata un esempio di URI è <endpoint>/primers/18, che identifica un set di primer caricato nel sistema. Le risorse sono concettualmente separate dalla loro rappresentazione che viene restituita al client, questa infatti solitamente consiste in un formato non coincidente con la risorsa interna al server.
- Architettura client-server: client e server hanno compiti diversi e sono connessi tramite un insieme di interfacce. Separare i ruoli dell'interfaccia utente da quelli dello spazio di archiviazione migliora la portabilità in piattaforme diverse, dal momento che il client non deve salvare le informazioni condivise dal server, e la scalabilità, semplificando le componenti server che non devono occuparsi dell'interfaccia grafica o dello stato del client. La conseguenza vantaggiosa è che

---

<sup>2</sup> <https://www.w3.org/TR/soap12-part1>

server e client possono essere sostituiti e sviluppati indipendentemente fintantoché l'interfaccia non viene modificata.

- **Comunicazione senza stato (stateless):** stabilisce che il server non memorizza dati di sessione client e implica che ciascuna richiesta non deve avere alcuna relazione con le richieste precedenti e successive. Ciò non significa che le applicazioni non possono avere uno stato, ma che il server registra e gestisce solo lo stato delle risorse da lui esposte; nel caso sia necessaria la presenza di dati specifici della sessione, questi devono essere conservati e gestiti dal client e, se necessario, trasferiti al server in ogni richiesta. La principale ragione di tale scelta è la scalabilità: un livello di servizio che non deve mantenere sessioni client è molto più semplice da ridimensionare, mantenere lo stato di una sessione ha infatti un costo in termini di risorse lato server e all'aumentare del numero di client tale costo può diventare insostenibile. Inoltre, con una comunicazione senza stato un client può ricevere risposta da un cluster di server, dal momento che questi non hanno vincoli sulla sessione corrente, ottimizzando le prestazioni globali dell'applicazione.
- **Cacheability:** le risposte del server devono definire, implicitamente o esplicitamente, se il client può salvarle nella memoria cache (vedi sez. 2.1.20), in modo da essere recuperate e usate successivamente, evitando di dover fare una nuova richiesta al server. Se gestito correttamente lo sfruttamento della cache migliora scalabilità e performance, diminuendo il numero di comunicazioni tra client e server, e assicurando che il client usi sempre dati validi e aggiornati.
- **Stratificazione del sistema:** principio per cui il client non sa se è connesso e sta comunicando direttamente ad un server di livello più basso o a uno di livello intermedio. Questo comporta che la presenza di un proxy (vedi sez. 2.1.10) o un bilanciatore di carico tra il client e il server non influisca sulla loro capacità di comunicare e non sia necessario aggiornare il loro codice, ma anzi la loro migliori la scalabilità del sistema, con bilanciamento del carico di richieste tra i server o con cache distribuite. La stratificazione permette a un server di chiamare

ulteriori altri server per generare la risposta ad un client. Il principio dà anche la possibilità di migliorare la sicurezza del sistema, permettendo di aggiungerla come strato superficiale e separato da quello di lavoro.

- Uniformità dell'interfaccia: semplifica e scompone in pezzi l'architettura, permettendo a ogni parte di svilupparsi in modo indipendente. Se il client è in possesso di un URI che punta ad un servizio, conosce di conseguenza anche quali metodi sono disponibili su quella risorsa. Grazie al protocollo HTTP infatti la risposta del server contiene sufficienti informazioni con le quali il client può individuare il metodo da chiamare per estrarre i dati, modificare o eliminare la risorsa nel server in cui questa è salvata.

#### **2.1.4. API - *Application Programming Interface***

Un'API è un protocollo di comunicazione tra un client e un server che definisce le modalità di realizzazione e integrazione degli applicativi in modo tale che possano comunicare tra loro senza sapere i dettagli specifici dell'implementazione. Le API semplificano sviluppo e utilizzo, offrendo flessibilità e opportunità di innovazione. Solitamente i software espongono un'interfaccia con un set di funzioni che permettono di condividere i dati con i clienti o utenti esterni mantenendo alti i livelli di sicurezza e di controllo delle risorse, inoltre facilitano lo sviluppo di app nativamente cloud.

In ambito web è possibile e molto diffuso lo sviluppo di interfacce alle quali è possibile accedere tramite il protocollo HTTP, queste vengono chiamate API web.

#### **2.1.5. Linguaggi lato server**

##### **2.1.5.1. Python**

Python<sup>3</sup> è un linguaggio di programmazione ad alto livello orientato agli oggetti molto utile per lo scripting, la computazione scientifica, la grafica 3D e, in particolar modo, lo sviluppo di applicazioni web [5].

---

<sup>3</sup> <https://www.python.org>

La prima versione di Python (0.9.0), pubblicata nel 1991 da Guido van Rossum, era già orientata agli oggetti e includeva la gestione delle eccezioni, le funzioni e i tipi di dati base come stringhe, liste e dizionari. La versione 1.0 è stata rilasciata nel gennaio 1994, includeva gli strumenti di programmazione funzionale come le funzioni lambda, map e reduce. Le list comprehensions e il supporto a Unicode sono stati introdotti con Python 2.0 nel 2000, che presenta anche un garbage collector completo.

L'attuale versione principale, Python 3, ha portato a modifiche al funzionamento di oggetti come dizionari e stringhe; sono state rimosse molte funzionalità e la libreria standard è stata riorganizzata rimuovendo costrutti e moduli duplicati.

Per il progetto è stata usata la versione di Python più recente attualmente disponibile, ovvero la 3.8 [6].

#### 2.1.5.2. SQL - *Structured Query Language*

SQL è un linguaggio nato come strumento per interagire con i database relazionali utilizzando come riferimento il modello relazionale di Edgar F. Codd [7, 8]. Si basa sull'algebra relazionale e si può suddividere in quattro sezioni o sotto-linguaggi:

- DDL per la definizione dei dati (create schema, ...);
- DML per la loro manipolazione (insert, update, delete);
- DQL per l'interrogazione (select);
- DCL per la gestione degli strumenti di controllo e accesso ai dati.

SQL è diventato uno standard ANSI<sup>4</sup> nel 1986, e ISO<sup>5</sup> un anno dopo [9], fino ad oggi è stato rivisto ed esteso con molteplici funzionalità. Lo standard attualmente in uso è quello del 2019 [10], tuttavia le diverse implementazioni del linguaggio non sempre sono completamente compatibili e portabili tra i diversi DBMS esistenti, questo è causato

---

<sup>4</sup> <https://www.ansi.org>

<sup>5</sup> <https://www.iso.org>



anche dalla complessità raggiunta. PostgreSQL (vedi sez. 2.1.19) e Mimer SQL<sup>6</sup> sono i sistemi di gestione di database che seguono gli standard più di tutti nel mercato.

### 2.1.6. Linguaggi lato client

La maggior parte delle applicazioni web si basano sui seguenti tre linguaggi per la presentazione delle risorse e del loro contenuto all'utente o, più in generale, al client.

#### 2.1.6.1. HTML - *HyperText Markup Language*

L'HTML [11] è un linguaggio di markup reso pubblico per la prima volta nel 1993, nato per formattare, impaginare e visualizzare documenti ipertestuali e altre risorse. Negli anni è stato aggiornato ed esteso con diverse versioni (HTML4 [12], XHTML [13], HTML5 [14]) ma di fatto rimane lo standard di base, mantenuto dal *World Wide Web Consortium* (W3C)<sup>7</sup>, per definire la struttura delle pagine web.

Per l'annotazione di testo e altri contenuti come ad esempio link, immagini e tabelle, utilizza marcature chiamate *tag*, che racchiudono l'elemento da impaginare. I tag possono essere di due tipi: puntuali o applicati ad una sezione. Un esempio per il primo caso può essere il tag per inserire un'immagine in una determinata posizione del documento `<img>`, questo tipo di tag non richiede la chiusura. Nel secondo caso invece l'elemento da visualizzare deve essere preceduto da un tag di apertura e seguito da uno di chiusura, un esempio auto esplicativo può essere `<i>questo testo sarà in corsivo</i>`.

#### 2.1.6.2. CSS - *Cascading Style Sheets*

Il CSS<sup>8,9</sup> è un linguaggio utilizzato per definire l'aspetto delle pagine web, vengono solitamente definiti diversi fogli stile contenenti una lista di regole che descrivono come gli elementi devono essere presentati a video. Alcuni esempi possono essere il font

---

<sup>6</sup> <https://www.mimer.com>

<sup>7</sup> <https://www.w3.org>

<sup>8</sup> <https://www.w3.org/Style/CSS>

<sup>9</sup> <https://www.w3.org/standards/webdesign/htmlcss#whatcss>

utilizzato, i colori, le dimensioni e la spaziatura dei contenuti, l'aggiunta di decorazioni e animazioni. Anche CSS è aperto e standardizzato in base alle specifiche del W3C [15].

#### 2.1.6.3. JavaScript

JavaScript è un linguaggio di programmazione di alto livello orientato agli oggetti e conforme allo standard ECMAScript [16]; il codice viene compilato just-in-time, ovvero al momento dell'esecuzione. JavaScript è stato pubblicato per la prima volta nel 1995 [17] e, assieme a HTML e CSS, costituisce una tecnologia essenziale nel campo delle applicazioni web, l'utilizzo principale di questo linguaggio di scripting consiste infatti nel rendere interattive le pagine web. Oggigiorno la quasi totalità dei browser dispone di un motore JavaScript dedicato ad eseguire il codice che viene trasferito dal server web al client e permette di modificare o aumentare temporaneamente le funzionalità di una pagina. Una delle caratteristiche fondamentali è l'esecuzione di funzioni in base agli eventi, ad esempio lo spostamento del mouse o il click.

#### 2.1.7. Framework web

Un framework web è un software progettato per alleggerire e semplificare lo sviluppo di siti dinamici, servizi web e applicazioni simili [18]. Esso permette di dedicare gli sforzi allo sviluppo dell'applicazione evitando di doversi occupare di ambiti più complessi, come ad esempio sicurezza e scalabilità. È progettato per integrare le librerie e funzionalità di utilizzo comune in questo campo di applicazione, alcuni esempi possono essere l'accesso ai database, la creazione di template HTML o la gestione delle sessioni utente, promuovendo così il riutilizzo del codice.

#### 2.1.8. Django

Django è un framework web scritto in Python gestito da un'organizzazione senza fini di lucro che l'ha reso gratuito e open source. La versione attuale è la 2.2 LTS e verrà supportata fino al 2022 [19]. L'obiettivo primario del software è facilitare la creazione di siti e applicazioni web, scopo che viene raggiunto seguendo il modello architettonico

model-view-controller (MVC)<sup>10</sup> in cui vengono separati i dati, la logica e la presentazione (vedi Figura 1) [20].

In Django la nomenclatura è leggermente diversa (model-template-view), ma la struttura è la medesima. La parte relativa ai dati è implementata con un mappatore relazionale di oggetti (ORM) in grado di fare da tramite tra il database e le classi definite nel codice Python. Le view sono oggetti che generano le risposte HTTP richiamando un sistema di template web, questo permette di

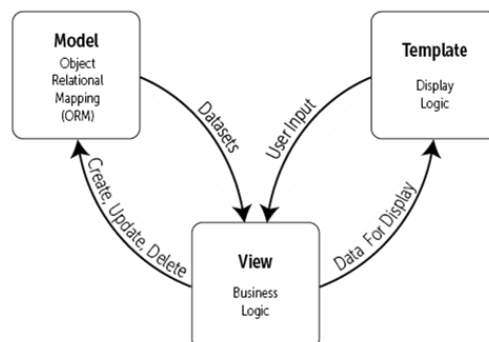


Figura 1 - Django model-template-view

Fonte: <https://djangobook.com/mdj2-django-structure>

generare HTML in modo dinamico. Un template contiene le parti statiche dell'output HTML desiderato, inoltre definisce le modalità di inserimento dei contenuti dinamici con alcune sintassi speciali<sup>11</sup>. La parte del controller risiede in un dispatcher di URL basato su espressioni regolari e percorsi, esso permette di definire la struttura degli indirizzi dell'applicazione.

Gli strumenti offerti da Django sono molteplici, e grazie alla sua architettura modulare è possibile selezionare ed utilizzare quelli necessari allo sviluppo della propria applicazione. Alcuni di questi strumenti sono il server di sviluppo, il sistema di template già citato, i moduli relativi all'interfaccia di amministrazione e il sistema di autenticazione. Tale organizzazione permette di importare con estrema facilità anche applicazioni Django esterne che seguono il principio di sviluppo di applicazioni riutilizzabili<sup>12</sup>.

Di seguito sono descritti alcuni componenti che sono stati utilizzati nel progetto.

<sup>10</sup> <https://www.codecademy.com/articles/mvc>

<sup>11</sup> <https://docs.djangoproject.com/en/2.2/topics/templates>

<sup>12</sup> <https://docs.djangoproject.com/en/2.2/intro/reusable-apps>

#### 2.1.8.1. Development server

Django dispone di un server web progettato appositamente per essere usato solamente durante lo sviluppo e i test al fine di semplificare il processo<sup>13</sup>. Non avendo bisogno dei livelli di sicurezza e stabilità necessari in ambiente di produzione, il server è stato sviluppato per essere leggero, autonomo e facile da usare; in più, non è necessario riavviarlo quando si apportano modifiche al codice, perché è in grado di rilevarle in automatico, riavviandosi in autonomia all'occorrenza. Questo strumento, usato assieme all'opzione DEBUG<sup>14</sup>, permette di controllare lo stato dell'applicazione in ogni momento, dalla console e direttamente dal browser con tutti i dettagli e gli errori di esecuzione.

#### 2.1.8.2. Django REST framework

Django REST framework<sup>15</sup> è un toolkit che facilita la creazione di API Web per la propria applicazione basata su Django. Tra le caratteristiche più utili si trovano i metodi di autenticazione ed una semplice interfaccia grafica per visualizzare le risposte in un formato velocemente leggibile, ad esempio i JSON vengono formattati con l'indentazione corretta.

#### 2.1.8.3. Django CSP

La Content Security Policy (CSP)<sup>16</sup> è uno standard di sicurezza aggiuntivo particolarmente importante per i siti web che aiuta a ridurre il rischio di attacchi come il cross-site scripting (XSS) e quelli di iniezione di dati. Gli attacchi XSS ad esempio tentano di far eseguire script dannosi nel browser della vittima sfruttando il fatto che questo si fida del contenuto ricevuto dal server. La CSP consente agli amministratori dei server web di limitare i vettori tramite i quali possono avvenire questi attacchi, è sufficiente specificare i domini che il browser deve considerare come fonti valide di script eseguibili, ignorando tutti gli altri.

---

<sup>13</sup> <https://docs.djangoproject.com/en/2.2/intro/tutorial01/#the-development-server>

<sup>14</sup> <https://docs.djangoproject.com/en/2.2/ref/settings/#std:setting-DEBUG>

<sup>15</sup> <https://www.django-rest-framework.org>

<sup>16</sup> <https://www.w3.org/TR/CSP3>

In Django la CSP è normalmente impostata al massimo livello di sicurezza, ovvero non consente qualsiasi esecuzione di script. Per modificarla in modo semplice è possibile usare Django CSP<sup>17</sup>, un'estensione che permette di configurare le politiche da utilizzare per l'intestazione HTTP Content-Security-Policy nel proprio server web.

#### 2.1.9. Server web

Consiste in un software in grado di soddisfare le richieste di rete in entrata provenienti dai client remoti, servendo pagine web e altre risorse, quali immagini, fogli di stile e script. Un server web può, in generale, gestire uno o più siti web ed elaborare contemporaneamente più comunicazioni che avvengono tramite HTTP.

#### 2.1.10. Proxy e reverse proxy

Con proxy si indica un tipo di server web con la funzione di intermediario tra un client e un server che espone risorse. Nell'ambito dei sistemi distribuiti, l'utilizzo di un tale apparato permette di aumentare sicurezza e isolamento ai vari strati software [21].

I proxy offrono principalmente di tre tipologie di servizio: garantire l'anonimato nella navigazione web, esporre una cache per risorse HTTP, fungere da firewall<sup>18</sup>.

I reverse proxy hanno la funzione specifica di recuperare risorse da uno specifico server web e di riportarle al client che le ha richieste. Spesso i proxy di questo tipo sono utilizzati per proteggere le applicazioni e i framework, o per bilanciare il carico tra più server.

#### 2.1.11. nginx

Si tratta di un server web leggero ed efficiente con diverse funzioni, tra le quali quella di reverse proxy. nginx<sup>19</sup> è distribuito sotto licenza BSD<sup>20</sup> ed è compatibile con diversi sistemi operativi.

---

<sup>17</sup> <https://django-csp.readthedocs.io/en/latest>

<sup>18</sup> [https://www.cisco.com/c/it\\_it/products/security/firewalls/what-is-a-firewall.html](https://www.cisco.com/c/it_it/products/security/firewalls/what-is-a-firewall.html)

<sup>19</sup> <https://www.nginx.org>

<sup>20</sup> <http://www.lininfo.org/bsdlicense.html>

### 2.1.12. WSGI - *Web Server Gateway Interface*

WSGI<sup>21</sup> consiste in uno schema a basso livello che definisce specifiche convenzioni di chiamata usate dai server web che servono applicazioni sviluppate in Python.

L'interfaccia è suddivisa in due parti: quella lato applicazione che definisce ed espone oggetti richiamabili dall'esterno (funzioni, metodi, classi, istanze, ...), e quella lato server che chiama, ad ogni richiesta proveniente da un client, uno di questi oggetti secondo le regole definite dal server stesso [22].

### 2.1.13. Gunicorn

Gunicorn<sup>22</sup> è un server WSGI che implementa il lato server dell'interfaccia appena delineata. È progettato in modo da poter comunicare con la maggior parte dei server web, e per interagire con tutte le applicazioni sviluppate secondo l'interfaccia WSGI indipendentemente dagli strumenti con i quali sono state realizzate.

### 2.1.14. Celery

Celery<sup>23</sup> è un gestore di code, basato sullo scambio di messaggi, per l'esecuzione di task pianificati o in tempo reale. I task possono essere eseguiti in modo asincrono, ovvero in background, o in modo sincrono, attendendo la risposta.

Il software è open source e sviluppato in Python, anche grazie a questo si integra molto bene con Django. Oltre a ciò il protocollo di comunicazione supporta altri linguaggi ed esistono molteplici client.

Grazie alla sua architettura, Celery permette di distribuire il lavoro in thread o in macchine diverse grazie alla possibilità di coesistenza di worker multipli. I worker sono le unità in grado di processare i messaggi inviati dai client, i task sono le unità di lavoro, e la comunicazione tra client e worker avviene tramite messaggi scambiati servendosi di un broker.

