

# 2p-kt web IDE

**Report finale per il corso di Applicazioni e Servizi Web**

**LM Ingegneria e Scienze Informatiche**

`davide.domini@studio.unibo.it`

June 27, 2023

# Contents

<b>1</b>	<b>Requisiti e funzionalità</b>	<b>4</b>
1.1	Personas . . . . .	4
1.1.1	Caterina . . . . .	4
1.1.2	Simone . . . . .	4
1.1.3	Sara . . . . .	4
1.2	Definizione dei requisiti . . . . .	5
1.2.1	Requisiti funzionali . . . . .	5
1.2.2	Requisiti non-funzionali . . . . .	5
<b>2</b>	<b>Desing</b>	<b>6</b>
2.1	Architettura del backend . . . . .	6
2.1.1	Web server . . . . .	6
2.1.2	Servizio tuProlog . . . . .	6
2.2	Architettura del frontend . . . . .	6
2.2.1	Componenti . . . . .	6
2.2.2	Comunicazioni con il backend . . . . .	7
2.2.3	Stato interno . . . . .	7
2.3	Struttura del database . . . . .	7
2.4	Interfaccia utente . . . . .	7
2.4.1	Accesso e registrazione . . . . .	7
2.4.2	Code area . . . . .	8
<b>3</b>	<b>Tecnologie</b>	<b>10</b>
3.1	Tecnologie backend . . . . .	10
3.1.1	Node.js . . . . .	10
3.1.2	Express . . . . .	10
3.1.3	Mongoose . . . . .	10
3.1.4	Axios . . . . .	10
3.1.5	RabbitMQ . . . . .	11
3.1.6	Kotlin . . . . .	11
3.2	Tecnologie database . . . . .	11
3.2.1	MongoDB . . . . .	11
3.3	Tecnologie frontend . . . . .	11
3.3.1	Vue.js . . . . .	11
3.3.2	PrimeVue . . . . .	12
<b>4</b>	<b>Test</b>	<b>13</b>
4.1	Postman . . . . .	13
4.2	Test con utenti . . . . .	13
<b>5</b>	<b>Deployment</b>	<b>14</b>
<b>6</b>	<b>Usare il web ide</b>	<b>15</b>



# 1 Requisiti e funzionalità

## 1.1 Personas

Al fine di delineare nello specifico i requisiti, si è deciso di creare delle Personas che siano rappresentative del target che si vuole raggiungere con l'applicativo.

### 1.1.1 Caterina

- Nome: Caterina;
- Età: 23 anni;
- Lavoro: studentessa di ingegneria informatica;
- Single, fuori sede;
- Abitudini: per motivi di studio Caterina deve studiare la programmazione logica, per questo motivo necessita di avere un IDE che le permetta di testare velocemente gli esempi con cui le vengono spiegati i concetti di teoria.

### 1.1.2 Simone

- Nome: Simone;
- Età: 26 anni;
- Lavoro: dottorando in matematica;
- Contesto familiare: single;
- Abitudini: durante le sue ricerche Simone fa spesso uso di programmazione logica.

### 1.1.3 Sara

- Nome: Sara;
- Età: 52 anni;
- Lavoro: professoressa universitaria;
- Contesto familiare: sposata con due figli;
- Abitudini: Sara insegna nel corso di "Paradigmi di programmazione avanzata", questo corso comprende anche la programmazione logica, per questo motivo necessita di un IDE con cui spiegare durante le lezioni.

## **1.2 Definizione dei requisiti**

### **1.2.1 Requisiti funzionali**

- Permettere di registrare un nuovo account;
- Permettere di accedere al proprio account;
- Permettere di scrivere una nuova teoria prolog;
- Permettere di indicare alcuni parametri relativi alle query, ovvero: i) massimo tempo di timeout, e ii) massimo numero di soluzioni;
- Permettere di risolvere una query in un'unica volta;
- Permettere di richiedere solo la prossima soluzione della query;
- Permettere di resettare la computazione.

### **1.2.2 Requisiti non-funzionali**

I requisiti non funzionali riguardano le caratteristiche e le proprietà che il sistema deve avere e in questo caso la web application deve:

- Essere utilizzabile agilmente sia in visualizzazione desktop che mobile;
- Essere intuitiva;
- Essere accessibile;

## 2 Desing

### 2.1 Architettura del backend

Il backend si è formato da due componenti principali, ovvero: il web server e il servizio tuProlog. Il workflow per risolvere una certa query è il seguente:

- Il web server riceve dal frontend la richiesta per computare le soluzioni della query;
- Il web server inoltra la richiesta al servizio kotlin sfruttando la sua API;
- Una volta ricevuta la risposta dal servizio kotlin, il web server inoltra la soluzione al frontend utilizzando Socket.io.

