Reqwhy

Applicazioni e Servizi Web

Alberto Donati - 000979599 {alberto.donati6@studio.unibo.it}

7 novembre 2021

Indice

Capitolo 1

Introduzione

Scopo del progetto è creare un sito web che sia una sorta di Quora, creato apposta per gli studenti universitari. Gli studenti potranno così fare domande alla community, su diversi ambiti, non solamente a quegli inerenti ai loro studi. Gli studenti potranno rispondere alle domande degli altri studenti. I professori saranno anche loro utenti attivi che potranno rispondere alle domande degli studenti.

Capitolo 2

Requisiti

Le principali funzionalità implementate sono:

- Gestione multiutente per Studenti e per Professori;
- Studenti possono inserire domande;
- Studenti possono mettere una sorta di "love" alle domande e toglierlo;
- Studenti possono mettere una sorta di "up" alle risposte che gli piacciono e toglierlo;
- Studenti possono segnare la risposta migliore alle loro domande (possono sceglierne poi un altra in sostituzione);
- I professori possono fare tutto quello che fanno gli studenti;
- I professori possono segnare come migliori le risposte anche alle domande degli altri;
- Possibile divisione in aree delle domande;
- Implementazione reale del progetto (tramite v.m. Azure hosted e dominio).

Capitolo 3

Design

In questo capitolo vengono descritte l'analisi dei target user, gli sketch, le interfacce, l'architettura e il database.

3.1 Analisi dei target user

Di seguito la creazione di alcune personas e la descrizione dei relativi scenari d'uso.

3.1.1 Personas

Di seguito desciverò alcune personas che potrebbero usare il sistema da me creato. Il sistema è creato per essere usato da studenti e professori. Non richiede particolari competenze e conoscenze. Il sito è stato creato per avere diverse categorie (chiamate aree) in cui porre domande.

Stefano

Stefano è un professore di Storia antica a Lettere, è prossimo alla pensione e gli piace passare le sue serate a teatro. Ha studiato materie letterarie tutta la vita e adesso si trova a dover passare diverse ore al computer per rispondere alla decine di mail che i suoi studenti gli inviano. Stefano ha una famiglia molto numerosa e non ha tempo per stare dietro a tutte le mail che gli studenti gli inviano. Non ha social network e non è molto capace ad usare il computer.

Luca

Luca è un brillante studente di matematica. Gli piacciono i problemi matematici complessi e i calcoli di integrali e derivate. E' alla fine del primo anno, ma causa Covid non è riuscito a confrontarsi con i suoi compagni di studi sui problemi irrisolti della matematica. Adora passare le ore a parlare di matematica. Oltre

alla matematica ha anche diversi hobby tra cui giocare a poker. Gli piacerebbe condividere online il suo sapere.

3.1.2 Scenari d'uso

Scenario d'uso di Stefano

Stefano ha terminato un giorno di lezione come un altro. Al termine della lezione, gli studenti hanno molte domande. Lui allora dice loro di mandargli una mail. La sera riceve decine di mail. Tra queste c'è una mail di un suo vecchio amico del DISI che gli dice che c'è questo nuovo sito in cui gli studenti e i professori possono interagire. Sebbene il professore non ami i computer, decide che potrebbe essere meno faticoso e meno dispendioso in termini di tempo usare il sito al posto delle mail.
Stefano il giorno dopo allora comunica agli studenti che tutti quelli che gli avevano inviato una mail si dovevano iscrivere al sito e pubblicare lì le loro domande. Inoltre, consiglia agli studenti di provare a rispondersi tra loro.
Stefano accende il computer verso sera e prova ad aprire il sito. Per utilizzarlo si iscrive al sito come professore. Accede poi al sistema.
Stefano si reca nella sezione che parla di storia e inizia a guardare le domande. Stefano ne trova qualcuna che gli piace e decide di mettere a queste un "love". Poi prende e guarda se qualcuno ha risposto alle domande. Stefano allora apre una domanda e vede che ci sono già molteplici risposte. Decide allora di segnare la risposta giusta come migliore. Poi prende altre risposte e quelle che comunque non gli dispiacevano decide di metterci un "up".
Prende poi un altra domanda, e vede però che la risposta segnata come migliore è in realtà leggermente sbagliata. Decide allora di rispondere lui alla domanda e di segnare la sua risposta come migliore. Scorre anche molte altre domande, fortunatamente molte sono già risposte e diverse hanno già segnata la risposta migliore (spesso giusta).
Stefano, al posto di leggere decine di mail è soddisfatto anche perchè così ha più tempo per riposarsi. Il giorno successivo chiede agli studenti il loro parere e viene deciso di utilizzare il sito mentre le mail saranno utilizzate solamente per le comunicazioni urgenti.
Stefano è quindi riuscito a:

- Iscriversi al sito come professore
- Accedere al sito
- Mettere "love" a delle domande
- Mettere "up" a delle risposte
- Rispondere ad una domanda
- Segnare una risposta come migliore (ad una domanda non sua)