---

<sup>21</sup> <https://www.python.org/dev/peps/pep-0333>

<sup>22</sup> <https://gunicorn.org>

<sup>23</sup> <http://www.celeryproject.org>

### 2.1.15. Flower

Flower<sup>24</sup> è uno strumento che permette di monitorare e amministrare i worker e le code di Celery, è accessibile tramite browser. Tramite la sua semplice ed efficace interfaccia grafica è possibile vedere i task in tempo reale, amministrare i worker e visualizzare statistiche riguardo alle code.

### 2.1.16. DBMS - *Database Management System*

Un DBMS è un software progettato per gestire una base di dati. Questo consiste nel definire e interrogare un database, e nel creare, modificare o eliminare i dati in esso contenuti garantendo sicurezza e integrità degli stessi [23, p. 64]. Per essere classificati come completi i DBMS devono anche supportare transazioni, concorrenza delle operazioni, accesso tramite autenticazione, metodi di backup e ripristino, metodi di accesso remoto, oltre a descrivere i dati servendosi di un catalogo o di un dizionario per i metadati, e garantire il rispetto dei vincoli sui dati [23, pp. 97-102].

Esistono diversi tipi di DBMS, e sono denominati in base al tipo di base di dati da gestire. I più comuni sono quelli relazionali, che implementano il modello relazionale definito da Edgar F. Codd [7, 8], ma con l'avvento dei *Big Data* si stanno diffondendo sempre più anche i NoSQL.

L'idea di Codd nasce dall'insoddisfazione riguardo al modello navigazionale dell'approccio CODASYL<sup>25</sup>. Codd ha descritto un nuovo sistema per l'archiviazione e l'utilizzo di database di grandi dimensioni nel quale anziché archiviare i record in un elenco in formato libero, si utilizzano tabelle di record, ognuna adibita ad un tipo di entità. Il modello relazionale ha risolto l'inefficienza dei modelli di archiviazione sparsi suddividendo i dati in una serie di relazioni normalizzate, nelle quali gli elementi opzionali vengono spostati in tabelle secondarie per occupare spazio solo se necessario.

---

<sup>24</sup> <https://github.com/mher/flower>

<sup>25</sup> <https://en.wikipedia.org/wiki/CODASYL>

### 2.1.17. Schema ER

Nell'ambito della progettazione di un database, il modello astratto entità-relazione definisce una struttura dati che può essere implementata in un database, in genere relazionale. Alcuni accenni all'idea erano presenti già nel 1975 [24], ma il modello ER è stato sviluppato e formulato da Peter Chen nel 1976 [25].

Il diagramma ER fornisce una rappresentazione grafica dello schema concettuale mettendo in evidenza le entità facenti parte del dominio di applicazione considerato e specifica le relazioni esistenti tra esse. Nella rappresentazione le entità sono raffigurate da rettangoli, le relazioni da rombi e gli attributi sono collegati all'entità o alla relazione che li contengono.

### 2.1.18. Schema logico

Lo schema logico, ideato per la prima volta nel 1975 dall'istituto americano per gli standard (ANSI)<sup>26</sup> [26], rappresenta la struttura astratta del dominio di applicazione usando il modello di dati di una specifica classe di DBMS e costituisce la base del modello fisico dei dati stessi. Nel caso di RDBMS, ad esempio, lo schema logico consiste in una struttura con tabelle e colonne relazionali.

### 2.1.19. PostgreSQL

PostgreSQL<sup>27</sup> è un ORDBMS, ovvero un sistema software in grado di gestire database relazionali con un orientamento agli oggetti. Questo significa che unisce gli approcci di un database relazionale con quelli della programmazione ad oggetti. È gratuito ed open source, risulta essere uno dei migliori DBMS in circolazione al momento. La versione attuale di PostgreSQL è la 12.1 e, come le precedenti, è molto ben documentata nel sito principale [27, 28].

Tra le funzionalità di maggior rilievo offerte da PostgreSQL troviamo il controllo multi-versione della concorrenza (MVCC)<sup>28</sup>, transazioni nidificate, lock sofisticati e replica

---

<sup>26</sup> <https://www.ansi.org>

<sup>27</sup> <https://www.postgresql.org>

<sup>28</sup> <https://vladmihalcea.com/how-does-mvcc-multi-version-concurrency-control-work>



asincrona. Queste funzioni gli permettono di essere competitivo rispetto alla maggior parte dei DBMS.

#### 2.1.20. **Cache**

In ambito informatico un sistema di *caching* è un componente software con lo scopo di memorizzare i dati in modo da servirli molto velocemente. I dati che vengono memorizzati in una cache sono solitamente il risultato di calcoli precedenti o la copia di dati memorizzati in altri sistemi che però vengono richiesti con alta frequenza.

L'area di memoria dedicata a questo tipo di sistemi è normalmente la RAM quindi la capacità è limitata, ma la velocità di accesso ai dati può essere ridotta di almeno un ordine di grandezza rispetto, per esempio, all'interrogazione di un DBMS.

#### 2.1.21. **Redis**

Redis<sup>29</sup> è un archivio di strutture dati in memoria basato sul concetto di associazione chiave-valore, collocandosi quindi nell'ambito NoSQL. Redis viene utilizzato come database, cache e broker di messaggi; le strutture dati che supporta sono molteplici, le più comuni sono stringhe, tabelle hash, liste e set.

L'ultima versione di Redis attualmente disponibile è la 5.0.7, rilasciata il 19 novembre 2019 [29]; nell'ottobre 2018 la versione principale 5.0 ha introdotto gli stream, ovvero un tipo di dato che rappresenta un flusso continuo di informazioni, come ad esempio i log. Il software è gratuito ed open source ed è utilizzabile con un'ampia gamma di linguaggi di programmazione, grazie ai numerosi client disponibili online<sup>30</sup>.

#### 2.1.22. **Debian**

Debian è un sistema operativo basato su Linux rilasciato per la prima volta nel settembre del 1993<sup>31</sup>. Il progetto stabilisce come goal principale livelli molto alti di stabilità, la maggior parte delle distribuzioni Linux ha infatti una stabilità simile a quella che Debian

---

<sup>29</sup> <https://redis.io>

<sup>30</sup> <https://redis.io/clients>

<sup>31</sup> <http://www.ibiblio.org/pub/historic-linux/distributions/debian-0.91/ChangeLog>

possiede nel livello *testing* o *unstable*. Al fine di eliminare il più possibile errori e malfunzionamenti, i test sui pacchetti vengono eseguiti in modo ossessivo rendendo il sistema operativo molto famoso nella comunità open source. Questa politica rallenta il ciclo degli aggiornamenti ma permette comunque di risolvere eventuali problemi di sicurezza.

### 2.1.23. Virtualizzazione a livello di SO

Più comunemente chiamata containerizzazione, la virtualizzazione a livello di sistema operativo è una tecnologia che permette di astrarre componenti hardware e software dalla macchina fisica creando partizioni isolate. Sono quindi il kernel e il SO che forniscono le funzionalità di virtualizzazione, non più il layer hypervisor delle tradizionali tecniche di virtualizzazione.

Nell'ottica delle architetture a microservizi, i container costituiscono una tecnica molto efficace per impacchettare singolarmente ogni microservizio e permetterne la distribuzione e l'esecuzione in un ambiente isolato dal resto. Questo apporta miglione di sicurezza, affidabilità e stabilità all'intero sistema poiché in caso di problemi ad un servizio gli altri non ne risentono, essendone completamente scollegati. La portabilità è un altro aspetto fondamentale di questa architettura, i container sono infatti standardizzati in maniera da essere svincolati dalla specifica macchina o da un SO definito, permettendone così l'installazione e l'esecuzione in un hardware diverso. Procedendo in questa direzione i vantaggi sono molteplici, il più importante è la diminuzione della quantità di macchine necessarie, che riduce di conseguenza anche i costi; un altro beneficio consiste nella gestione semplificata delle versioni dei software. Solitamente le applicazioni che vengono installate in una macchina unica senza le dovute accortezze creano, durante il loro ciclo di vita, problemi di dipendenze a causa di aggiornamenti ai pacchetti; la containerizzazione permette di eliminare completamente questo scenario grazie all'isolamento.

Singularity e Docker sono due software che si avvalgono del kernel di Linux e delle sue funzionalità per esporre le capacità della tecnologia appena descritta.

## 2.2. Metagenomica e sequenziamento

In questa parte del capitolo vengono introdotti i concetti di microbiota e microbioma (2.2.1), assieme alla loro importanza nello studio della salute umana. Nella sezione 2.2.2 si espone il processo del sequenziamento del DNA, nella 2.2.3 viene approfondita una delle relative tecniche utilizzate, la reazione a catena della polimerasi (PCR), e congiuntamente viene definito il primer come suo innesco. Si descrivono infine nella sezione 2.2.4 l'importanza della progettazione dei primer per migliorare i risultati della PCR e nella 2.2.5 la possibilità di automatizzare questo processo.

### 2.2.1. Il microbiota

Il microbiota umano è formato da microorganismi simbiotici che convivono con l'organismo umano in maniera fisiologica o, a volte, patologica. Esso risiede nei tessuti e biofluidi umani: principalmente intestinali e gastrici, ma anche quelli della cavità orale, della pelle e del tratto urogenitale; comprende batteri, archeobatteri, funghi, protisti e virus.

Negli ultimi vent'anni, grazie allo sviluppo della metagenomica, l'interesse nei confronti di microbiota e microbioma è fortemente aumentato. Studi e ricerche [30, 31, 32] hanno mostrato che il

microbiota è fondamentale per l'omeostasi immunologica, ormonale e metabolica dell'ospite, ed hanno permesso di scoprire l'enorme legame esistente tra batteri e organismi pluricellulari.

Il termine microbioma descrive il patrimonio genetico posseduto dalla collettività del microbiota, alcuni dei geni in esso contenuti codificano per molecole non producibili in autonomia dal corpo umano. Il microbioma e l'ospite si sono evoluti nel tempo come un'unità sinergica, la componente genetica umana deriva infatti per il 99% dai batteri,

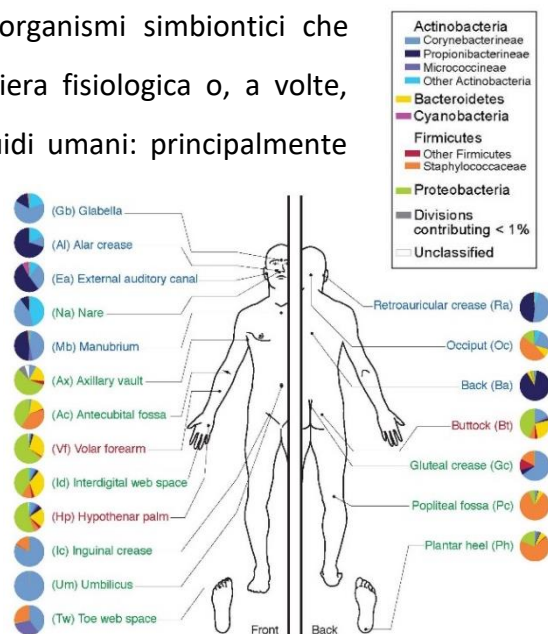


Figura 2 - I batteri predominanti sulla pelle umana  
Fonte: <http://www.genome.gov>

ed il microbiota può essere considerato come un organo supplementare, data l'integrazione che sostiene con il sistema e la sua capacità di fornire composti necessari al funzionamento degli altri organi del corpo umano.

Lo scopo fondamentale della ricerca sul microbioma umano consiste nel misurare la struttura e la dinamica delle comunità microbiche, i rapporti tra i loro membri, quali sono le sostanze prodotte e consumate, l'interazione con l'ospite e le differenze tra ospiti sani e malati. Gli studi che indagano sui rapporti tra microbiota e organismo ospitante sono principalmente di tipo correlativo o causale.

Si osservano due stati in cui il microbiota umano può trovarsi: eubiosi, in cui vengono prodotti i metaboliti necessari al corpo umano con effetti positivi per la sua salute; disbiosi, in cui viene meno la codifica genica delle molecole utili, ma vengono metabolizzati anche composti dannosi.

Progetti come *Human Microbiome Project*<sup>32</sup> e *MetaHIT*<sup>33</sup>, hanno studiato e definito l'immensa variabilità esistente nel microbioma sia all'interno di un singolo soggetto umano sia tra soggetti diversi, e la sua associazione con fattori ambientali, come la dieta e lo stile di vita; si osservano differenze addirittura tra campioni prelevati in serie dallo stesso sito nella stessa persona. La maggior parte delle persone condivide, in quantità apprezzabili, solo alcune delle specie microbiche; le diversità tra gli individui sono solitamente molto più grandi delle differenze nel microbiota di un singolo individuo nel tempo.

Lo studio è reso difficile anche dalla complessità delle comunità microbiche, dal momento che potrebbero esserci centinaia di specie diverse; inoltre non è possibile elencare, tramite tecniche microbiologiche standard, quali organismi sono presenti, poiché potrebbero richiedere condizioni di crescita speciali sconosciute e quindi non riproducibili in coltura. L'abbondanza di alcuni microbi può prevalere di ordini di

---

<sup>32</sup> <http://commonfund.nih.gov/hmp>

<sup>33</sup> <http://www.metahit.eu>

grandezza rispetto ad altri meno abbondanti, ed è quindi necessario un campionamento profondo per rilevarli.

### 2.2.2. Sequenziamento

Le prime acquisizioni di dati sul microbiota sono iniziate circa 25 anni fa; si basavano sul sequenziamento mirato dei geni ribosomiali RNA 5S e 16S, che differiscono per ciascuna specie e costituiscono un efficace identificatore. Lo sviluppo dei primi progetti di metagenomica su larga scala è cominciato un decennio dopo, con l'avvento delle nuove tecnologie di sequenziamento (NGS)<sup>34</sup>, e sta ampliando la conoscenza in materia.

Il sequenziamento del DNA è il processo atto a determinare la sequenza degli acidi nucleici, ovvero l'ordine delle quattro basi (adenina, guanina, citosina e timina). Le tecnologie NGS, che hanno la capacità di sequenziare in parallelo milioni di frammenti di DNA, hanno consentito analisi più approfondite sia per quanto riguarda il sequenziamento mirato del gene ribosomiale RNA 16S (processo indicato di seguito come seq-16S), che per lo *shotgun sequencing*<sup>35</sup>, così da estenderle a tutto il microbioma, accelerando la ricerca e la scoperta biologica e medica.

Il seq-16S è attualmente la strategia prevalente nella microbiologia per l'identificazione e la quantificazione della popolazione batterica poiché permette di considerare adeguatamente anche gli organismi meno presenti nel microbiota. Questo metodo veniva inizialmente utilizzato per identificare i batteri ma si è scoperto che è in grado di classificarli più dettagliatamente in generi, distinguendo anche quelle specie che non erano mai state coltivate con esito positivo. L'rRNA 16S è un gene dell'RNA presente in tutti i procarioti che serve a produrre i ribosomi responsabili della sintesi proteica, identificandolo si riesce a risalire alla singola specie batterica. Il gene 16S contiene sia regioni a rapida evoluzione (o ipervariabili), che forniscono sequenze di firma specifiche della specie e sono quindi utili per identificare e classificare organismi a diversi livelli

---

<sup>34</sup> <https://www.izsvenezie.it/temi/tecnologia-innovazione/next-generation-sequencing>

<sup>35</sup> <https://www.sciencedirect.com/topics/neuroscience/shotgun-sequencing>

tassonomici<sup>36</sup>, sia regioni a lenta evoluzione (o altamente conservate) che sono caratteristiche della specie. La definizione di una specie batterica è complicata dalla mancanza di conoscenza del genoma di tutte le specie batteriche. Pertanto viene spesso

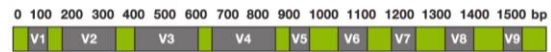


Figura 3  
In verde: regioni altamente conservate;  
in grigio: regioni ipervariabili.

Fonte: alimetrics.com

considerata, invece della specie, un raggruppamento di organismi aventi sequenze 16S con almeno il 97% di identità, detta unità tassonomica operativa (OTU)<sup>37</sup>. Gli OTU sono cluster di sequenze del gene 16S simili, ciascuno di essi rappresenta un'unità tassonomica di una specie, o di un genere di batteri a seconda della soglia di somiglianza della sequenza. Per separazioni più accurate si utilizzano soglie del 98% o del 99%.

Le regioni a lenta evoluzione del gene, che sono quindi molto simili per specie diverse, vengono utilizzate come obiettivi per progettare coppie di primer (gli inneschi) ad ampio spettro per la reazione a catena della polimerasi (PCR<sup>38</sup>), che a sua volta può essere utilizzata per isolare le regioni variabili.

### 2.2.3. Reazione a catena della polimerasi e primer

La PCR (*Polimerase Chain Reaction*) è una tecnica usata in biologia molecolare per l'amplificazione, ovvero la moltiplicazione, di frammenti di acidi nucleici dei quali si conoscono le sequenze nucleotidiche iniziali e terminali. L'amplificazione mediante PCR è un processo esponenziale che consente di generare in vitro molto rapidamente milioni di copie del segmento di DNA, in modo da avere una quantità di materiale genetico sufficiente per le successive applicazioni. Questa operazione è possibile grazie agli enzimi DNA-polimerasi di batteri resistenti ad alte temperature che svolgono la duplicazione cellulare, ovvero la ricostruzione di un segmento di DNA partendo da un filamento a elica singola, disponendo i nucleotidi in modo complementare.

<sup>36</sup> <https://basicbiology.net/biology-101/taxonomy>

<sup>37</sup> <http://www.metagenomics.wiki/pdf/definition/operational-taxonomic-unit-otu>

<sup>38</sup> <https://www.khanacademy.org/science/biology/biotech-dna-technology/dna-sequencing-pcr-electrophoresis/a/polymerase-chain-reaction-pcr>

Per avviare il processo della PCR è necessario disporre, in soluzione, dei seguenti elementi:

- il segmento di DNA che si desidera riprodurre;
- nucleotidi liberi in quantità opportuna;
- primer adeguati;
- una DNA-polimerasi termo-resistente;
- un Buffer, necessario a costituire l'ambiente adatto alla reazione;
- altri elementi di supporto indispensabili per il corretto funzionamento della DNA-polimerasi;
- acqua per portare a volume la soluzione.

Un primer è un acido nucleico a singolo filamento di lunghezza ridotta necessario a tutti gli organismi viventi per iniziare la sintesi del DNA. Le DNA polimerasi sono in grado di aggiungere nucleotidi solamente all'estremità 3' di un acido nucleico esistente, questo processo richiede che un primer sia legato al modello per l'inizio della sintesi di un filamento complementare.