Il backend offre anche le funzionalità di login e registrazione di un nuovo utente.

#### 2.1.1 Web server

Il server è stato implementato usando *Node.js* ed *Express*. Per quanto riguarda la persistenza dei dati si interfaccia con un database MongoDB.

Il design del servizio di backend è basato su una API RESTful, per cui ogni funzionalità è accessibile mediante una richiesta HTTP ben costruita, in termini di metodo della richiesta e URI.

#### 2.1.2 Servizio tuProlog

Il servizio tuProlog è stato realizzato utilizzando Kotlin e, tramite Ktor, espone una API RESTful che permette di accedere all'interfaccia di un *Solver*, componente necessario per risolvere query prolog data una certa teoria.

### 2.2 Architettura del frontend

Il client è stato sviluppato utilizzando il framework Vue.js; è possibile discutere della sua architettura concentrandosi su tre parti principali: i componenti, le comunicazioni con il backend e lo stato interno.

#### 2.2.1 Componenti

I componenti vanno a costituire l'interfaccia grafica dell'applicazione. Nel caso del sistema sviluppato sono presenti 3 componenti principali:

- **signin**: componente che effettua l'accesso all'account dell'utente;
- **signup**: componente che permette all'utente di registrarsi nel sistema;
- **code area**: componente che permette di scrivere teorie prolog e risolvere query.

### 2.2.2 Comunicazioni con il backend

Le comunicazioni con il backend sono gestite utilizzando:

- **Axios**: per interfacciarsi con l'API RESTful del backend;
- **Socket.io**: per ricevere i risultati delle query.

### 2.2.3 Stato interno

Lo stato interno del frontend è stato gestito utilizzando il componente *LocalStorage* di Vue.js, nello specifico vengono salvati token, id e username dell'utente ottenuti con il login nel sistema.

## 2.3 Struttura del database

Utilizzando lo stack MEVN, il database è di tipo document-based implementato con MongoDB, ed è costituito da una collezione di documenti:

- **Users**: ogni documento di questa collezione contiene l'anagrafica relativa esclusivamente ad uno specifico utente.

## 2.4 Interfaccia utente

### 2.4.1 Accesso e registrazione

Come si può notare dalle immagini 1 e 2, lo stile delle pagine di signin e signup è stato tenuto il più semplice e minimalista possibile, questo per permettere all'utente di avere un'esperienza lineare con la piattaforma.

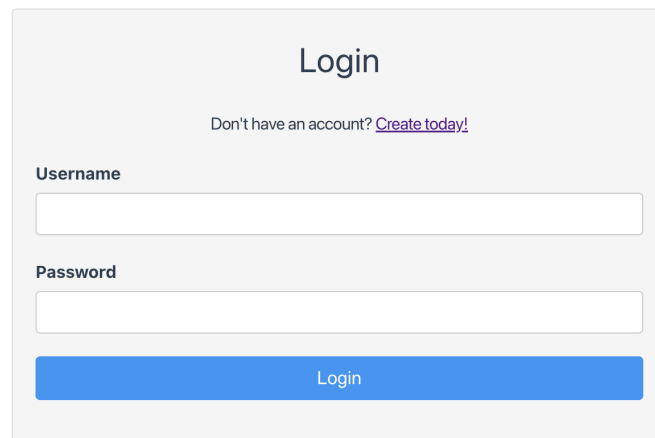
A login form with a light gray background. At the top, the word "Login" is centered in a dark gray font. Below it, a link "Don't have an account? [Create today!](#)" is centered in a smaller font. The form contains two input fields: "Username" and "Password", both with light gray borders. Below the "Password" field is a solid blue button with the text "Login" in white.

Figure 1: Signin

The image shows a web form titled "Signup". It contains five input fields, each with a label above it: "Name", "Surname", "Email", "Username", and "Password". Below these fields is a blue button with the text "Signup". The form is set against a light gray background.

Figure 2: Signup

#### 2.4.2 Code area

La pagina principale dell'applicativo (figura 3) è quella che permette di programmare in prolog, questa è stata strutturata per richiamare immediatamente l'attenzione dell'utente sull'editor per la teoria prolog e sul componente che espone i risultati delle query. Sono presenti poi i componenti secondari che permettono di: inserire una nuova query ed impostare alcuni parametri necessari per la computazione dei risultati.



[-> Logout]

1 Code goes here...

Insert the query here...