È utile definire il concetto di direzionalità, che si riferisce all'orientamento estremità-estremità di un singolo filamento di acido nucleico. Per convenzione le estremità vengono chiamate 5' (cinque primo) e 3' (tre primo), in base al gruppo carbonio legato alla base. Le polimerasi scrivono in direzione 5' -> 3', l'unica modalità di sintesi possibile negli organismi viventi.

Per la PCR sono necessarie coppie di primer personalizzate per dirigere la sintesi del DNA nelle due direzioni, l'una verso l'altra, dalle estremità opposte della sequenza da amplificare. Una coppia di primer consiste in un *forward* (3' -> 5') ed un *reverse* (5' -> 3'), ciascuno

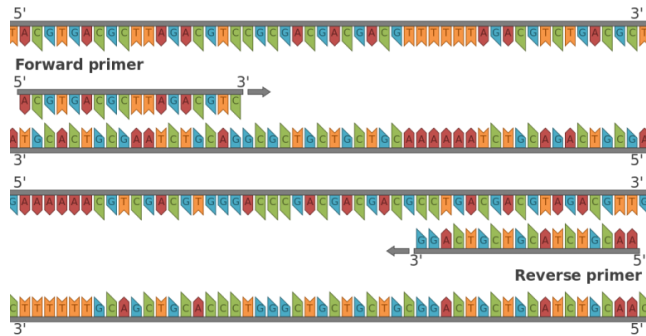


Figura 4 - Forward e reverse primer

Fonte: <https://commons.wikimedia.org/w/index.php?curid=26867486>

ha lo scopo di abbinarsi ad una delle due direzioni. Questi primer devono codificare solo i siti di interesse, infatti un primer che può legarsi a più regioni lungo il DNA le amplificherebbe tutte, vanificando lo scopo del processo.

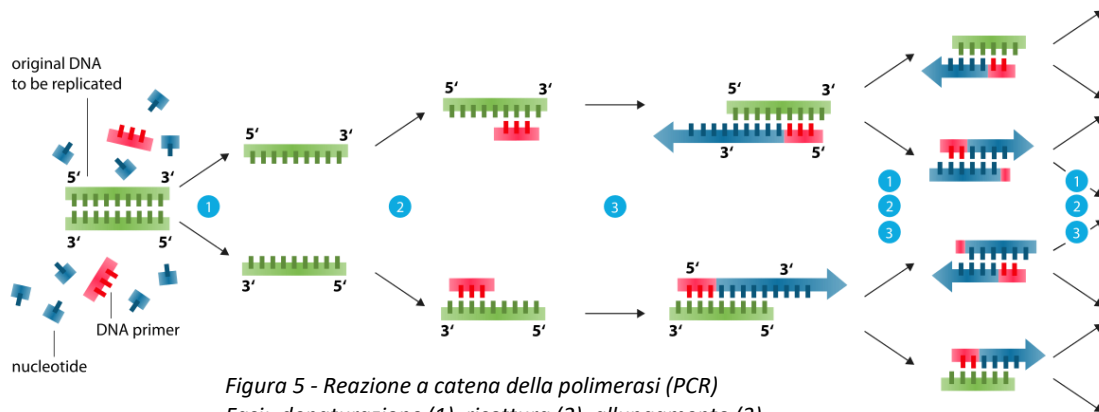


Figura 5 - Reazione a catena della polimerasi (PCR)

Fasi: denaturazione (1), ricottura (2), allungamento (3)

Fonte: [https://commons.wikimedia.org/wiki/File:Polymerase\\_chain\\_reaction.svg](https://commons.wikimedia.org/wiki/File:Polymerase_chain_reaction.svg)

La PCR consiste in una serie di cicli termici (solitamente tra i 20 e i 40), di seguito viene schematizzato il procedimento con alcuni dettagli sulle varie fasi. I passaggi comuni alla maggior parte dei metodi PCR sono i seguenti:

- Inizializzazione (non sempre necessaria): il ciclo può essere preceduto da un passaggio in cui la camera di reazione viene portata ad una temperatura di circa 94-98°C per 1-10 minuti, ciò è necessario solamente per le DNA polimerasi che richiedono l'attivazione per calore (chiamato avviamento a caldo).



- Denaturazione: consiste nel riscaldare la camera di reazione a 94–98°C per 20–30 secondi provocando la fusione della doppia elica, vengono rotti i legami a idrogeno e prodotte due molecole di DNA a filamento singolo.
- Ricottura: la temperatura viene abbassata a 50–65°C per 20–40 secondi permettendo ai primer (forward e reverse) di legarsi alle molecole nelle regioni loro complementari. Per questo passaggio è fondamentale determinare una temperatura adeguata poiché l'efficienza e la specificità ne sono fortemente influenzate. Deve essere abbastanza bassa da consentire l'ibridazione del primer al filamento, ma abbastanza alta da consentire l'ibridazione specifica, ovvero una corrispondenza complementare esatta con la parte del filamento a cui si lega; inoltre in caso di temperatura troppo alta il primer potrebbe non legarsi, se troppo bassa potrebbe legarsi in modo imperfetto. I legami a idrogeno tra le basi complementari sono stabili solo quando la sequenza dei primer è molto simile alla sequenza del filamento di DNA. Durante questa fase la polimerasi si lega alla doppia elica formatasi nel punto in cui il primer si è ibridato.
- Allungamento: la DNA polimerasi comincia la sintesi del nuovo filamento aggiungendo in modo complementare i nucleotidi presenti nella soluzione, lavorando in direzione 5' -> 3'. Alla loro temperatura ottimale, la maggior parte delle DNA polimerasi polimerizza migliaia di basi al minuto; ad ogni fase di allungamento il numero di sequenze target viene pressoché raddoppiato. Nel ciclo successivo i nuovi filamenti modello sono, oltre all'originale, anche tutti quelli appena generati; questo porta ad un'amplificazione esponenziale della regione target.
- Allungamento finale: passaggio singolo facoltativo, si porta la soluzione ad una temperatura di 70–74°C per 5-15 minuti, questo serve a garantire il completo allungamento dell'eventuale DNA a singolo filamento rimanente.
- Tenuta finale: la camera di reazione viene raffreddata a 4–15°C, i prodotti della PCR possono essere conservati in questo modo per un breve periodo.

Le tre fasi di denaturazione, ricottura e allungamento compongono un ciclo, che è necessario ripetere più volte al fine di produrre milioni di copie del segmento target, detto anche amplicone. Il processo è esponenziale, quindi ripetendo  $n$  volte il ciclo, il numero di copie formate sarà  $2^n$ . Tutte le temperature utilizzate nel processo e il periodo di tempo per il quale vengono applicate in ciascun ciclo dipendono da molteplici parametri, alcuni di essi sono: l'enzima utilizzato per la sintesi del DNA, la concentrazione di ioni bivalenti e nucleotidi nella soluzione, la temperatura di fusione dei primer.

Con il termine amplicone ci si riferisce ad un segmento di DNA che sia il prodotto di un processo di amplificazione.

#### **2.2.4. Design dei primer**

Per ottimizzare le condizioni di esecuzione della PCR sono state sviluppate alcune tecniche e procedure, tra le quali troviamo la progettazione ottimizzata dei primer. È fondamentale infatti, per migliorare gli ampliconi prodotti, focalizzare l'attenzione sul design di primer in modo tale che possano ibridarsi il più specificamente possibile alle regioni adiacenti al target da amplificare. Per farlo i primer devono essere complementari a queste regioni, ma in alcuni casi accade che non lo siano in modo assoluto, si possono quindi utilizzare primer degeneri oppure includere nei primer alcuni nucleotidi non complementari. Fondamentale anche la considerazione della temperatura di fusione, che dovrebbe essere condivisa nella coppia di primer e simile alla temperatura di ricottura della reazione; in caso contrario i primer potrebbero non ibridarsi o farlo in posizione errata.

Un primer viene chiamato degenero quando contiene nucleotidi diversi in posizioni definite. Questo tipo di primer consente di identificare geni di cui sia nota solo la sequenza proteica o geni omologhi tra diverse specie, e viene utilizzato per il seq-16S perché la sequenza genica 16S, pur non essendo identica nei vari organismi, presenta similitudini.

Una coppia di primer degeneri forward-reverse può essere esplosa in una coppia di insiemi di primer non degeneri, è sufficiente assegnare ai nucleotidi degeneri tutte le possibili combinazioni di valori rispettivamente per il forward e per il reverse.

Nel resto del documento ci riferiremo a questo tipo di collezione di sequenze come "coppia di set di primer" o "*primer-set-pair*", della quale viene rappresentato un esempio in Figura 6.

Forward primer degenero	Reverse primer degenero	Forward primer non degenero	Reverse primer non degenero
ACGTHACGT	RACGYACGT	ACGTAACGT	AACGTCACGT
		ACGTCACGT	AACGTTACGT
		ACGTTACGT	GACGTCACGT
			GACGTTACGT

Figura 6 - Coppia di set di primer (*primer-set-pair*)

*Nelle due colonne di sinistra le basi degeneri sono delineate in grassetto, a destra troviamo i due insiemi non degeneri creati dopo la loro espansione.*

Con "combinazioni" ci riferiremo ai possibili accostamenti forward-reverse

ricavabili dai set non degeneri (nel caso in Figura 6 risulterebbero 12 combinazioni); per indicare un insieme di coppie di primer (degeneri o meno a seconda del contesto) utilizzeremo invece "set di coppie di primer".

Per la rappresentazione delle sequenze viene usata la notazione IUPAC<sup>39</sup>, che oltre alle basi *A*, *C*, *G* e *T*, comprende anche tutte le basi degeneri; nella Figura 6 troviamo *H*, *R* e *Y*, che codificano rispettivamente *A – C – T*, *A – G* e *T – C*.

La progettazione dei primer necessita della valutazione delle seguenti proprietà fondamentali per ogni coppia di set di primer:

- Efficienza (e specificità), ovvero quanto ogni coppia è in grado di amplificare la regione target senza amplificarne altre; questo richiede il controllo di alcuni parametri importanti, ad esempio le lunghezze di primer e amplicone, il numero e la posizione delle incompatibilità, il contenuto-GC del primer, la possibilità di creare strutture secondarie per interazione tra primer e DNA.
- Copertura, il numero di diverse sequenze batteriche 16S abbinata ad almeno una coppia di primer presi dal primer-set-pair.

<sup>39</sup> <https://iupac.org/what-we-do/nomenclature>

- Matching-bias, misura del bias introdotto nell'analisi a causa delle differenti capacità di amplificazione dei vari primer.

### 2.2.5. Automatizzazione del design

La procedura di design ottimo dei primer è automatizzabile eseguendo simulazioni al computer con software creati ad hoc in grado di ricostruire i risultati teorici della PCR. Spesso questi software sono disponibili gratuitamente, alcuni utilizzabili direttamente online.

I calcoli sono basati sull'ottimizzazione di primer forniti in input come punto di partenza e su sequenze batteriche conosciute e catalogate. Sono indispensabili quindi:

- un set di primer studiati appositamente per lo scopo da raggiungere nel singolo esperimento;
- un set di sequenze batteriche a cui far riferimento.

Dal momento che gli studi sul gene ribosomiale RNA 16S sono molto diffusi, le sequenze sono state raccolte negli anni in diversi database, come Ribosomal Database Project<sup>40</sup>, GreenGenes<sup>41</sup>, SILVA<sup>42</sup>, EzBioCloud<sup>43</sup>, probeBase<sup>44</sup>. I database di sequenze sono utilizzabili come fonte di per i software di ottimizzazione.

---

<sup>40</sup> <https://rdp.cme.msu.edu>

<sup>41</sup> <http://greengenes.secondgenome.com>

<sup>42</sup> <https://www.arb-silva.de>

<sup>43</sup> <https://www.ezbiocloud.net>

<sup>44</sup> <http://probebase.csb.univie.ac.at>

## Capitolo 3 mopo16S

Tra i software di ottimizzazione per il design di primer ottimi per la PCR, nell'ambito degli studi sul gene rRNA 16S, troviamo mopo16S<sup>45</sup> (*Multi-Objective Primer Optimization for 16s experiments*), che implementa un nuovo algoritmo [33] studiato per considerare contemporaneamente le tre proprietà descritte nella sezione 2.2.4.

Si descrivono in questo capitolo i principi di funzionamento del software in esame. Nella sezione 0 si espongono i vincoli che codificano l'ottimalità delle soluzioni e nella 3.2 si può trovare una breve descrizione dell'algoritmo di ottimizzazione. Le sezioni 3.3 e 3.4 infine contengono i dettagli tecnici utili alla progettazione e allo sviluppo dell'applicazione web, dai file e parametri che il software richiede in input, ai file che produce in output, illustrando anche alcune informazioni a proposito del codice.

### 3.1. Score di ottimizzazione

Di seguito vengono descritti nel dettaglio i vincoli che il software considera nel calcolo di efficienza, coverage e matching-bias. Ognuno di questi tre punteggi assume valori reali tra 0 e 10.

#### 3.1.1. Efficienza

Per quanto riguarda la valutazione dell'efficienza mopo16S considera dieci criteri di giudizio, ad ognuno di essi è associato un punteggio da 0 a 1, la loro somma fornisce il primo dei tre score di ottimizzazione. I primi sette termini valutano i primer presi

---

<sup>45</sup> <http://sysbiobig.dei.unipd.it/?q=Software#mopo16S>

singolarmente, gli altri tre forniscono un punteggio alle coppie di set di primer; di seguito l'elenco completo<sup>46</sup>:

- Temperatura di fusione;
- Contenuto GC: la frazione di basi Guanina e Citosina nel primer;
- Stabilità dell'estremità 3' (1): il contenuto delle ultime tre basi deve essere diverso da AAA o TTT;
- Stabilità dell'estremità 3' (2): nelle ultime cinque basi non devono esserci più di tre C o G;
- Omopolimeri: non devono esserci sequenze con più di 4-5 nucleotidi uguali;
- Auto-dimeri: non devono essere presenti regioni complementari ad altre regioni nel primer stesso, per evitare che due primer uguali si ibridino tra loro;
- Stem-loop (hairpin): non devono esserci regioni auto-complementari, soprattutto all'estremità 3', per evitare che il primer formi cicli con se stesso;
- Intervallo delle temperature di fusione: differenza tra il massimo e il minimo valore delle temperature di fusione tra tutte le combinazioni forward-reverse;
- Dimeri: considera il massimo numero di match in tutti gli allineamenti possibili tra le combinazioni.
- Intervallo di lunghezza degli ampliconi: dal momento che con l'aumentare della lunghezza degli ampliconi l'efficienza della PCR si riduce, si vogliono mantenere le lunghezze degli ampliconi generati in un intervallo ristretto. Questo per evitare sia di amplificare eccessivamente gli ampliconi molto più corti della lunghezza target, sia di penalizzare primer potenzialmente preziosi a causa della rarità di alcune sequenze. Viene quindi considerata la differenza tra la mediana e un

---

<sup>46</sup> [http://www.premierbiosoft.com/tech\\_notes/PCR\\_Primer\\_Design.html](http://www.premierbiosoft.com/tech_notes/PCR_Primer_Design.html)

quantile della lunghezza dell'amplicone tra tutti i possibili ampliconi ottenuti abbinando ogni combinazione di primer con le sequenze di riferimento.

Il software offre la possibilità all'utente di impostare le varie soglie e gli intervalli di tolleranza fuzzy a piacimento (vedi sezione 3.3.2).

### 3.1.2. Coverage

Le sequenze di un primer e di un gene 16S vengono considerate abbinate se esiste una sezione del 16S che corrisponde con esattezza al seme del primer, e con massimo due discrepanze al resto del primer; dove il seme è definito come gli ultimi cinque nucleotidi all'estremità 3'. Una sequenza 16S si considera coperta da una coppia di set di primer se le si abbina almeno una combinazione forward-reverse.

Viene inoltre considerata la lunghezza dell'amplicone, dato che l'efficienza della PCR diminuisce con essa. Il calcolo consiste nello sfavorire le sequenze 16S che differiscono di più di 100 nucleotidi in lunghezza dalla lunghezza dell'amplicone desiderata, quest'ultima è definita come la mediana delle lunghezze di tutti gli ampliconi che si ottengono abbinando tutte le combinazioni di primer forward-reverse del primer-set-pair con l'insieme delle sequenze 16S.

Il coverage fornisce il secondo punteggio per l'ottimizzazione.

### 3.1.3. Matching-bias

Una coppia di primer può potenzialmente amplificare più porzioni del gene 16S; questo si verifica soprattutto nei primer che subiscono l'ottimizzazione, nei quali vengono aggiunte, modificate o rimosse alcune basi.

Un altro caso riscontrato è quello in cui la coppia è in grado di coprire una porzione di 16S per una specie, ma non per un'altra. Dato che le regioni conservate non sono esattamente uguali tra specie, una specie potrebbe avere una regione conservata molto diversa da quella per cui il primer è progettato.

Per alcuni studi biologici è importante rilevare in che quantità le varie specie sono presenti nel campione analizzato, i casi descritti sopra introducono quindi un bias nell'analisi. Il matching-bias è il terzo ed ultimo punteggio di mopo16S che tenta di misurare questo bias.

Tiene conto della variabilità del numero di combinazioni di primer forward e reverse abbinate a ciascuna sequenza target. Si considera quindi la deviazione standard relativa alla copertura attraverso le sequenze 16S, pesata in base alla media del numero di combinazioni corrispondenti a ciascuna sequenza.

### 3.2. Algoritmo

L'algoritmo di ottimizzazione implementato in mopo16S è basato sull'ottimizzazione multi-obiettivo<sup>47</sup> con una funzione di punteggio che, simultaneamente, minimizza il matching-bias e massimizza efficienza e coverage, gli score definiti nella sezione precedente. Lo spazio di ricerca è definito come l'insieme di tutte le possibili coppie di set di primer.

Questa modalità di calcolo del punteggio è necessaria perché i tre parametri sono contrastanti; non si è interessati ad un'unica soluzione ma all'insieme delle soluzioni ottimali di Pareto<sup>48</sup>, ovvero all'insieme di soluzioni per le quali il miglioramento di un parametro comporta il peggioramento di almeno un altro.

Il risultato dell'ottimizzazione multi-obiettivo consiste quindi in un insieme di primer-set-pair che non sono peggiori di qualsiasi altro e sono strettamente migliori secondo almeno uno dei parametri; nell'ottimizzazione a singolo obiettivo il risultato sarebbe stato contrariamente un'unica coppia di set di primer ottima. L'obiettivo dell'algoritmo è di determinare o approssimare questo insieme di ottimi paretiani; la loro immagine nello spazio tridimensionale considerato è chiamata fronte di Pareto.

La funzione obiettivo consiste in un vettore di funzioni  $f = (f_E, f_C, f_M)$  dove  $E$ ,  $C$  e  $M$  si riferiscono, rispettivamente, a efficienza, coverage e matching-bias. Le coppie di set di

---

<sup>47</sup> <https://www.sciencedirect.com/topics/computer-science/multi-objective-optimization>

<sup>48</sup> <https://www.sciencedirect.com/topics/computer-science/pareto-optimal-solution>



primer vengono valutate nel seguente modo: date  $p$  e  $p'$ ,  $p$  è una coppia migliore di  $p'$  (ovvero  $p < p'$ ) se e solo se sono verificate contemporaneamente le seguenti:

$$f(p) \neq f(p'), \quad f_E(p) \geq f_E(p'), \quad f_C(p) \geq f_C(p'), \quad f_M(p) \leq f_M(p')$$

Se non esiste alcuna  $p'$  tale che  $p' < p$  allora  $p$  viene considerata come ottimo paretoiano.

L'algoritmo usa l'approccio di ricerca locale iterata in due fasi: parte da una soluzione iniziale e la raffina iterativamente applicando cambiamenti minimi, rivalutando ogni volta la qualità della soluzione. Il processo si interrompe quando nessun ulteriore modifica locale è in grado di migliorare la soluzione. I passi vengono ripetuti partendo da punti diversi e viene restituita la migliore soluzione trovata complessivamente, come approssimazione dell'ottimo da ricercare.

L'estensione al caso multi-obiettivo di questo metodo di ricerca locale parte da una serie di soluzioni di Pareto iniziali, prende in modo casuale una soluzione dal fronte, e massimizza con la ricerca locale una combinazione lineare dei punteggi di ottimizzazione con pesi casuali  $\sum_{i=0}^2 \alpha_i = 1$ , infine aggiorna il fronte di Pareto e ripete la procedura fino a quando non si incontra una condizione di terminazione. Di fatto il problema multi-obiettivo è ridotto ad una sua rappresentazione scalare, necessaria per ridurre la complessità e a poter utilizzare una ricerca locale a singolo-obiettivo. A questo scopo gli score sono normalizzati e il matching-bias ( $M$ ) viene ridefinito come  $1 - M$  per essere massimizzato assieme agli altri due.

Durante l'ottimizzazione l'algoritmo tenta di trovare soluzioni migliori di quelle ricevute in input perturbandole, ovvero modificando singoli nucleotidi oppure aggiungendone uno all'inizio o alla fine, e ricominciando l'operazione dal nuovo ottimo trovato. Il ciclo termina quando non esistono più perturbazioni che migliorano la funzione obiettivo.

### 3.3. Input e output

Per l'esecuzione mopo16S richiede in input due file, e permette anche di specificare molteplici parametri opzionali per adattare le esecuzioni del software al campo di applicazione dell'esperimento di metagenomica che si sta svolgendo. Una volta terminata l'ottimizzazione mopo16S scrive i risultati in alcuni file di output.

In questa sezione vengono descritti il tipo di input richiesto, i parametri che si possono specificare da riga di comando, e la struttura dell'output.

#### 3.3.1. File di sequenze in input

L'ottimizzazione viene eseguita su un insieme di sequenze nucleotidiche di riferimento (i geni 16S) per le quali progettare l'insieme ottimale di primer. Esse sono rappresentate da stringhe di caratteri provenienti da un file che nel resto del documento nomineremo `rep_set`.

I tre punteggi sono calcolati sul secondo insieme di sequenze, i primer, letti dal file `good_pairs`. Quest'ultimo contiene coppie di primer (possibilmente degeneri) da cui iniziare l'ottimizzazione, salvate alternando il primer forward ed il corrispondente reverse.

Entrambi i file devono essere in formato FASTA<sup>49</sup> (Figura 7), un formato testuale comunemente utilizzato per rappresentare sequenze di nucleotidi o peptidi con la nomenclatura IUPAC [34, 35]. Una sequenza inizia con una descrizione a riga singola, che ha come primo carattere ">", seguita dalle righe contenenti i caratteri; le righe possono avere lunghezza massima di 80 caratteri.

```
>Descrizione sequenza 1
<sequenza 1>
>Descrizione sequenza 2
<sequenza 2>
...
```

```
>S-D-Bact-0008-d-S-20
AGAGTTTGATCMTGGCTCAG
>S-D-Bact-0785-a-A-19
CTACCAGGGTATCTAATCC
>S-D-Bact-0019-a-S-20
TGGCTCAGRWYGAACGCTRG
>S-D-Bact-0785-a-A-19
CTACCAGGGTATCTAATCC
...
```

*Figura 7 - Rappresentazione schematica del formato FASTA e relativo esempio*

<sup>49</sup> <https://zhanglab.ccmb.med.umich.edu/FASTA>

Per la creazione dei file necessari all'esecuzione di mopo16S occorre processare i dati con una modalità precisa; in questo documento utilizzeremo lo stesso esempio esposto nella descrizione di mopo16S.

Il primo passo consiste nello scaricare da GreenGenes, un database di riferimento 16S pubblico e non più aggiornato dal 2013, i dati organizzati in OTU. Si estraggono dall'archivio, per il livello di similarità tassonomica richiesto, il *rep\_set* e il file con le informazioni sulla tassonomia; quest'ultimo viene corretto con alcune procedure di riannotazione, ed infine si selezionano dal *rep\_set* solo le sequenze batteriche. Il processo di correzione della tassonomia è necessario perché i dati di GreenGenes non sono progettati per distinguere le sequenze di eucarioti o virus, e quindi permettere la selezione delle sole appartenenti al dominio dei batteri. L'insieme ottenuto è un sottoinsieme rappresentativo dell'intero database, sempre più accurato all'aumentare del livello di somiglianza selezionato, questo si riflette sul numero crescente di sequenze incluse.

I primer iniziali vengono scaricati da SILVA, un database di RNA ribosomiale, e successivamente processati selezionando solo il dominio richiesto in base a lunghezza del primer, lunghezza dell'amplicone e degenerazione.

I processi appena descritti sono automatizzati da due script disponibili con mopo16S.

### 3.3.2. Parametri

Di seguito si elencano tutti i parametri accettati in input da mopo16S accompagnati da una breve descrizione. Essi sono tutti opzionali, in caso di mancata specifica infatti assumono in automatico il valore impostato come predefinito all'interno del codice.

I valori di default risultano essere adeguati nella maggior parte dei casi, ma possono essere modificati in base ad eventuali esigenze sperimentali.

Opzioni generali:

- *seed*: seme del generatore di numeri casuale (intero lungo, default 0);
- *restarts*: numero di iterazioni dell'algoritmo (intero, default 20);

- runs: numero di esecuzioni dell'algoritmo (intero, default 20);
- outFileNames: prefisso per il nome dei file di output relativi ai primer ottimizzati (default "out");
- outInitFileName: prefisso per il nome dei file di output relativi ai primer forniti in input (default "init");
- threads: numero di thread per l'esecuzione parallela (intero, default 1);
- verbose: livello di verbosità (intero, default 0). Se "0", non vengono creati ulteriori output; se diverso da "0", per ogni run vengono creati 3 file: punteggi dei primer; sequenze dei primer; passi di ottimizzazione eseguiti ad ogni riavvio.

Opzioni relative al coverage:

- maxMismatches: numero massimo di mancate corrispondenze (mismatch) tra l'estremità non-3' del primer e una sequenza 16S per considerare quest'ultima coperta dal primer, nel caso in cui anche l'estremità 3' corrisponda perfettamente (intero, default 2);
- maxALenSpanC: intervallo massimo della lunghezza dell'amplicone da considerare durante il calcolo del coverage (intero, default 200).

Opzioni relative all'efficienza:

- minPrimerLen: lunghezza minima del primer (intero, default 17);
- maxPrimerLen: lunghezza massima del primer (intero, default 21);
- minTm: temperatura di melting minima (intero, default 52);
- minGCCont: minimo contenuto GC nel primer (reale, default 0.5);
- maxGCCont: massimo contenuto GC nel primer (reale, default 0.7);

- maxDimers: numero massimo di auto-dimeri (intero, default 8);
- maxHomopLen: lunghezza massima degli omopolimeri (intero, default 4);
- maxDeltaTm: intervallo massimo della temperatura di fusione per i set di primer (default 3);
- maxALenSpanE: intervallo massimo tra la mediana e il quantile della lunghezza dell'amplicone (maxALenSpanE) (intero, default 50);
- maxALenSpanEQ: quantile della lunghezza dell'amplicone (reale, default 0.01).

Intervalli di tolleranza fuzzy per le opzioni relative all'efficienza:

- minTmInterv: intervallo di tolleranza fuzzy per la temperatura minima di fusione (intero, default 2);
- minGCContInt: intervallo di tolleranza fuzzy per il contenuto GC minimo (reale, default 0.1);
- maxDimersInt: intervallo di tolleranza fuzzy per il numero massimo di auto-dimeri (intero, default 3);
- deltaTmInt: intervallo di tolleranza fuzzy per l'intervallo di temperature di fusione del set di primer (intero, default 2);
- maxHLenInt: intervallo di tolleranza fuzzy per la massima lunghezza dell'omopolimero (intero, default 2);
- maxALenSpanEI: intervallo di tolleranza fuzzy per la massima distanza tra la mediana e il quantile della lunghezza dell'amplicone (intero, default 50).

### 3.3.3. File di output

Al termine della computazione, il software scrive i risultati in quattro file separati:

- `init.primers` e `init.scores`: rappresentano il set iniziale di primer, forniti in input, con i relativi score calcolati;
- `out.primers` e `out.scores`: contengono i primer ottimi calcolati dall'algoritmo, con i relativi score;

I file con estensione `primers` contengono una coppia di set di primer per ogni riga; ciascuna riga contiene un elenco delimitato da tabulazioni di tutti i primer forward, seguito da "x" e quindi dall'elenco di tutti i primer reverse.

I file `scores` contengono invece una riga con i valori dei punteggi del primer-set-pair per ogni riga dei corrispondenti file `primers`. La prima riga del file contiene l'intestazione delle colonne, il carattere di separazione è sempre la tabulazione.

La Figura 8 mostra un esempio dei file appena descritti.

```
AGAGTTTGATCATGGCTCAG AGAGTTTGATCCTGGCTCAG x CTACCAGGGTATCTAATCC
AGAGTTTGATCATGGCTCAG AGAGTTTGATCCTGGCTCAG x GACTACCAGGGTATCTAAT
TCCTACGGGAGGCAGCAGT x CACGACACGAGCTGACGAC CACGGCACGAGCTGACGAC
CCTACGGGAGGCAGCAG x CACGACACGAGCTGACGAC CACGGCACGAGCTGACGAC
...
```

*Figura 8 - Esempio di contenuto dei file di output*

*In alto .primers  
a destra .scores*

Efficiency	Coverage	Matching-bias
7.21084	0.307019	1.50485
7.1354	0.312268	1.48647
10	0.711787	0.640456
8.40466	0.705445	0.650146
...		

### 3.4. Dettagli tecnici

mopo16S è disponibile gratuitamente sotto licenza GNU GPL<sup>50</sup>; è scritto in C++ (c++11 o c++14), e presenta come unica dipendenza esterna la libreria “SeqAn c++”<sup>51</sup>, con versione non inferiore alla 2.0.0.

Esistono due versioni del software: una semplice e una con supporto al multithreading (openMP<sup>52</sup>). Nella versione parallelizzata le run (si veda il parametro *runs* nella sez. 3.3.2) vengono eseguite in thread diversi, permettendo di velocizzare la computazione.

L’esecuzione di mopo16S può essere lanciata da riga di comando e durante l’ottimizzazione stampa a terminale il livello di avanzamento della computazione.

La durata delle computazioni non è fissata, ma dipende dai dati forniti in input, dalle impostazioni e dalla natura non-deterministica dell’algoritmo. Per questi motivi le esecuzioni possono durare da pochi secondi ad alcune ore. A tal proposito una caratteristica importante è che le prestazioni computazionali non dipendono dalla lunghezza degli ampliconi considerati.

---

<sup>50</sup> [https://it.wikipedia.org/wiki/GNU\\_General\\_Public\\_License](https://it.wikipedia.org/wiki/GNU_General_Public_License)

<sup>51</sup> <https://www.seqan.de>

<sup>52</sup> <https://www.openmp.org>





## Capitolo 4 Analisi dei requisiti

L'analisi dei requisiti risulta essere un'attività preliminare nel campo dell'ingegneria del software, essa consiste nell'individuare l'insieme delle caratteristiche che il sistema deve avere per svolgere correttamente il proprio scopo, ovvero deve soddisfare le funzionalità richieste dal committente, del cliente, oppure, come nel caso in esame, dell'utente.

In questo capitolo vengono esposti gli obiettivi dell'applicazione (sez. 4.1) e gli interessati alla buona riuscita del progetto (sez. 4.2). Nelle ultime due sezioni si elencano le funzionalità da implementare richieste, suddivise rispettivamente in requisiti funzionali e non funzionali. Nella sezione 4.3 si possono trovare requisiti funzionali, che definiscono cosa il sistema deve fare, quelli non funzionali espongono invece le modalità con le quali deve assolvere le sue funzioni, e sono delineati nella sezione 4.4.

### 4.1. Obiettivi

Il progetto presentato in questo documento nasce dalla necessità di fornire accesso alle funzioni del software mopo16S tramite un'interfaccia grafica di semplice uso. Lo scopo è quello di permettere, anche ad utenti con poca dimestichezza nei confronti dell'utilizzo del programma da riga di comando, l'esecuzione in ambiente cloud dei calcoli. Esempi di applicazione possono essere l'impossibilità di far uso del software in locale o la volontà di tenere un log delle eventuali ricerche in corso in un ambiente che in futuro potrebbe diventare il riferimento per quanto riguarda la verifica delle date di esecuzione dei calcoli e delle date di pubblicazione dei risultati.

Per far fronte a questa esigenza si è pensato di realizzare un'applicazione web in modo da fornire un servizio globalmente accessibile e agevolmente modificabile con eventuali nuove funzionalità garantendo agli utenti la continuità del servizio.

## 4.2. Utenti e stakeholder

Il sistema risulta vantaggioso per la comunità scientifica che lavora nel campo della metagenomica relativa al sequenziamento del gene ribosomiale 16S, permettendogli di aumentare la produttività diminuendo sforzo e risorse necessarie al loro lavoro.

Gli utenti del sistema sono principalmente due tipi di ricercatori:

- coloro che hanno bisogno dello strumento per svolgere il loro lavoro;
- quelli che sviluppano le tecnologie per fornire alla comunità nuovi algoritmi e metodi per effettuare gli studi.

Queste figure possono essere considerate anche stakeholder del progetto, poiché sono tutte interessate alla qualità e all'efficienza del servizio.

## 4.3. Requisiti funzionali

### 4.3.1. Gestione utenti

L'utente può registrarsi in autonomia all'applicazione, fornendo un username, un'email e una password.

All'interno dell'applicazione è necessaria la distinzione tra ruoli di amministratore e utente semplice, per permettere di assegnare permessi diversi in base al ruolo. Questo serve a definire le risorse alle quali ciascuno di essi può accedere, e le azioni che può effettuare su di esse.

### 4.3.2. Sequenze in input per mopo16S

L'applicazione permette all'utente di scegliere se caricare i propri set di sequenze di geni, di utilizzare quelli già caricati da lui stesso in precedenza o, in alternativa, quelli messi a disposizione con l'applicazione stessa. L'utente può infatti utilizzare dei set di sequenze precaricate nel sistema. I dati sono presi dal database GreenGenes 2013 e filtrati con le tassonomie aggiornate, come esposto nella sezione 3.3.1.

In genere i dati di questo tipo non variano nel tempo, a meno di correzioni di errori nei file o scoperte di nuove specie batteriche. Per questo motivo gli utenti utilizzeranno abitualmente i dati già proposti dal sistema e molto raramente caricheranno i propri set di sequenze.

La dimensione dei file in questione è mediamente di alcune decine di megabyte, dato che può contenere migliaia di sequenze lunghe solitamente tra i 700 e gli 800 caratteri. Molti contengono poche decine di sequenze perciò occupano pochi kilobyte (KB), alcuni sono dell'ordine delle centinaia di megabyte (MB), ma in generale è molto raro che superino i 500 MB.

È possibile scaricare i file delle sequenze di GreenGenes e di tutti i file che, a regime, verranno eventualmente contrassegnati come curati (vedi sez. 4.3.4). Non è invece permesso ricaricare i file delle altre sequenze, l'utente deve quindi provvedere di persona a mantenere i propri file per eventuali usi futuri in locale.

#### **4.3.3. Primer in input per mopo16S**

Gli insiemi di coppie di primer devono essere caricati dall'utente, sono possibili il riutilizzo ed il download di quelli caricati in precedenza.

Questi dati sono molto sperimentali quindi l'utente caricherà, normalmente, un file per ogni lavoro che andrà a sottoporre al sistema; non è escluso che possa riutilizzare lo stesso file per più lavori.

Le dimensioni di questi set sono molto ridotte, si tratta di decine di sequenze lunghe di solito tra i 17 ed i 21 caratteri, per un file che risulta occupare alcuni KB, ed in generale meno di 10 MB.

#### **4.3.4. Specifiche in comune per l'input**

I dati possono essere caricati in due modi differenti: specifica diretta tramite casella di immissione testo (sconsigliato per file corposi) oppure facendo l'upload di un file di testo. In entrambi i casi il contenuto deve essere in formato FASTA, le specifiche sono elencate nella sezione 3.3.1.

L'utente deve fornire un nome breve ma descrittivo per l'input che sta caricando sull'applicazione, assieme ad una più dettagliata descrizione ed un numero di versione, utili a lui stesso per identificare con precisione, anche a distanza di tempo, le sequenze caricate. Necessario il salvataggio della data di caricamento dei dati nel sistema.

L'utente può decidere se rendere pubblici o privati i dati di input da lui caricati, può vedere solo i propri e quelli pubblici caricati da altri utenti. Gli amministratori dell'applicazione web sono in grado di visionare anche i file privati.

Gli amministratori del sistema possono assegnare uno stato di "curato" ai file delle sequenze e dei primer. Vuole essere un indicatore per tutti gli utenti stante a rappresentare che il relativo contenuto è stato verificato e scientificamente approvato. Questo permette ad un utente generico di utilizzare senza preoccupazioni ed in maniera sicura le sequenze non caricate da lui. I dati di GreenGenes devono avere questo flag attivo.