Timeout (ms): 50

Max solutions: 10

Solve Next

Solve All

Reset

Query result:

Figure 3: Code area

## 3 Tecnologie

### 3.1 Tecnologie backend

#### 3.1.1 Node.js

Node.js è un ambiente di runtime open-source che consente di eseguire codice JavaScript lato server. Basato sul motore JavaScript V8 di Google, Node.js offre una piattaforma scalabile e ad alte prestazioni per lo sviluppo di applicazioni web e server. Ciò che rende Node.js unico è la sua natura asincrona e orientata agli eventi, che consente di gestire efficacemente un grande numero di connessioni simultanee senza bloccare il thread principale.

In questo progetto è stato usato per creare un server in event-loop responsivo in grado di gestire le richieste in modo asincrono e non bloccante.

#### 3.1.2 Express

Express è un framework web leggero e flessibile per Node.js. Serve come strumento per semplificare la creazione di applicazioni web e API REST. Express fornisce una serie di funzionalità e middleware che aiutano gli sviluppatori a gestire il routing, la gestione delle richieste e delle risposte, la gestione degli errori e la creazione di endpoint API. Con Express, è possibile creare rapidamente server web efficienti e scalabili utilizzando JavaScript.

#### 3.1.3 Mongoose

Mongoose è una libreria di Object-Document Mapping (ODM) per Node.js e MongoDB. Serve come strumento per semplificare l'interazione con il database MongoDB utilizzando JavaScript, fornisce un set di funzionalità che consentono di definire modelli di dati, creare schemi, eseguire operazioni di query e gestire la validazione dei dati. Attraverso Mongoose, è possibile eseguire operazioni CRUD (Create, Read, Update, Delete) in modo più semplice e intuitivo, evitando la necessità di scrivere query MongoDB direttamente. Inoltre, offre funzionalità aggiuntive come popolamento, middleware e gestione delle relazioni tra i dati, semplificando lo sviluppo di applicazioni basate su MongoDB con Node.js.

#### 3.1.4 Axios

Axios è una libreria JavaScript basata su promise che consente di effettuare richieste HTTP da browser o da server utilizzando Node.js. Serve come strumento per semplificare l'interazione con API esterne o per comunicare con un backend. Axios fornisce un'interfaccia semplice ed intuitiva per eseguire richieste GET, POST, PUT, DELETE e altre operazioni comuni. Supporta anche l'intercettazione di richieste e risposte, l'invio di dati formattati come JSON e la gestione degli errori in modo elegante.

### 3.1.5 RabbitMQ

RabbitMQ è un sistema di messaggistica open-source basato su protocollo AMQP (Advanced Message Queuing Protocol). Serve come middleware per facilitare la comunicazione asincrona tra applicazioni distribuite. RabbitMQ gestisce la coda dei messaggi, garantendo la consegna affidabile e l'elaborazione scalabile dei messaggi tra i diversi componenti del sistema. È particolarmente utile in scenari di microservizi, dove diverse applicazioni devono scambiarsi dati in modo affidabile e scalabile. RabbitMQ supporta anche modelli di messaggistica avanzati come il publish-subscribe, il request-reply e il routing basato su pattern, consentendo una maggiore flessibilità nella progettazione dell'architettura di un sistema distribuito.

In questo progetto è stato utilizzato per implementare la comunicazione asincrona fra il web server e il servizio kotlin per tuProlog.

### 3.1.6 Kotlin

Kotlin è un linguaggio di programmazione moderno e versatile che si esegue sulla piattaforma Java. Serve come alternativa a Java, offrendo una sintassi più concisa, un sistema di tipi statico e una migliore interoperabilità con il codice Java esistente.

In questo progetto è stato utilizzato per implementare il servizio kotlin per tuProlog.

## 3.2 Tecnologie database

### 3.2.1 MongoDB

MongoDB è un database NoSQL orientato ai documenti. Serve come soluzione di archiviazione dati flessibile e scalabile per applicazioni web e mobile. Consente di memorizzare dati in documenti JSON, offrendo una struttura di dati dinamica e adattabile. È caratterizzato da una rapida velocità di scrittura e lettura dei dati e da una scalabilità orizzontale semplice da implementare. Spesso viene utilizzato per gestire grandi volumi di dati non strutturati o semi-strutturati, ad esempio dati di sensori, log o profili utente.