#### **4.3.5. Sottomissione del job**

L'utente può creare nuove richieste di esecuzione del software indicando: nome, descrizione, dati input da utilizzare ed eventuali parametri di mopo16S differenti da quelli di default.

Il sistema memorizza la versione attuale di mopo16S installata nel sistema e la data di sottomissione del job.

È possibile visualizzare la lista dei job completati, in esecuzione, in attesa o falliti.

Come per i dati di input, l'utente può definire se rendere il job privato o meno, i criteri per la visualizzazione sono gli stessi.

#### **4.3.6. Esecuzioni di mopo16S**

Il sistema memorizza le esecuzioni dei lavori richiesti dagli utenti con data di inizio e fine; eventuali errori vengono salvati ed inviati agli admin. A lavoro completato, l'utente viene informato con una email automatica.

#### **4.3.7. File di output da mopo16S**

I quattro file prodotti dalle esecuzioni di mopo16S vengono salvati per il successivo download o utilizzo nell'applicazione stessa. Viene salvata anche la data di completamento del job.

La dimensione di questi file si aggira mediamente intorno ad 1 KB; in base alle impostazioni con le quali viene lanciato il programma, e alla quantità di primer forniti in ingresso, potrebbe raggiungere i 10 KB, ma molto raramente.

#### **4.3.8. Visualizzazione dei risultati**

L'interfaccia propone all'utente due tipi di visualizzazione con le quali l'utente può analizzare i dati di output prodotti dall'esecuzione di mopo16S: una in forma tabellare e una in forma grafica. I dati vengono presentati associando le coppie di set di primer ai relativi score, in questo modo risultano di immediata e semplice comprensione, rispetto all'utilizzo dei soli file di output, che separano le sequenze dai punteggi.

Dal momento che l'algoritmo assegna tre punteggi ad ogni sequenza, è necessario studiare una soluzione per la visualizzazione di tali informazioni nel grafico.

L'utente può scaricare i file di output, le tabelle e l'immagine del grafico per utilizzarle al di fuori dell'applicazione all'evenienza.

#### **4.3.9. API**

Le funzioni principali dell'applicazione sono disponibili anche tramite una web API, attraverso la quale potranno interfacciarsi eventuali client futuri.

#### **4.4. Requisiti non funzionali**

Dal momento che le esecuzioni di mopo16S possono richiedere anche diverse ore di calcolo, è necessario eseguirle in background permettendo all'utente di controllare i risultati in un secondo momento. In caso di errori, i lavori possono venire rilanciati per un massimo di tre volte.

Il sistema decide quando far eseguire i lavori richiesti dagli utenti in base al carico del calcolatore. Inoltre assegna la priorità attenendosi a policy che tengono conto del numero di lavori dell'utente e del suo stato di amministratore o meno. Ogni utente può eseguire un massimo di cinque lavori al giorno.

## Capitolo 5 Progettazione

In questo capitolo sono descritte le varie fasi di progettazione dell'applicazione partendo dal design concettuale e logico della base di dati, per poi definire l'architettura del sistema. Vengono quindi delineate le modalità in cui le funzioni richieste, elencate nel capitolo precedente, vengono rese disponibili all'utente.

La progettazione concettuale (sez. 5.1) fornisce una rappresentazione del dominio di interesse mediante un modello ad alto livello, integrando tutti i concetti rilevanti e indipendenti dal DBMS.

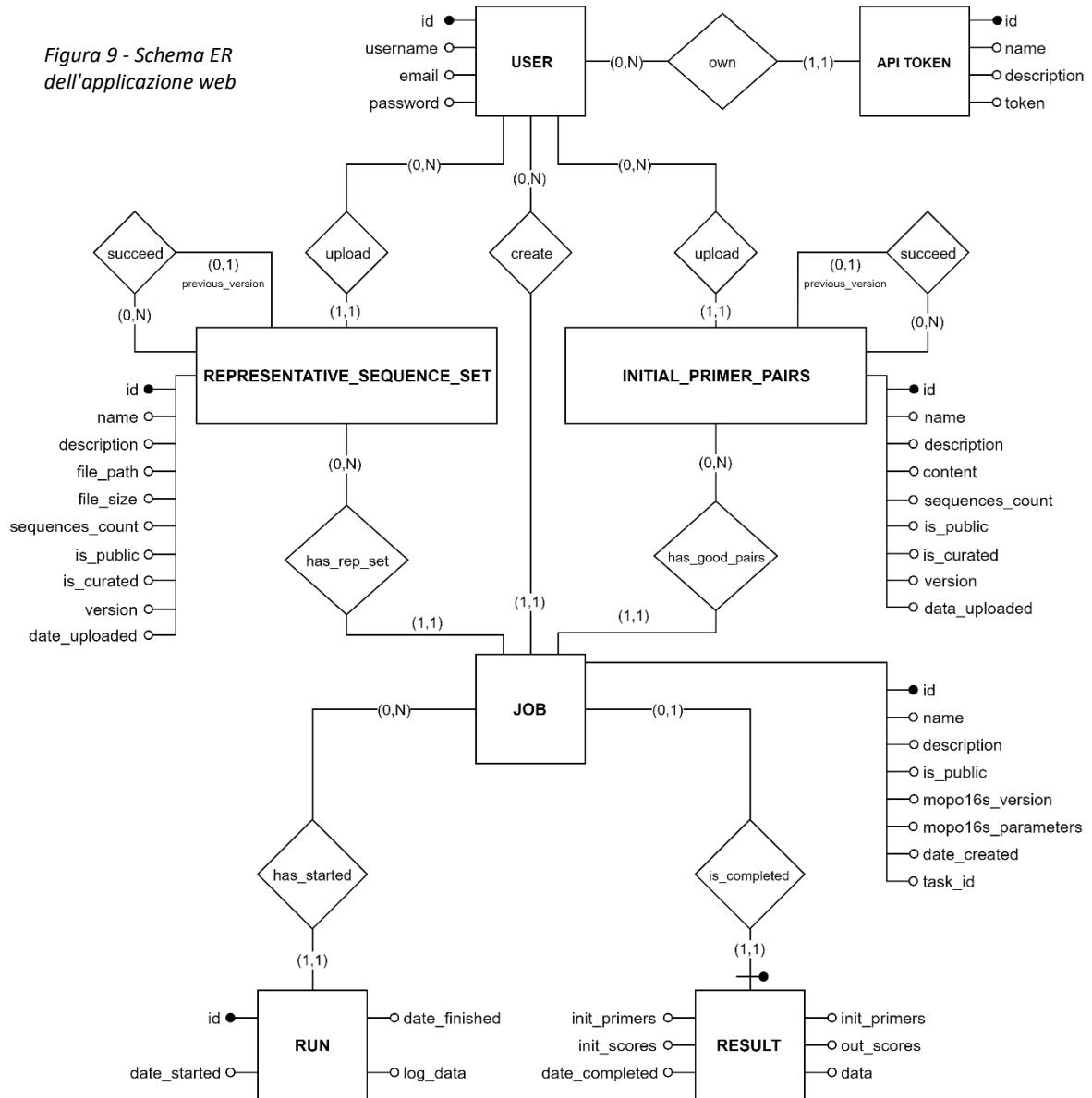
La progettazione logica (sez. 5.2) ha come obiettivo la disposizione dei dati in una serie di strutture logiche, indipendentemente dalla struttura fisica adibita a contenerle e memorizzarle. L'unica caratteristica da considerare è la classe di DBMS nella quale implementare il modello logico, che nel nostro caso è quella dei database relazionali.

Lo scopo della fase di progettazione consiste nella redazione di una lista di tutte le componenti necessarie al sistema nel contesto della soluzione concepita, queste sono esposte nella sezione 5.3.

### 5.1. Progettazione concettuale

Viene qui presentato il modello entità-relazione (schema ER), che permette di progettare in modo efficiente il database. Lo schema concettuale è molto utile ai fini della documentazione del progetto, risulta infatti facilmente comprensibile anche dagli stakeholder e costituisce il punto di riferimento per eventuali future modifiche o estensioni dell'applicazione.

Figura 9 - Schema ER dell'applicazione web



La funzionalità base del sistema riguarda la gestione utenti, è necessario quindi definire un'entità che contenga i dati necessari ad identificare coloro che possono accedere all'applicazione. Questi sono salvati nell'entità *User*, in particolar modo è doveroso sottolineare che per questioni di sicurezza il campo password non conterrà la password in chiaro ma la funzione hash della stessa.

I set di sequenze e di primer sono rappresentati, rispettivamente, da *Representative Sequence Set* e *Initial Primer Pairs*. I campi descrittivi di base sono in comune, inclusi i flag che permettono la distinzione tra pubblico e privato e lo stato di curato o meno, e il



numero di sequenze contenute. I set di sequenze, che sono utilizzati molto spesso, vengono salvati nel file system e sono descritti anche da informazioni quali il percorso del file e la sua dimensione. Al contrario i set di primer vengono salvati nel database nel campo *content*, poiché il loro utilizzo è molto più raro ed il contenuto è molto meno oneroso in termini di spazio.

*Job* rappresenta le richieste di esecuzione che vengono create dagli utenti, memorizza la versione di mopo16S installata nel sistema e i parametri da usare per il lancio del software stesso, assieme all'identificatore del task che eseguirà il job associato. Quest'ultimo identifica un'operazione asincrona e non sarà un campo con integrità referenziale.

L'entità *Run* memorizza le esecuzioni del software, che possono essere molteplici, e il log di eventuali errori associati a ciascuna di esse.

In *Result* vengono salvati i risultati delle computazioni effettuate. Ognuno dei quattro campi corrisponde ad un file in output da mopo16S, si è incluso anche un campo *data* per memorizzare in forma strutturata e pronta all'uso i dati in uscita.

Per quanto riguarda l'API si è deciso di adottare un sistema di autenticazione tramite token, le relative informazioni sono racchiuse nell'entità *Api Token*.

Le relazioni riguardanti l'utente sono auto-esplicative: *upload* e *create* si riferiscono alle operazioni di caricamento delle sequenze e alla sottomissione dei lavori al sistema. *own* invece indica se l'utente possiede dei token per utilizzare l'API, e in quale quantità, perché può richiedere agli amministratori del sistema più di un token.

Ogni job ha come riferimento un set di sequenze (*has\_rep\_set*) ed un set di primer (*has\_good\_pairs*) e dovrebbe essere idealmente completato con una sola esecuzione. Si è previsto tuttavia un sistema di gestione errori tale per cui in caso di fallimento, esse possano essere rieseguite da capo; a tal fine, la relazione *has\_started* permette di creare più *Run* per ogni *Job*.

I risultati, infine, sono collegati al relativo *Job* tramite la relazione *is\_completed*; qui il nome indica lo stato del *Job* dal momento in cui possiede un risultato.

Considerato il fatto che tutti i *Job*, ad esclusione degli sperabilmente pochi falliti, avranno il loro risultato dopo al massimo qualche ora, sarebbe stato possibile conferire a *Job* tutti gli attributi di *Result*. Si è deciso tuttavia di mantenere la relazione *is\_completed* per facilitare eventuali estensioni future riguardanti il possibile caricamento di dati esterni da parte dell'utente (computazioni mopo16S eseguite in locale). Per esempio il fatto di poter memorizzare nel proprio account i file caricati solo per la visualizzazione.

Per quanto riguarda gli identificatori, l'unico esterno è quello di *Result*, questo perché ogni *Job* può avere solamente un risultato.

## 5.2. Progettazione logica

Riportiamo in questa sezione lo schema logico derivante dalla precedente progettazione concettuale, considerando che il tipo di database incaricato di ospitare i dati è di tipo relazionale.

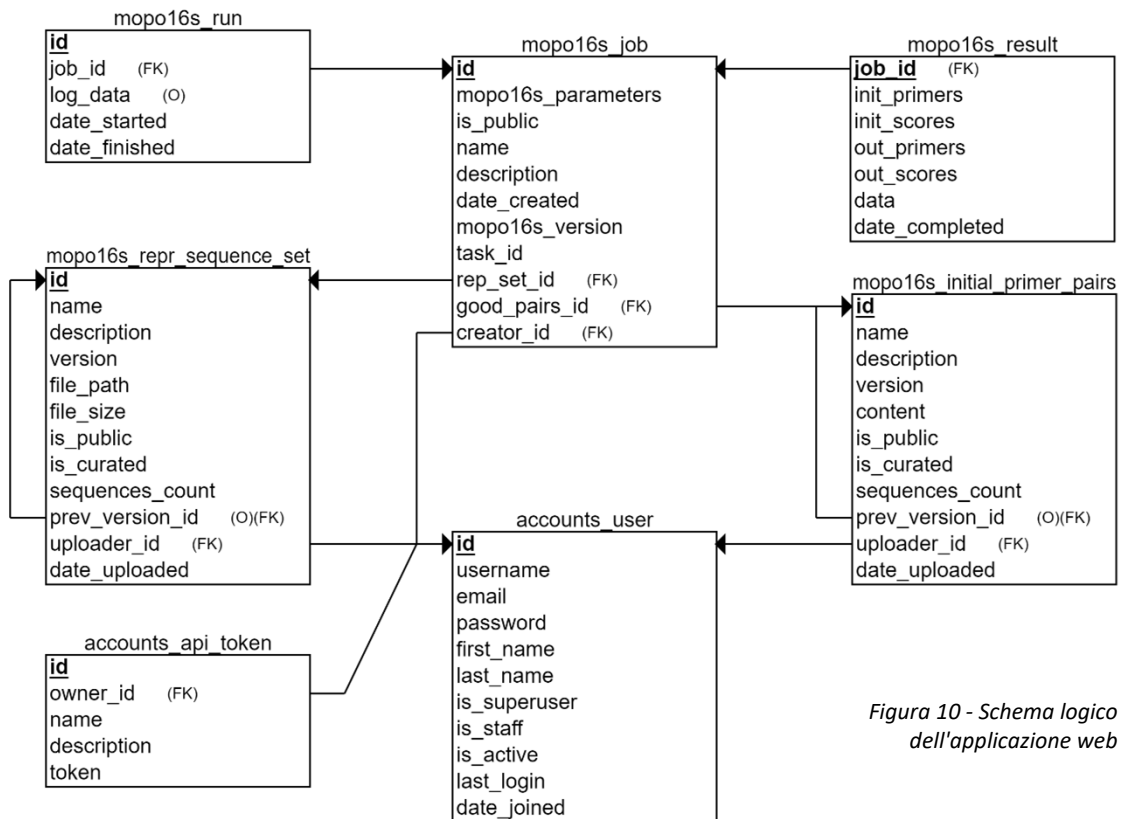


Figura 10 - Schema logico dell'applicazione web

Lo schema in Figura 10 - Schema logico dell'applicazione web è facilmente deducibile dal diagramma ER (Figura 9), data la linearità dei dati necessari all'applicazione. Le relazioni uno-a-molti sono state incorporate nelle entità dalla parte con cardinalità 1. L'unica relazione uno-a-uno risulta essere *is\_completed*, che è stata inclusa nella tabella che rappresenta *Result* ed usata come chiave. I nomi delle tabelle seguono la nomenclatura di Django, il framework utilizzato per il progetto, mantenendo la coerenza all'interno di tutta l'applicazione.

Nello schema appaiono anche alcuni attributi di *User* non presenti nello schema ER della sezione precedente, questo perché il framework mette a disposizione alcuni strumenti di default per semplificarne l'implementazione. Nella Figura 11 si elencano altre tabelle che non sono oggetto di progettazione nell'ambito dell'applicazione web in analisi, sono stati infatti utilizzati sistemi che implementano già alcune funzionalità utili agli scopi da raggiungere. Nel capitolo successivo si trovano maggiori informazioni a riguardo, in particolare nelle sottosezioni 6.2.2, 6.2.4 e 6.2.7.

```
accounts_user_groups
accounts_user_user_permissions
auth_group
auth_group_permissions
auth_permissions
django_admin_log
django_content_type
django_migrations
django_session

django_celery_beat_clockedschedule
django_celery_beat_crontabschedule
django_celery_beat_intervalschedule
django_celery_beat_periodictask
django_celery_beat_periodictasks
django_celery_beat_solarschedule
django_celery_results_taskresult
```

*Figura 11 - Altre tabelle non oggetto di progettazione*

Per completare l'analisi si è voluto stimare anche il volume di dati che l'applicazione genererà a regime, per permettere di dimensionare adeguatamente i sistemi. La natura dell'applicazione e i dati da essa elaborati risultano in un carico operativo del database non molto elevato. Si stima una media a regime di 5-10 utenti attivi al giorno, ognuno di questi può richiedere l'esecuzione di massimo 5 lavori al giorno. Riguardo alla mole di dati da gestire si è eseguito un calcolo tenendo conto solamente dei file caricati dagli utenti e di quelli prodotti dalle esecuzioni di mopo16S, dal momento che il resto del contenuto delle tabelle ha dimensioni più piccole di almeno un ordine di grandezza. Ne risulta che lo spazio operativo necessario è minimo e sarebbe dominato da eventuali caricamenti di database di sequenze nuovi o aggiornati.

$$5 \text{ job} * 10 \text{ utenti/gg} * 10 \text{ KB/gg} = 500 \text{ KB/gg} < 200 \text{ MB/anno}$$

### 5.3. Architettura

Nella definizione dell'architettura dell'applicazione web si è scelto di adottare una soluzione di tipo *three-tier*, e di seguire i principi architetturali del paradigma REST per conferire al sistema le molteplici caratteristiche associate (esposte nella sezione 2.1.3), come ad esempio l'uniformità nella presentazione delle risorse, la semplicità di manutenzione e la possibilità di scalare facilmente l'applicazione.

Con l'architettura *three-tier* l'applicazione web viene suddivisa in tre strati, ognuno dedicato ad una specifica funzione: interfaccia utente, logica funzionale e gestione dei dati persistenti. L'interazione tra gli strati, che possono anche essere distribuiti su nodi diversi in rete, segue il paradigma client-server descritto da REST, permettendo così eventuali modifiche ai singoli componenti senza richiedere la sostituzione anche degli altri.

La soluzione comprende quindi:

- una parte dedicata alla presentazione dei contenuti all'utente, realizzata con un server web che serve anche contenuti statici;
- uno strato che implementa la logica a cui deve rispondere l'applicazione per generare dinamicamente i contenuti;
- un DBMS per la persistenza dei dati.

Date le specifiche dell'applicazione bisogna considerare anche un componente indispensabile ad eseguire operazioni asincrone e quelle periodiche, in particolar modo le run di mopo16S. Solitamente si usa un gestore di code di lavoro, il quale deve poter fare chiamate a sistema per avviare il software da riga di comando.

Nel campo delle applicazioni web è molto pratico aggiungere all'architettura un sistema di cache, questo permette un servizio più veloce di dati utilizzati di frequente.

Si è voluto rendere il tutto configurabile in base alle necessità e facilmente portabile; pertanto, al fine di permettere una distribuzione più veloce ed efficace con elevata compatibilità d'ambiente, è stato scelto di containerizzare l'applicazione.

La Figura 12 rappresenta l'architettura nel suo complesso, dove le frecce indicano le comunicazioni tra i vari componenti.

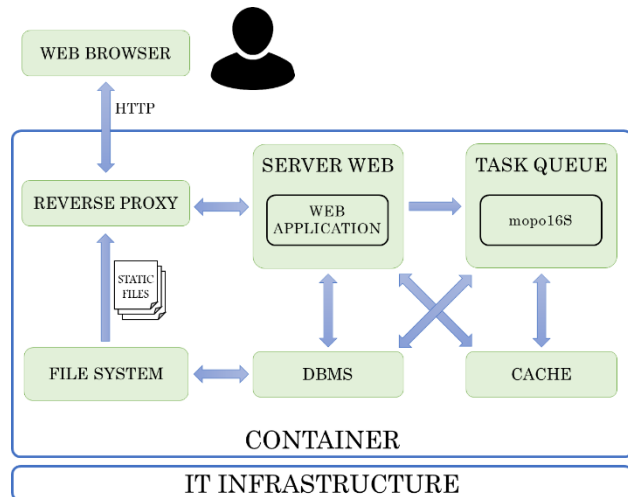


Figura 12 - Architettura dell'applicazione

## Capitolo 6 Implementazione

In questo capitolo si illustrano la configurazione dei componenti scelta per concretizzare l'architettura determinata nella sezione 5.3 e i dettagli riguardanti lo sviluppo del codice.

La soluzione identificata come ottimale per il progetto si basa su diversi fattori, che sono presentati assieme ai vari componenti nella sezione 6.1. La 6.2 riporta alcuni dettagli tecnici a proposito del codice sviluppato, e nella sezione 6.3 si accenna al procedimento da eseguire per la creazione dei container.

### 6.1. Componenti del sistema

Le motivazioni che hanno influito maggiormente sulla preferenza dei componenti elencati in questa parte del capitolo rispetto ad altri disponibili sono principalmente tre:

- molti di essi sono stati utilizzati dallo sviluppatore per altri progetti, ne traspare quindi una competenza, seppur non professionale, acquisita negli anni;
- recentemente questo tipo di configurazione si è diffusa nel campo delle applicazioni web in Python;
- sono tutti gratuiti e open source.

Il linguaggio di programmazione che si è deciso di utilizzare è Python 3.7 e per lo sviluppo si è deciso di servirsi di un framework per applicazioni web. Il principio DRY (*don't repeat yourself*) descrive efficacemente il fine principale della soluzione: le tecniche di riuso di codice già scritto sono fortemente preferite e consigliate, permettono di concentrarsi sulla scrittura del solo codice necessario al software da sviluppare senza reinventare la ruota, riducendo così in parte la scrittura di codice back-end e migliorando qualità e sicurezza del prodotto. Lo schema three-tier presenta diverse analogie con il pattern MVC e questo è

implementato nella maggior parte dei framework web, tra i quali troviamo Django, che è stato scelto come base per il progetto, nella sua versione stabile 2.2.5<sup>53</sup>.

Indipendentemente dal linguaggio, l'esecuzione del software relativo ad un applicativo web richiede un server web che ne gestisca le chiamate. Per permettere a Django di prendersi carico delle richieste provenienti dall'esterno vi è stato abbinato un server web di tipo WSGI, dato che si parla di un'applicazione scritta in Python. Il componente è Gunicorn, che si è visto essere il più adatto ed efficiente per lavorare con Django, senza bisogno di parti aggiuntive o dipendenze esterne<sup>54</sup>. Oltre a ciò, nel campo delle applicazioni accessibili attraverso una rete è consigliato utilizzare un reverse proxy per servire file statici e aggiungere uno strato aggiuntivo di protezione per il sistema. Il componente selezionato per svolgere questa funzione è nginx.

La persistenza delle informazioni necessarie all'applicazione è fondamentale, sia per la gestione degli utenti e per i file da essi caricati, sia per i dati generati dal software stesso. Per salvare le informazioni strutturate ci serviamo di PostgreSQL 12.1, il salvataggio dei file con dimensioni più elevate avviene invece nel file system, che permette di gestirli più efficacemente. Allo stesso tempo, l'applicazione fa uso di alcune informazioni volatili e altre che vengono utilizzate molto frequentemente, risulta perciò efficiente memorizzarle in modo non strutturato in un sistema di cache che si affianca al database per garantire maggiore reattività, fondamentale per garantire la migliore esperienza all'utente dell'applicazione web. Il database si occupa dell'archiviazione continua e a lungo termine, la cache si concentra invece su risposte rapide e cambiamenti repentini; per il caching la scelta del software è ricaduta su Redis 5.0. La decisione è stata presa considerando in particolar modo le ottime prestazioni e l'elevata affidabilità di questi due sistemi.

Nell'ambito dei task asincroni si è deciso di utilizzare Celery, un gestore di code di lavoro che dalla versione 3.1 si integra nativamente con Django<sup>55</sup>. Celery permette di eseguire le run di mopo16S senza preoccuparsi del tempo impiegato per le computazioni che, come

---

<sup>53</sup> <https://docs.djangoproject.com/en/2.2>

<sup>54</sup> <https://docs.djangoproject.com/en/2.2/howto/deployment/wsgi/gunicorn>

<sup>55</sup> <http://docs.celeryproject.org/en/latest/django/first-steps-with-django.html>



descritto in precedenza, non hanno un tempo di esecuzione definito o limitato. Per il monitoraggio di Celery è molto utile l'abbinamento con lo strumento Flower.

Tutti i componenti vengono eseguiti sul sistema operativo Debian. La motivazione principale di questa preferenza risiede nella notevole stabilità assicurata del sistema. Non di minore importanza le questioni di sicurezza e snellezza dello stesso, che si riportano per transitività anche al prodotto finito.

Per la containerizzazione, infine, si è scelto Singularity, dal momento che risulta conosciuto maggiormente, rispetto ad altri metodi di virtualizzazione a livello di SO, dai sistemisti che dovranno gestire il sistema in produzione. Si è comunque constatato che Singularity si integra molto bene con Docker<sup>56</sup>, che è molto più diffuso e offre molteplici container già pronti all'uso<sup>57</sup>. È possibile infatti convertire le immagini Docker nel formato nativo di Singularity, ovvero *Singularity Image Format (SIF)*.

In Figura 13 sono rappresentati con maggiore chiarezza e praticità tutti i componenti citati evidenziando le relazioni tra loro esistenti.

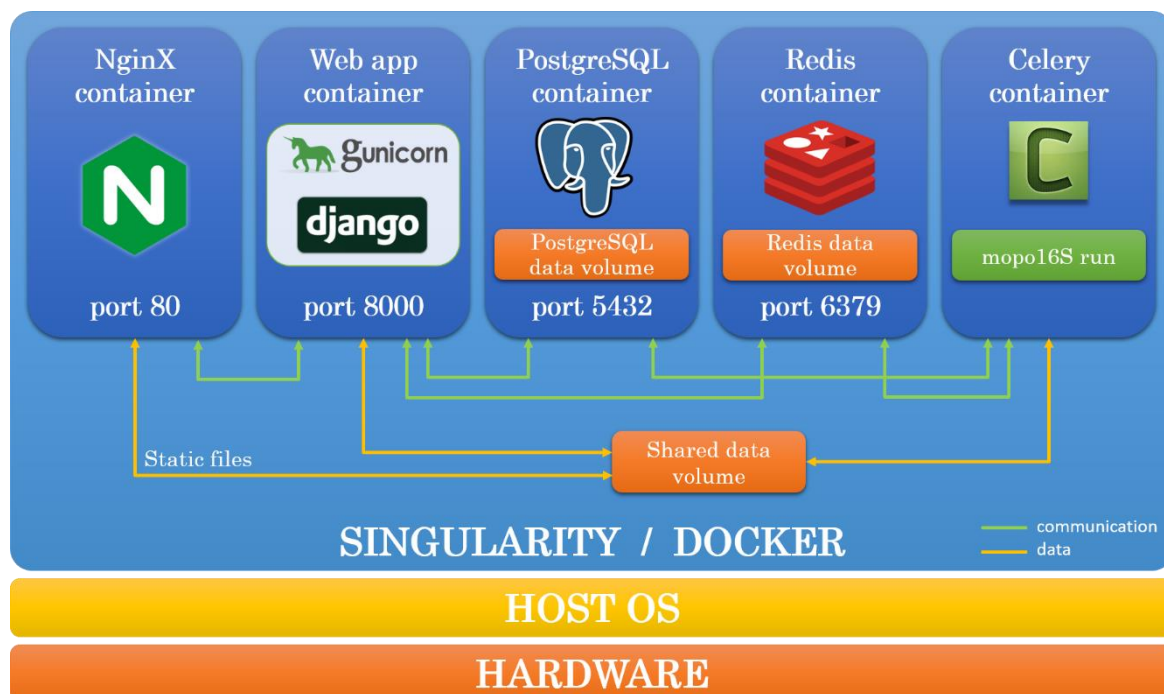


Figura 13 - Componenti del sistema nell'architettura completa

<sup>56</sup> [https://sylabs.io/guides/3.5/user-guide/singularity\\_and\\_docker.html](https://sylabs.io/guides/3.5/user-guide/singularity_and_docker.html)

<sup>57</sup> <https://github.com/docker-library>

## 6.2. Codice

In questa sezione si forniscono i principali dettagli implementativi, come ad esempio stile utilizzato per la programmazione, scelte operative e strumenti usati.

### 6.2.1. Struttura modulare

Django, come altri framework o convenzioni, consiglia di utilizzare una struttura a moduli separati per funzionalità per scrivere applicazioni riusabili e facilmente manutenibili. Si è quindi deciso di suddividere in cinque moduli principali il progetto:

- configurazioni riguardanti l'ambiente di produzione e mopo16S;
- funzioni principali e le dinamiche relative ai modelli;
- gestione account;
- interfaccia web;
- web API.

In questo modo si permette a chiunque voglia riutilizzare l'applicazione per un suo progetto di installare solo i moduli necessari. Chiaramente i componenti principali sono quelli contenenti le configurazioni e le dinamiche associate al progetto, il modulo per gli account è necessario ma sostituibile, e gli altri due sono opzionali.

### 6.2.2. Applicazioni di terze parti

Seguendo il principio DRY citato in precedenza si è scelto di utilizzare alcune applicazioni sviluppate da terzi, disponibili come pacchetti importabili nella propria applicazione Django. Sono state importate *django\_celery\_beat*<sup>58</sup> e *django\_celery\_results*<sup>59</sup> per gestire i task di celery, e *django\_rest\_framework* (vedi sez. 2.1.8.2) è stato utilizzato per costruire una web API con più funzionalità.

---

<sup>58</sup> <https://github.com/celery/django-celery-beat>

<sup>59</sup> <https://github.com/celery/django-celery-results>

### 6.2.3. Configurazioni

Nel modulo di base contenente le configurazioni di Django<sup>60</sup> sono state aggiunte delle impostazioni ad hoc per gestire mopo16S, le sue funzionalità e le sue esecuzioni.

`MOPO16S_VERSION` e `MOPO16S_PATH` riportano rispettivamente la versione e il percorso dell'eseguibile. `MOPO16S_MAX_THREADS` indica il massimo numero di thread utilizzabili dal sistema per evitare di sovraccaricare la macchina, e `MOPO16S_MAX_THREADS_PER_INSTANCE` precisa il massimo numero di thread utilizzabile da ogni singola run. `MOPO16S_PARAMETERS`, infine, contiene le informazioni riguardo a tutti i parametri accettati da mopo16S.

Questo modo di procedere assicura la possibilità di cambiare le impostazioni in modo molto semplice in caso di modifiche al software, oltre al fatto di rendere il tutto molto dinamico e chiaro evitando di codificare tutti i parametri nelle parti più disparate del codice. Nel caso si decida, ad esempio, di cambiare la versione di mopo16S disponibile nel server di produzione, è possibile, con qualche semplice ritocco di queste impostazioni, cambiare automaticamente il comportamento dell'applicazione web.

La scelta di gestire i parametri con le modalità appena descritte deriva dalla scelta progettuale di chiamare il software da riga di comando specificando ogni volta tutti i parametri. Lo scopo è di evitare che eventuali modifiche ai valori di default nel codice non riportate nell'applicazione possano riflettersi in errori di esecuzione. Potrebbe succedere che l'utente, convinto di usare il parametro di default propostogli a video, non si accorga che in realtà il codice esegue con un altro valore. In questo modo è anche possibile salvare nel database i parametri utilizzati per uno specifico job.

Si è deciso di non consentire all'utente di impostare i parametri *thread* e *verbosity*. Il numero dei thread assegnati ad un determinato job viene calcolato da un algoritmo studiato appositamente per bilanciare il carico del server. L'algoritmo alloca le risorse rispettando i parametri descritti sopra, che sono valorizzati in modo da lasciare sempre

---

<sup>60</sup> <https://docs.djangoproject.com/en/2.2/topics/settings>

almeno un core scarico, al fine di non bloccare completamente la macchina, e garantire uno smaltimento della coda job più efficiente ed efficace possibile. Per quanto riguarda la verbosità dell'output si è deciso di fissare a 0 il valore, e salvare nel database il log semplice stampato nello standard output di mopo16S. La motivazione risiede nel fatto che non è necessario mostrarlo all'utente, ma risulta utile per eventuali operazioni di debug o controlli sulle esecuzioni; la forma più dettagliata del log (`verbose = 1`) è utile solamente ai fini di debug del software mopo16S, ma non dell'applicazione.

#### 6.2.4. Mappatura degli oggetti

Grazie all'estesa API offerta dallo strumento di mappatura relazionale degli oggetti (ORM) messo a disposizione da Django<sup>61</sup>, è possibile definire direttamente dal codice entità e relazioni. Tuttavia si è scelto di utilizzare questa funzione solamente in parte, data la progettazione effettuata e la conoscenza di PostgreSQL e del suo linguaggio.

Django rende disponibile agli sviluppatori un sistema di gestione utenti integrato e completamente personalizzabile a seconda delle esigenze. Per l'utente dell'applicazione web in oggetto è stato utilizzato il modello di base prendendo in considerazione un'eventuale estensione futura delle funzionalità e dei campi, si è quindi creata una sottoclasse del modello astratto di utente definito dal framework<sup>62</sup>.

Sono stati definiti nel codice i modelli relativi alle tabelle progettate già create nel database con le convenzioni di nomenclatura di Django<sup>63</sup>. Ognuno dei modelli circoscritto nell'apposito modulo, ad esempio *User* e *Api Token* in quello per gestire gli utenti. Attraverso il comodo strumento di gestione delle migrazioni fornito da Django<sup>64</sup> sono poi state propagate in automatico le definizioni delle tabelle aggiuntive necessarie alle funzioni base del framework e delle applicazioni di terze parti installate ed importate. Grazie a questo strumento è possibile apportare automaticamente modifiche allo schema del database dopo aver cambiato i modelli nel codice. Le tabelle che sono state definite

---

<sup>61</sup> <https://docs.djangoproject.com/en/2.2/topics/db/models>

<sup>62</sup> <https://docs.djangoproject.com/en/2.2/topics/auth/customizing/#extending-django-s-default-user>

<sup>63</sup> <https://docs.djangoproject.com/en/2.2/ref/models/options/#table-names>

<sup>64</sup> <https://docs.djangoproject.com/en/2.2/topics/migrations>

direttamente su PostgreSQL sono tutte quelle citate nella progettazione del database, ad esclusione della tabella che rappresenta gli utenti.

### 6.2.5. URL

Per lo schema degli URL si è voluto seguire i principi dell'architettura REST, a cui aderisce in generale anche Django<sup>65</sup>. Il framework permette di definire in moduli appositi la struttura che si preferisce dare agli URL della propria applicazione. Questa configurazione consiste in un mapping tra il percorso degli URL e le funzioni Python che sono chiamate views. Di seguito l'elenco degli URL con la risorsa associata:

ENDPOINT	RISORSA
/	Home page
/primers/	Lista dei set di primer
/primers/<id>/	Singolo set di primer
/primers/new/	Form per la creazione di un nuovo set di primer
/sequences/	Lista dei set di sequenze
/sequences/<id>/	Singolo set di sequenze
/sequences/new/	Form per la creazione di un nuovo set di sequenze
/jobs/	Lista dei job
/jobs/<id>/	Job sottomesso
/jobs/new/	Form per la creazione di un nuovo job
/results/<id>/	Interfaccia interattiva per la visualizzazione dei risultati di un job
/results/<id>/download/	File .zip contenente tutti i file di output di mopo16S

### 6.2.6. Template HTML

Tutta la parte di visualizzazione è sviluppata con il potente motore di template HTML<sup>66</sup> del quale Django è corredato. Questo permette il rendering delle informazioni da presentare all'utente tramite una sintassi intuitiva.

<sup>65</sup> <https://docs.djangoproject.com/en/2.2/topics/http/urls>

<sup>66</sup> <https://docs.djangoproject.com/en/2.2/ref/templates/api>

### 6.2.7. Gestione utenti

Il modulo di autenticazione di Django<sup>67</sup> mette a disposizione diversi strumenti di base per la gestione degli utenti e dei relativi permessi. Le informazioni che permettono queste funzionalità sono memorizzate nelle tabelle accessorie elencate nel primo blocco della Figura 11, nella sezione 5.2.

Sono state sviluppate le apposite view per la registrazione di nuovi utenti e per la modifica di quelli esistenti in modo da adempiere ai requisiti esposti nella sezione 4.3.1. È anche possibile il reset della password con il consueto link “password dimenticata?”.

### 6.2.8. Caricamento delle sequenze e dei primer

L’upload dei file da fornire in input a mopo16S viene effettuato pressoché con lo stesso procedimento sia per le sequenze 16S che per i primer. È stato creato un form nel quale l’utente inserisce le informazioni relative al file che sta caricando, e può decidere se inserire in una casella di testo i dati o allegare un file di testo.

### 6.2.9. Creazione job

Il sistema permette di sottomettere job tramite un form dove inserire nome, descrizione, selezionare i dati già caricati in precedenza da utilizzare come input per mopo16S e di cambiare a necessità i parametri di default.