## 3.3 Tecnologie frontend

### 3.3.1 Vue.js

Vue.js è un framework JavaScript progressivo e leggero per la creazione di interfacce utente interattive. Serve come strumento per la creazione di applicazioni web a singola pagina (SPA) o per l'integrazione di componenti interattivi nelle pagine esistenti. Vue.js facilita la gestione dello stato dell'applicazione, consentendo la creazione di componenti riutilizzabili e la gestione reattiva dei dati. Supporta la creazione di interfacce dinamiche attraverso la sua sintassi dichiarativa e il sistema di template. Inoltre, offre un'ampia gamma di librerie e plugin che estendono le sue funzionalità di base, consentendo di sviluppare applicazioni complesse e scalabili.

### 3.3.2 PrimeVue

PrimeVue è una libreria di componenti UI per Vue.js. Serve come strumento per semplificare la creazione di interfacce utente ricche e moderne. Offre un'ampia gamma di componenti predefiniti, come bottoni, tabelle e molto altro, che possono essere facilmente integrati nelle applicazioni. Oltre ai componenti, PrimeVue fornisce anche funzionalità come il supporto per il drag and drop, la gestione degli eventi e la validazione dei form. La libreria è altamente personalizzabile e supporta temi personalizzati per adattarsi alle esigenze di progetto.

## **4 Test**

### **4.1 Postman**

Postman è uno strumento estremamente pratico per testare le API REST. Prima di essere utilizzate dal client, le API fornite dal server sono state sottoposte a test tramite Postman al fine di verificare il loro comportamento, sia in caso di successo che in caso di errore. Questo ci ha permesso di lavorare con API testate, evitando la necessità di individuare successivamente se una risposta di errore fosse causata dal server o dal client.

### **4.2 Test con utenti**

Una volta completata la realizzazione, abbiamo chiesto a degli utenti che non erano coinvolti nello sviluppo (e quindi non erano soggetti a pregiudizi cognitivi come quelli che sviluppano i programmatori) di provare la piattaforma. Dai loro feedback abbiamo potuto capire che alcuni aspetti dell'interfaccia non erano chiari e non aiutavano gli utenti a comprendere pienamente come utilizzarla. Ad esempio, i nomi dei pulsanti precedenti per calcolare le soluzioni risultavano ambigui e confondevano gli utenti riguardo al comportamento effettivo dell'IDE. Inoltre, osservando gli utenti durante l'uso della piattaforma, siamo riusciti a individuare alcuni errori di programmazione che erano sfuggiti in precedenza.

## 5 Deployment

L'applicativo è costituito da quattro componenti principali: *backend*, *frontend*, *database* e *servizio tuProlog*. Per il database si è scelto di utilizzare *Atlas*, ovvero un servizio per il deployment di mongo in cloud. Per gli altri componenti, invece, è stato creato un docker file ed è stato poi utilizzato il tool *Docker compose* per specificare il deployment in locale. Dunque per eseguire l'applicativo è sufficiente:

- Clonare il repository dal seguente link GitHub: <https://github.com/davidedomini/2p-kt-web-ide>
- Posizionarsi all'interno della corrispondente cartella
- Lanciare il comando `docker compose up --build -d`

## 6 Usare il web ide

Per utilizzare il web ide è necessario compiere i seguenti passi:

1. Registrare un nuovo account;
2. Autenticarsi con l'account appena creato;
3. Scrivere una nuova teoria prolog nell'apposito editor;
4. Scegliere i valori preferiti per i parametri `maxSol` e `timeout` (a meno di esigenze particolari è possibile lasciare quelli di default);
5. Scrivere una nuova query per la teoria scelta;
6. Computare le soluzioni usando gli appositi comandi.

Un esempio di possibile teoria prolog per testare il web ide è il seguente:

```
find ([E | _], E).  
find ([_ | T], E) :- find (T, E).
```

Su questa teoria è possibile scrivere diverse query, ad esempio elencare tutti gli elementi di una data lista:

```
find ([10, 20, 30], S).
```

## 7 Conclusioni

Grazie allo sforzo per soddisfare i requisiti minimi dell'applicazione, ho potuto approfondire diverse tecnologie estremamente versatili, acquisendo una conoscenza che può essere applicata anche in altri settori dell'informatica. Ad esempio, sviluppare interfacce grafiche che si interfacciano ad un'API e gestire un database non relazionale. Lo stack MEVN utilizzato ha permesso di scrivere codice conciso e in maniera piuttosto veloce, senza troppe ripetizioni.

Il risultato finale rispecchia pienamente la visione originale dell'applicazione proposta, e non è esclusa la possibilità di espanderne le funzionalità in futuro. Inoltre, i test eseguiti hanno mostrato un corretto funzionamento di tutte le principali attività offerte dall'applicazione.

Alcuni possibili ambiti di miglioramento includono ad esempio implementare funzionalità di tuProlog più avanzate e una migliore gestione del profilo utente.