### 6.2.10. Esecuzione job

Dal momento che i job hanno tempi di esecuzione molto variabili, si è scelto di assegnare la priorità in base al numero di job richiesti da parte dell’utente tenendo conto anche se fa parte o meno dello staff.

La politica di gestione prevede che vengano eseguite non più di due run di mopo16S contemporaneamente per evitare che i task blocchino il sistema. L’applicazione è corredata anche di task asincroni che controllano se si verificano errori nell’esecuzione dei job, ad esempio se i processi vengono terminati forzatamente.

---

<sup>67</sup> <https://docs.djangoproject.com/en/2.2/topics/auth/default>

#### 6.2.11. Visualizzazione delle risorse

Ogni utente può vedere le risorse da lui create e quelle pubbliche; gli amministratori sono invece in grado di vedere qualsiasi risorsa disponibile nel sistema. È possibile visualizzare le risorse singolarmente oppure, per una panoramica più rapida, sotto forma di lista.

#### 6.2.12. Visualizzazione dei risultati

La parte più dinamica dell'applicazione risiede nella funzione di visualizzazione tabellare e grafica dei risultati.

I risultati prodotti in output da mopo16S vengono raffigurati in un grafico 2D a punti dove ogni primer-set-pair è rappresentato da un pallino. Dal momento che gli score sono tre si è scelto di rappresentare la terza dimensione con il raggio dei pallini. La considerazione alla base di questa decisione sta nel fatto che un grafico che rappresenta la terza dimensione con forme o colori non è facilmente leggibile ed interpretabile. La diversa dimensione dei pallini permette di capire al volo la relazione tra i vari punti, cosa che invece non possono fare una lista di forme o una paletta colori dato che non sono riconducibili nell'immediato ad una scala lineare.

Vengono fornite anche due rappresentazioni in forma tabellare, una per l'input e una per l'output, in cui i primer-set-pair sono elencati in forma degenera per ragioni visive ed estetiche. Per ognuno dei tre score i punteggi migliori e peggiori sono colorati rispettivamente in verde e rosso.

Un'ultima importante considerazione riguarda la gestione del matching-bias: per la sua natura è valorizzato in maniera opposta agli altri due score.

#### 6.2.13. Web API

Per quanto riguarda l'accesso tramite interfaccia si è deciso di fornire il servizio con autenticazione tramite token. Ogni utente, previo richiesta agli amministratori, può ricevere una o più chiavi di autenticazione che permettono ai propri client di essere riconosciuti dal sistema e utilizzare le risorse disponibili nello stesso.

Il token consiste in una sorta di password da specificare nell'URL della richiesta, che deve essere in questo formato: <endpoint>/api/<token>/<risorsa>. L'utente deve provvedere a mantenere al sicuro il proprio token ed evitare usi impropri, pena la revoca dell'accesso al servizio.

Per uniformità, la struttura degli URL è la stessa di quella dell'interfaccia grafica, semplicemente preceduta da api/<token>/. La tabella che segue evidenzia i parametri obbligatori e opzionali per ogni tipo di richiesta. Le indicazioni tra parentesi si riferiscono al tipo di dato, i parametri senza la dicitura sono stringhe.

METODO e ENDPOINT	PARAMETRI OBBLIGATORI	PARAMETRI OPZIONALI	RISPOSTA
GET /			Informazioni dell'utente detentore del token
GET /primers/		offset (int)	Lista dei set di primer
GET /primers/<id>/			Set di primer
POST /primers/new/	name, description, content, version	public (bool), previous_version (int)	Set appena creato
GET /sequences/		offset (int)	Lista dei set di sequenze
GET /sequences/<id>/			Set di sequenze
POST /sequences/new/	name, description, content, version	public (bool), previous_version (int)	Set appena creato
GET /jobs/		offset (int)	Lista dei job
GET /jobs/<id>/			Job sottomesso
POST /jobs/new/	name, description, rep_set (int), good_pairs (int)	public (bool), parameters (dict)	Job appena creato
GET /results/<id>/download/			File .zip contenente tutti i file di output di mopo16S

I parametri sono gli stessi che si trovano nei form dell'interfaccia grafica e hanno i medesimi vincoli. La funzionalità di paginazione è rispecchiata nel parametro *offset* che permette di avanzare nell'ispezione della lista, dato che i metodi restituiscono al massimo 20 elementi per volta. I parametri per un nuovo job vanno inseriti in un dizionario che va a sovrascrivere i default solo di quelli specificati.



#### 6.2.14. Sorgenti e licenza

Il codice sorgente è disponibile sotto licenza Creative Commons BY-NC-SA 4.0<sup>68</sup> ed è accessibile pubblicamente tramite il repository GitHub<sup>69</sup> raggiungibile all'indirizzo: [https://github.com/gastaldelloalberto/mopo16s\\_webapp](https://github.com/gastaldelloalberto/mopo16s_webapp).

### 6.3. Singularity e Docker

Grazie all'integrazione esistente tra i due sistemi di virtualizzazione, è possibile trasformare le immagini Docker in quelle per Singularity. Questo permette di semplificare il processo di containerizzazione, dal momento che quasi tutti i componenti utilizzati sono diventati standard per gran parte delle applicazioni web. La strada da intraprendere quindi risulta nel creare immagini Docker per poi trasformarle in SIF.

Le immagini necessarie per il nostro progetto sono quelle di nginx<sup>70</sup>, PostgreSQL<sup>71</sup>, Redis<sup>72</sup> e Python<sup>73</sup>, tutte disponibili in versione ufficiale all'interno dell'hub di Docker.

Per quanto riguarda Django e Celery è consigliato utilizzare l'immagine di Python come base di partenza, per poi installare i pacchetti necessari. Si fornisce di seguito un esempio di Docker file per Django.

```
FROM python:3.7-alpine

RUN apk update && apk add build-base py3-gunicorn

WORKDIR /mopo16s

COPY mopo16s_webapp_proj .
COPY requirements.txt .

RUN pip install -r requirements.txt

CMD ["/usr/local/bin/gunicorn", "--workers 2", "mopo16s_webapp_proj.wsgi:application"]
```

Figura 14 - Esempio di file Docker per la creazione di un'immagine

<sup>68</sup> <https://creativecommons.org/licenses/by-nc-sa/4.0>

<sup>69</sup> <https://github.com>

<sup>70</sup> [https://hub.docker.com/\\_/nginx](https://hub.docker.com/_/nginx)

<sup>71</sup> [https://hub.docker.com/\\_/postgres](https://hub.docker.com/_/postgres)

<sup>72</sup> [https://hub.docker.com/\\_/redis](https://hub.docker.com/_/redis)

<sup>73</sup> [https://hub.docker.com/\\_/python/](https://hub.docker.com/_/python/)



## Capitolo 7 Valutazione e collaudo

In questo capitolo sono presentate le modalità attraverso le quali si sono effettuati i test dell'applicazione.

Nella sezione 7.1 si descrive la macchina in cui è stato installato il sistema per il collaudo, nella 7.2 sono elencate le procedure seguite per la creazione dei database di sequenze e dei set primer per eseguire i test.

La sezione 7.3, che vuole essere una guida per l'utente finale, presenta l'applicazione web con il supporto di *screenshot*, spiegando passo passo le operazioni da eseguire per servirsi di ogni funzionalità.

### 7.1. Ambiente di test

Per permettere uno scorrevole sviluppo e per effettuare contestualmente i test necessari, si è creato un ecosistema di collaudo con la virtualizzazione, installando Debian in una macchina virtuale, con tutti i componenti descritti nella sezione 6.1.

Dopo aver effettuato le configurazioni necessarie alla comunicazione tra i vari sistemi è stato compilato il codice sorgente di mopo16S ed è stata eseguita una run di prova direttamente da terminale.

Per poter accedere all'applicazione anche da remoto, è stato creato un nome di dominio provvisorio ed impostato per indirizzare ad una determinata porta del router nella cui rete era in esecuzione la macchina. Questo a sua volta è stato settato per inoltrare le connessioni verso la porta del server WSGI. Tale accorgimento è stato di fondamentale importanza per permettere test efficaci sull'applicazione.

L'applicazione necessita di un server di posta per l'invio delle email. Si è provveduto a creare un'utenza in uno dei maggiori provider di posta elettronica per assolvere

temporaneamente a questa esigenza, ed effettuare i dovuti test con tutte le funzioni abilitate.

## 7.2. File di test

Per concretizzare le prove con l'applicazione sono stati caricati nel sistema molteplici file d'esempio, creati seguendo le istruzioni fornite con mopo16S e i due script scaricabili.

Questi ultimi sono stati leggermente modificati per correggerne in qualche punto la logica e renderli più chiari e funzionali permettendo la creazione di file con configurazioni differenti in una singola esecuzione.

Il primo script, scritto in Bash, scarica in automatico il database di GreenGenes del 2013 ed aggiorna i file contenenti le informazioni sulla tassonomia in base ad alcune regole. Il secondo, scritto nel linguaggio di programmazione  $R^{74}$ , è stato diviso in due parti: una crea i set di sequenze 16S per ogni livello tassonomico disponibile includendo solo quelle batteriche; l'altra crea le coppie di primer con diverse lunghezze minime e massime.

I file con le sequenze caricati sono quelli richiesti nei requisiti, sono quindi già pronti per quando il sistema andrà in produzione ed hanno il flag di curato.

Non essendo esperti nel campo della metagenomica i file dei primer generati sono chiaramente solamente a scopo di test e non vogliono rappresentare alcun esperimento effettivo.

Sono stati creati molteplici lavori per accertare il regolare funzionamento della parte del sistema con il compito di avviare le esecuzioni di mopo16S, simulando anche qualche fallimento per controllare la correttezza della gestione degli errori.

---

<sup>74</sup> <https://www.r-project.org/about.html>

## 7.3. Demo e guida per l'utente

Si elencano in questa sezione i vari passaggi da compiere per fare un corretto uso delle varie funzionalità offerte dall'applicazione. Questa sezione vuole anche essere una veloce guida illustrativa per l'utente finale.

### 7.3.1. Registrazione e login

Per utilizzare l'applicazione è necessario registrarsi fornendo un username, un'email e una password; successivamente l'autenticazione avviene tramite username e password. In caso di password dimenticata è possibile richiedere un reset della stessa, che può essere effettuato seguendo le istruzioni che vengono inviate tramite email.



**mopo16S Webapp**

Sign up

Username:

Required: 150 characters or fewer. Letters, digits and @/!/+/-/\_ only.

Email:

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

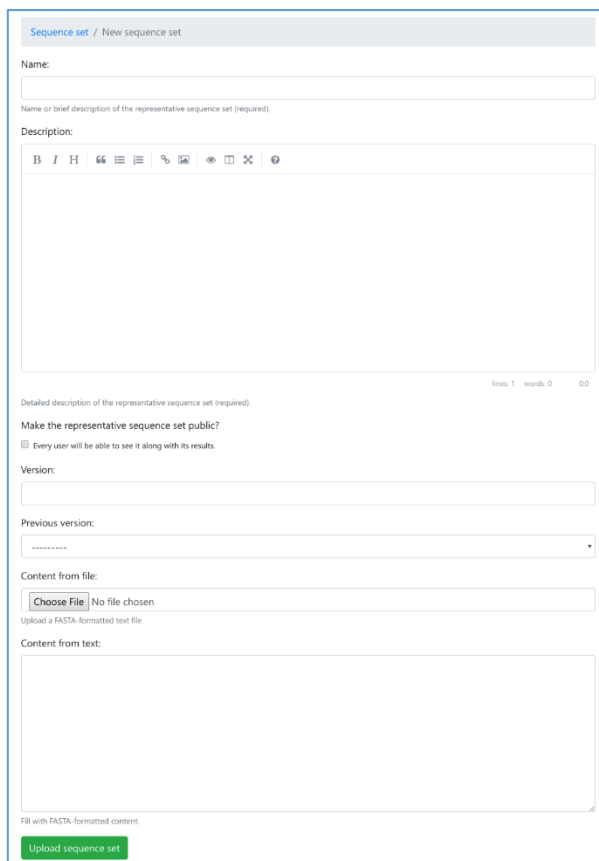
Password confirmation:

Enter the same password as before, for verification.

[Create an account](#)

[Already have an account? Log in](#)

Figura 15 - Form di registrazione



Sequence set / New sequence set

Name:

Name or brief description of the representative sequence set (required).

Description:

Detailed description of the representative sequence set (required). lines: 1 words: 0 0.0

Make the representative sequence set public?  
 Every user will be able to see it along with its results.

Version:

Previous version:

Content from file:  
 No file chosen  
Upload a FASTA-formatted text file.

Content from text:

Fill with FASTA-formatted content.

[Upload sequence set](#)

### 7.3.2. Caricamento sequenze

Il sistema propone set di sequenze 16S precaricate ma, se necessario, l'utente può fare l'upload di quelle in suo possesso tramite un semplice form. Deve specificare un nome esplicitivo, una descrizione dettagliata del contenuto, la versione dei dati con un eventuale riferimento alla versione precedente (se già presente nel sistema), ed infine allegare un file o incollare i dati testuali nell'apposita casella.

Figura 16  
Form per il caricamento di un set di sequenze

### 7.3.3. Caricamento primer

Le modalità di caricamento dei file contenenti le coppie di primer sono uguali a quelle relative alle sequenze. L'unica differenza è che il sistema non propone set già pronti all'uso, data la natura sperimentale di questi dati.

### 7.3.4. Creazione job

L'utente può creare le richieste di esecuzione tramite l'apposito form che richiede, oltre al nome e alla descrizione, di selezionare dalle risorse già caricate nel sistema un file di sequenze e uno di coppie di primer. I parametri opzionali relativi a mopo16S sono presentati con i valori di default già impostati, l'utente può scegliere se lasciarli invariati o adattarli alle proprie esigenze.

Figura 17 (a destra) - Form per la creazione del job con tutti i parametri modificabili all'occorrenza

Figura 18 (in basso) - Visualizzazione della lista dei job sottomessi

Jobs / New job

Name:

Name or brief description of the job (required):

Description:

Detailed description of the job (required):

Make the job public?  
 Every user will be able to see it along with its results.

Good pairs:

Rep set:

seed:

Seed of the random number generator (default 0):

restarts:

Number of restarts of the multi-objective optimisation algorithm (default 20):

runs:

Number of runs of the multi-objective optimisation algorithm (default 20):

maxMismatches:

Maximum number of mismatches between the non-3'-end of the primer and a 16S sequence to consider the latter covered by the primer, in case also the 3'-end perfectly matches (default 2):

maxALenSpanC:

Maximum amplicon length: span considered when computing coverage (half above, half below median) (default 200):

minPrimerLen:

Minimum primer length (default 17):

maxPrimerLen:

Maximum primer length (default 21):

minTm:

Minimum primer melting temperature (default 52):

minGCCont:

Minimum primer GC content (default 0.5):

maxGCCont:

Maximum primer GC content (default 0.7):

maxDimers:

Maximum number of self-dimers, ie of dimers between all possible gap-less alignments of the primer with its reverse complement (default 8):

maxDimersInt:

Fuzzy tolerance interval for maximum number of self dimers (default 3):

maxHomopLen:

Maximum homopolymer length (default 4):

maxHLenInt:

Fuzzy tolerance interval for maximum homopolymer length (default 2):

maxDeltaTm:

Maximum span of melting temperatures for the primer sets (default 3):

deltaTmInt:

Fuzzy tolerance interval for span of melting temperatures of the primer set (default 2):

maxALenSpanE:

Maximum span between median and given quantile (maxALenSpanEQ) of amplicon length (default 50):

maxALenSpanEQ:

Quantile of amplicon length (default 0.01):

minInterv:

Fuzzy tolerance interval for minimum melting temperature (default 2):

minGCContInt:

Fuzzy tolerance interval for minimum GC content (default 0.1):

Start job

mopo16S Webapp Jobs Sequences Primers Mopo16S gasta

Jobs

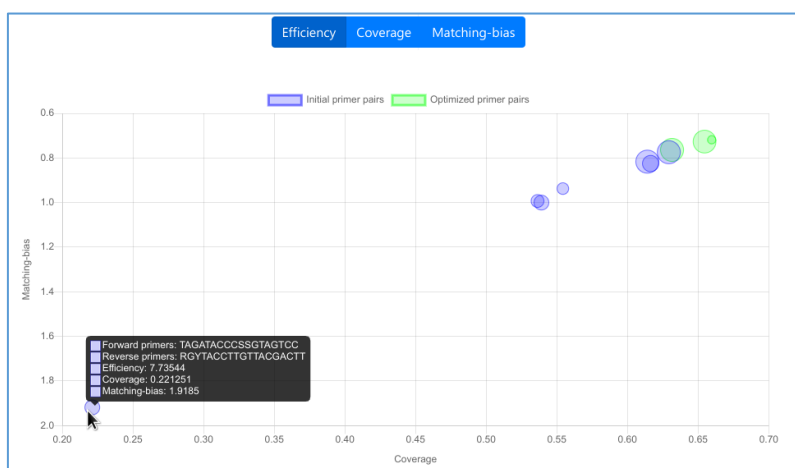
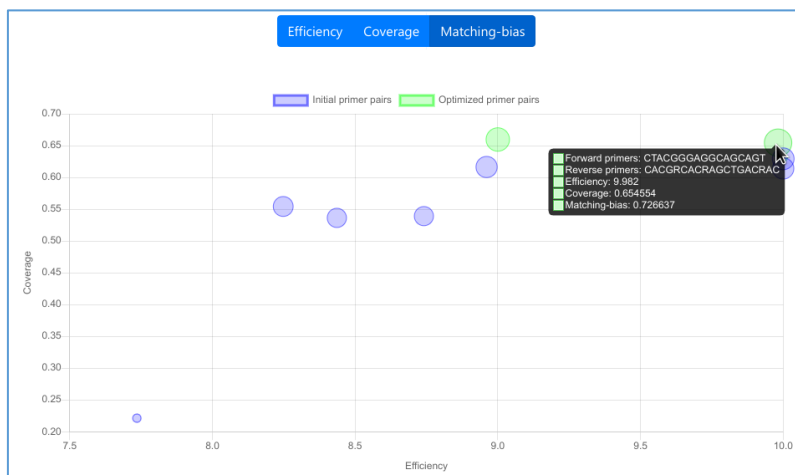
Create new job

Name	Description	Creator	Public	Created	Status
Sample job 9	Sample job 9	dario	<input checked="" type="checkbox"/>	2 weeks, 5 days ago Nov. 15, 2019, 11:58 a.m.	completed
Sample job 8	Sample job 8	chiara	<input checked="" type="checkbox"/>	3 weeks ago Nov. 13, 2019, 7:16 a.m.	completed
Sample job 7	Sample job 7	gasta	<input type="checkbox"/>	3 weeks ago Nov. 13, 2019, 7:11 a.m.	completed
Sample job 6	Sample job 6	chiara	<input type="checkbox"/>	3 weeks ago Nov. 13, 2019, 7:05 a.m.	completed
Sample job 5	Sample job 5	andre	<input checked="" type="checkbox"/>	3 weeks, 2 days ago Nov. 11, 2019, 10:04 p.m.	completed
Sample job 4	Sample job 4	dario	<input type="checkbox"/>	3 weeks, 2 days ago Nov. 11, 2019, 9:21 p.m.	completed
Sample job 3	Sample job 3	gasta	<input type="checkbox"/>	3 weeks, 2 days ago Nov. 11, 2019, 9:21 p.m.	completed
Sample job 2	Sample job 2	andre	<input checked="" type="checkbox"/>	3 weeks, 2 days ago Nov. 11, 2019, 8:59 p.m.	completed
Sample job 1	Sample job 1	gasta	<input type="checkbox"/>	3 weeks, 2 days ago Nov. 11, 2019, 5:52 p.m.	completed

### 7.3.5. Visualizzazione e download risultati

Nella tab relativa ai risultati l'utente può interagire con i grafici e le tabelle, si riportano alcuni screenshot per maggiore chiarezza.

Per quanto riguarda il grafico è possibile invertire gli assi con gli appositi pulsanti e, se necessario, scaricarne l'immagine. Anche le tabelle sono interattive: permettono di copiare o esportare in formato pdf o excel i dati, eventualmente selezionando solo le righe necessarie.



Results

Charts Initial primer pairs Optimized primer pairs Download data

Copy PDF Excel

Search:

Forward primers	Reverse primers	Efficiency	Coverage	Matching-bias
ACCTACGGGAGGCAGCACT	CACGRCACRAGCTGACRAC	10	0.431555	0.763978
CTACGGGAGGCAGCACT	CACGGCACRAGCTGACRAC	9	0.659614	0.718524
CTACGGGAGGCAGCACT	CACGRCACRAGCTGACRAC	9.982	0.654554	0.728637

Showing 1 to 3 of 3 entries

Previous 1 Next

Results

Charts Initial primer pairs Optimized primer pairs Download data

Copy PDF Excel

Search:

Forward primers	Reverse primers	Efficiency	Coverage	Matching-bias
GGAGGCAGCTRRGGAAAT	CGACARCCATGCASCACCT	8.43616	0.536339	0.99296
GGAGGCAGCTRRGGAAAT	CGACRRCATGCANACCT	8.74113	0.539998	1.00012
GGAGGCAGCTRRGGAAAT	CACRRCACRAGCTGACRAC	8.24821	0.554278	0.937417
TAGATACCCSSGTAGTCC	RGYTACCTTGTACGACTT	7.73544	0.221251	1.9185
TCCTACGGGAGGCAGCACT	CGACARCCATGCASCACCT	10	0.614875	0.816621
TCCTACGGGAGGCAGCACT	CGACRRCATGCANACCT	8.96878	0.616379	0.825134
TCCTACGGGAGGCAGCACT	CACRRCACRAGCTGACRAC	10	0.629255	0.774351

Showing 1 to 7 of 7 entries 2 rows selected

Previous 1 Next

### 7.3.6. API web

Anche l'API è accessibile tramite interfaccia grafica, utile per eseguire test o per fare pratica. È studiata per ricevere ed inviare contenuto in formato JSON. Nella figura a lato è esposta la rappresentazione di una risorsa



Figura 20 - Esempio di richiesta non andata a buon fine

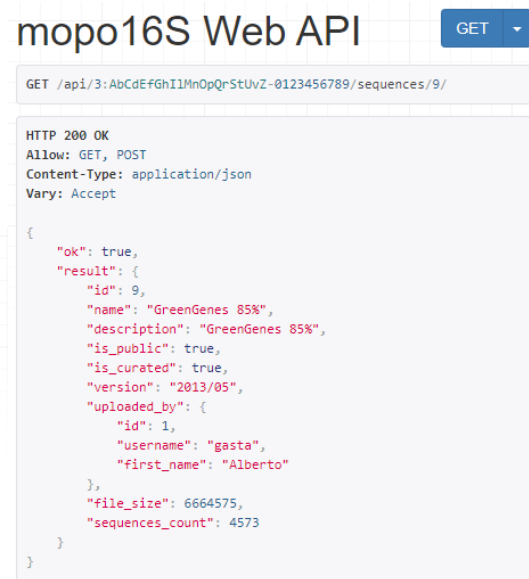


Figura 19 - Esempio di richiesta delle informazioni di un database di sequenze



## 7.4. Risultati

L'applicazione è stata sottoposta agli stakeholder per la verifica delle funzionalità e la relativa approvazione. Si è voluto in aggiunta presentare il prodotto ad alcuni utenti casuali per verificarne l'usabilità anche dal lato dei non addetti ai lavori. I feedback sono stati positivi ed hanno portato suggerimenti per la sistemazione di dettagli minori per quanto riguarda la parte grafica, che sono stati accolti ed implementati.

Durante i test il monitoraggio delle risorse di sistema è stato fondamentale per verificare la correttezza e l'efficacia dell'algoritmo sviluppato per l'assegnazione automatica del numero di thread alle singole esecuzioni di mopo16S. Dai test eseguiti l'algoritmo risulta rispettare le politiche di priorità e le modalità di esecuzione definite.

Anche l'utilizzo della memoria RAM è stato oggetto di monitoraggio, permettendo di identificare un particolare comportamento dell'applicazione: per alcune configurazioni dei parametri di input si è constatato un considerevole uso di memoria RAM da parte di mopo16S. Tale quantitativo di RAM potrebbe essere legato alle particolari configurazioni di parametri di input o ad una gestione non completa della memoria all'interno di mopo16S. Le esatte cause di tale comportamento non sono state oggetto di ulteriori indagini essendo l'argomento estraneo agli obiettivi primari del progetto.



## Capitolo 8 Conclusioni e lavoro futuro

In questo capitolo finale sono presentati conclusioni, alcuni accorgimenti implementativi e possibile lavoro futuro.

Nella sezione 8.1 si descrivono le motivazioni da cui è nato questo lavoro e i suoi aspetti innovativi soprattutto in merito al contesto in cui è inserito, nella 8.2 sono indicati alcuni aspetti a cui prestare attenzione nella messa in produzione del sistema, infine nella sezione 8.3 si presentano dei possibili sviluppi futuri dell'applicazione.

### 8.1. Conclusioni

È stata realizzata un'applicazione web innovativa per la comunità scientifica che si occupa della metagenomica. Essa permette di raccogliere i risultati delle ricerche fatte e di perfezionare quelle in corso rendendo accessibile lo strumento mopo16S e le sue funzionalità. Questo lavoro viene incontro alle esigenze di utenti appartenenti ad un contesto interdisciplinare e pone le basi per sviluppi futuri.

L'accesso, che può avvenire sia tramite interfaccia sia in modo programmatico, offre la possibilità di sviluppare un'applicazione di terze parti, ad esempio mobile, che ne ampli le funzionalità e le modalità di utilizzo. Un ulteriore elemento innovativo è dato dalla struttura api rest dell'applicazione.

La webapp si propone di essere uno spazio di ricerca sia pubblico che privato, con la possibilità di condividere i propri risultati con il resto della comunità scientifica.

Questi aspetti si allontanano dallo standard in questo campo e aprono nuove prospettive per l'ottimizzazione del lavoro, l'interpretazione ordinata dei dati e la condivisione del sapere.

## 8.2. Deployment

Per la messa in produzione del sistema sono necessari alcuni accorgimenti da implementare una volta che l'host sarà accessibile per la configurazione. Questo perché è fondamentale conoscere le modalità in cui saranno disponibili le risorse, ad esempio se ci sarà l'esigenza di creare istanze dei servizi come PostgreSQL, Redis, nginx, o se saranno già usufruibili. Singularity e i container andranno quindi impostati di conseguenza, definendo anche quante risorse assegnare alle varie componenti dell'applicazione, in particolar modo quelle che eseguiranno i processi di mopo16S.

Altra parte integrante risiede nella configurazione particolare<sup>75</sup> richiesta da Django per portare la sicurezza ad un livello adeguato ad un ambiente di produzione. È necessario proteggere alcune impostazioni critiche che non devono essere scritte in chiaro nel codice, tra le quali: la chiave segreta per la firma crittografica<sup>76</sup>, password per l'accesso ai database, dettagli per l'invio delle email, configurazione dell'HTTPS. Solitamente le informazioni di questo genere sono passate ai sistemi e alle applicazioni tramite variabili d'ambiente, permettendo anche configurazioni dinamiche ed eventualmente dipendenti dal sistema o dal contesto.

Saranno inoltre necessari l'acquisizione di un dominio e la creazione dei certificati SSL necessari per permettere connessioni sicure via HTTPS.

Gli sviluppatori di mopo16S sono stati avvisati riguardo il non uniforme uso di memoria RAM con alcune configurazioni di parametri di input descritto nella sezione 7.4. Tale informazione permetterà loro di studiare il fenomeno e ottimizzare di conseguenza il codice in vista della messa in produzione dell'applicazione, evitando così utilizzi eccessivamente variabili di risorse nella macchina di destinazione.

---

<sup>75</sup> <https://docs.djangoproject.com/en/2.2/howto/deployment/checklist>

<sup>76</sup> [https://docs.djangoproject.com/en/2.2/ref/settings/#std:setting-SECRET\\_KEY](https://docs.djangoproject.com/en/2.2/ref/settings/#std:setting-SECRET_KEY)

### 8.3. Possibili sviluppi

La relazione *is\_completed* (si vedano gli schemi nelle sez. 5.1 e 5.2) tiene separati i concetti di job e relativo risultato. È stata fatta questa scelta per facilitare un'eventuale modifica futura nel caso sorgesse la necessità di associare più risultati allo stesso job, data la natura non-deterministica dell'algoritmo di mopo16S.

Un'altra possibile estensione è quella di permettere di caricare nell'applicazione job eseguiti in macchine locali, ad esempio computer di laboratorio, con i relativi risultati. Per farlo sarà necessario definire con gli sviluppatori di mopo16S un formato ad hoc per esportare i dati dell'esecuzione.

È da valutare un'eventuale espansione della parte di visualizzazione risultati aggiungendo altri tipi di grafici e rappresentazioni. Per farlo è necessario approfondire con qualche esperto le modalità con le quali vengono interpretati i dati in output forniti da mopo16S. Nel caso fossero utili rappresentazioni più complesse graficamente, ad esempio grafici in tre dimensioni, è possibile considerare tool che offrono più funzionalità di *chart.js*, come *d3.js*<sup>77</sup>.

Per quanto riguarda i set di sequenze "curate", è sicuramente utile preparare altri database, oltre a quelli di GreenGenes, prendendo i dati da fonti altrettanto attendibili e rendendoli disponibili una volta che il servizio sarà online.

Nei giorni immediatamente precedenti alla stampa di questo documento è stata rilasciata una nuova versione principale di Django (3.0)<sup>78</sup>, si può quindi valutare di aggiornare l'applicazione a questa release. A tale fine è necessario controllare le note di rilascio e la guida all'upgrade disponibile sul sito<sup>79</sup>.

---

<sup>77</sup> <https://d3js.org>

<sup>78</sup> <https://docs.djangoproject.com/en/3.0/releases/3.0>

<sup>79</sup> <https://docs.djangoproject.com/en/3.0/howto/upgrade-version>



## Bibliografia

- [1] World Wide Web Consortium (W3C), «Web Services Architecture,» 11 Febbraio 2004. [Online]. Available: <https://www.w3.org/TR/ws-arch/>. [Consultato il giorno Settembre 2019].
- [2] «Web Services Glossary,» 11 Febbraio 2004. [Online]. Available: <https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice>. [Consultato il giorno Settembre 2019].
- [3] R. T. Fielding, «Architectural Styles and the Design of Network-based Software Architectures,» 2000. [Online]. Available: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>. [Consultato il giorno Settembre 2019].
- [4] R. T. Fielding, «Principled Design of the Modern Web Architecture,» Maggio 2002. [Online]. Available: <https://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf>. [Consultato il giorno Settembre 2019].
- [5] G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), 1995.
- [6] Python Software Foundation, «Python Language Reference, version 3.7,» 27 Giugno 2018. [Online]. Available: <https://docs.python.org/3.7/>. [Consultato il giorno Settembre 2019].
- [7] E. F. Codd, «A relational model of data for large shared data banks,» *Communications of the ACM*, pp. 377-387, 6 Giugno 1970.
- [8] E. F. Codd, «Relational database: A practical foundation for productivity,» *Communications of the ACM*, pp. 109-117, Febbraio 1982.
- [9] «ISO 9075:1987, Information processing systems — Database language — SQL,» [Online]. Available: <https://www.iso.org/standard/16661.html>.
- [10] «ISO/IEC 9075-2:2016/COR 1:2019, Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation) — Technical Corrigendum 1,» [Online]. Available: <https://www.iso.org/standard/75936.html>.
- [11] World Wide Web Consortium (W3C), «HTML Living Standard,» [Online]. Available: <https://html.spec.whatwg.org/multipage/>. [Consultato il giorno Novembre 2019].
- [12] World Wide Web Consortium (W3C), «HTML 4.01 Specification,» 27 Marzo 2018. [Online]. Available: <https://www.w3.org/TR/2018/SPSD-html401-20180327/>. [Consultato il giorno Novembre 2019].

- [13] World Wide Web Consortium (W3C), «XHTML™ 2.0,» 16 Dicembre 2010. [Online]. Available: <http://www.w3.org/TR/2010/NOTE-xhtml2-20101216>. [Consultato il giorno Settembre 2019].
- [14] World Wide Web Consortium (W3C), «HTML 5.1 2nd Edition,» 3 Ottobre 2017. [Online]. Available: <https://www.w3.org/TR/2017/REC-html51-20171003/>. [Consultato il giorno Novembre 2019].
- [15] World Wide Web Consortium (W3C), «Cascading Style Sheets (CSS) — The Official Definition,» [Online]. Available: <https://www.w3.org/TR/CSS/#css>. [Consultato il giorno Novembre 2019].
- [16] «Standard ECMA-262,» 3 Luglio 2017. [Online]. Available: <https://www.ecma-international.org/publications/standards/Ecma-262.htm>. [Consultato il giorno Novembre 2019].
- [17] «Netscape and Sun announce JavaScript,» 4 Dicembre 1995. [Online]. Available: <https://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newrelease67.html>.
- [18] «Web application framework,» 23 Luglio 2015. [Online]. Available: [https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web\\_applications\\_framework](https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web_applications_framework). [Consultato il giorno Settembre 2019].
- [19] «Django 2.2.3 release notes,» 1 Luglio 2019. [Online]. Available: <https://docs.djangoproject.com/en/2.2/releases/2.2.3/>. [Consultato il giorno Settembre 2019].
- [20] «The DCI Architecture: A New Vision of Object-Oriented Programming,» 20 Marzo 2009. [Online]. Available: [https://web.archive.org/web/20090323032904/https://www.artima.com/articles/dci\\_vision.html](https://web.archive.org/web/20090323032904/https://www.artima.com/articles/dci_vision.html). [Consultato il giorno Settembre 2019].
- [21] M. Shapiro, «Structure and Encapsulation in Distributed Systems: the Proxy Principle,» in *Int. Conf. on Distr. Comp. Sys. (ICDCS)*, Cambridge, MA, USA, United States, IEEE, 1986, pp. 198-204.
- [22] «PEP 3333 -- Python Web Server Gateway Interface v1.0.1,» 26 Settembre 2010. [Online]. Available: <https://www.python.org/dev/peps/pep-3333/>. [Consultato il giorno Settembre 2019].
- [23] T. M. Connolly e C. E. Begg, *Database Systems – A Practical Approach to Design Implementation and Management*, Pearson, 2014.
- [24] A. Brown, «Modelling a Real-World System and Designing a Schema to Represent It,» in *Douque and Nijssen*, North-Holland, 1975.



- [25] P. Chen, «The Entity-Relationship Model - Toward a Unified View of Data,» *ACM Transactions on Database Systems*, Marzo 1976.
- [26] American National Standards Institute (ANSI), «ANSI/X3/SPARC Study Group on Data Base Management Systems; Interim Report,» *FDT(Bulletin of ACM SIGMOD)*, 1975.
- [27] «PostgreSQL 12.1, 11.6, 10.11, 9.6.16, 9.5.20, and 9.4.25 Released!,» 14 11 2019. [Online]. Available: <https://www.postgresql.org/about/news/1994/>.
- [28] «PostgreSQL 12.1 Documentation,» 2019. [Online]. Available: <https://www.postgresql.org/docs/12/index.html>.
- [29] «Redis 5.0 release notes - Redis 5.0.7 Released,» 19 Novembre 2019. [Online]. Available: <https://raw.githubusercontent.com/antirez/redis/5.0/00-RELEASENOTES>.
- [30] F. Bäckhed, R. E. Ley, J. L. Sonnenburg, D. A. Peterson e J. I. Gordon, «Host-Bacterial Mutualism in the Human Intestine,» *Science*, vol. 307, n. 5717, pp. 1915-1920, 25 Marzo 2005.
- [31] P. J. Turnbaugh, R. E. Ley, M. Hamady, C. M. Fraser-Liggett, R. Knight e J. I. Gordon, «The Human Microbiome Project,» *Nature*, n. 449, pp. 804-810, 17 Ottobre 2007.
- [32] R. E. Ley, D. A. Peterson e J. I. Gordon, «Ecological and Evolutionary Forces Shaping Microbial Diversity in the Human Intestine,» *Cell*, vol. 124, n. 4, pp. 837-848, 24 Febbraio 2006.
- [33] F. Sambo, F. Finotello, E. Lavezzo, G. Baruzzo, G. Masi, E. Peta, M. Falda, S. Toppo, L. Barzon e B. Di Camillo, «Optimizing PCR primers targeting the bacterial 16S ribosomal RNA gene,» *BMC Bioinformatics*.
- [34] L. DJ e P. WR, «Rapid and sensitive protein similarity searches,» *Science*, vol. 227, n. 4693, pp. 1435-1441, Marzo 1985.
- [35] P. WR e L. DJ, «Improved tools for biological sequence comparison,» *Proceedings of the National Academy of Sciences of the United States of America*, vol. 85, n. 8, pp. 2444-2448, Aprile 1988.