Scenario d'uso di Luca

Luca ha appena finito la lezione di Analisi 2. Essendo ormai le 19 i suoi compagni tornano subito tutti a casa. Lui avrebbe ancora molte domande da fare sia ai compagni che al professore ma decide di lasciar stare. Essendo un ottimo studente ed avendo un buon rapporto con il professore, decide di consigliargli l'uso del sito. A Luca il sito è stato consigliato da un suo amico che studia lettere.
Il giorno seguente il professore decide di consigliare agli studenti il sito. Luca, appena iniziata la pausa pranzo, decide di iscriversi al sito (non come professore) e di loggarsi.
Vede che nella sezione relativa a matematica non c'erano domande disponibili. Inizia quindi a inserire delle domande. Il pomeriggio a lezione riesce a parlare con qualche compagno e gli consiglia di usare il sito. I compagni decidono allora di inserire anche loro delle domande nel sito.
Arrivato a casa la sera, Luca vede che ci sono diverse risposte alle sue domande. Decide di segnare le risposte migliori alla sue domande.
Essendo lui uno studente molto bravo, riesce a rispondere a molte domande che erano state poste dai suoi compagni di corso.
Guardando un po' le risposte a diverse domande, mette in quelle che preferisce l' "up". Luca dedice inoltre di porre qualche altra domanda sul sito. Si accorge, mentre guarda le domande, che ha per sbaglio posto una domanda molto simile ad una già inserita. Decide allora di eliminare la sua domanda simile all'altra.
Ci sono diverse domande che gli piacciono e decide di mettere a queste il "love".
Luca è soddisfatto perchè riesce ad avere alcune risposte a delle domande da lui poste. Inoltre, riesce a condividere il suo sapere matematico con altri utenti del sito (anche professori).
Luca è quindi riuscito a:

- Iscriversi al sito (non come professore)
- Accedere al sito
- Mettere "love" a delle domande
- Mettere "up" a delle risposte
- Rispondere a delle domande
- Inserire delle domande
- Eliminare una sua domanda
- Segnare una risposta come migliore (a domande sue)

3.2 Sketch

Ho creato alcuni sketch, mi sono stati utili alla realizzazione del progetto. Questi sketch sono disordinati e non corrispondono esattamente al sito come è ora. A scopo di documentazione ho deciso comunque di inserirne alcuni.

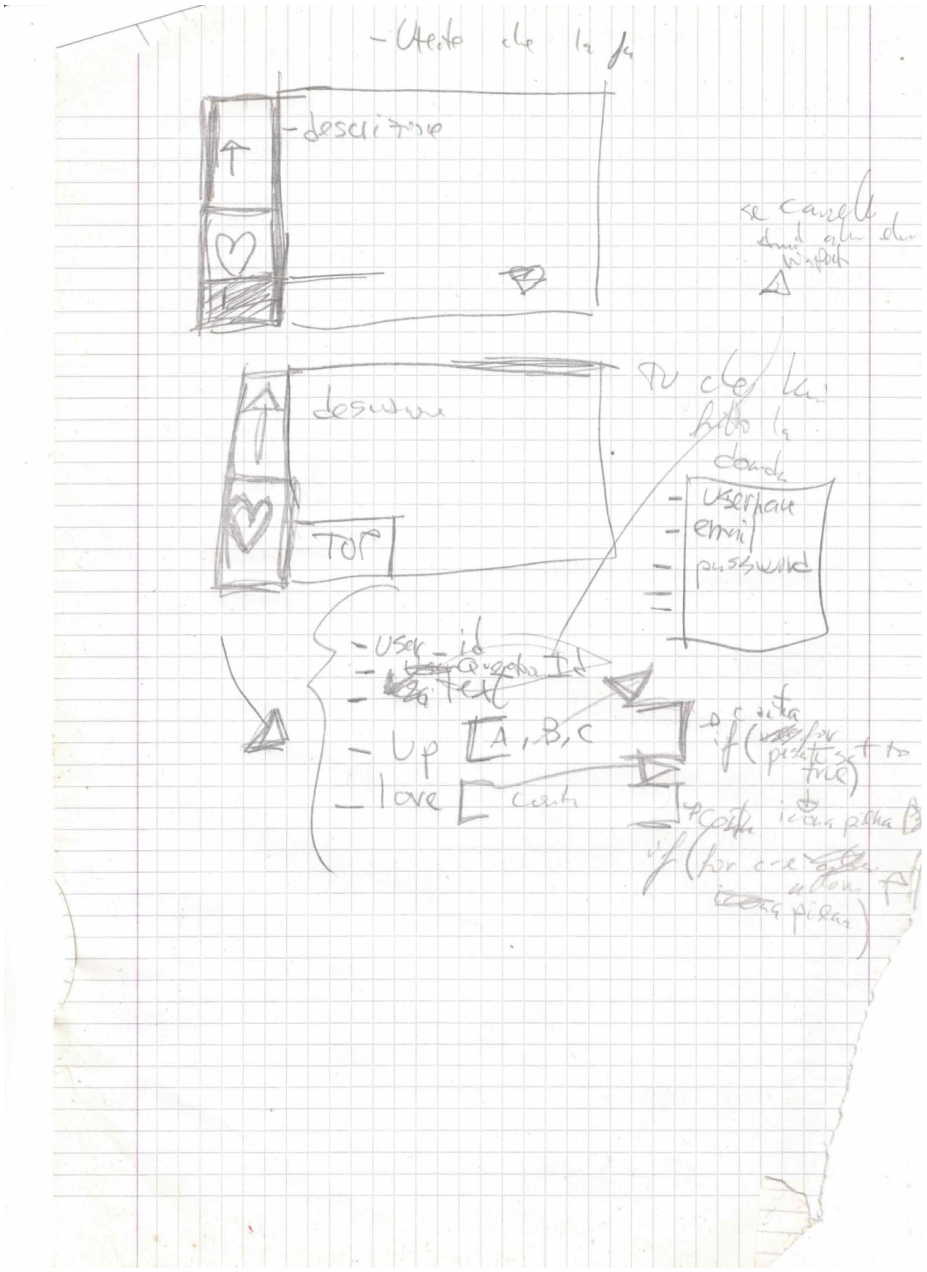

Figura 3.1: Sketch 1

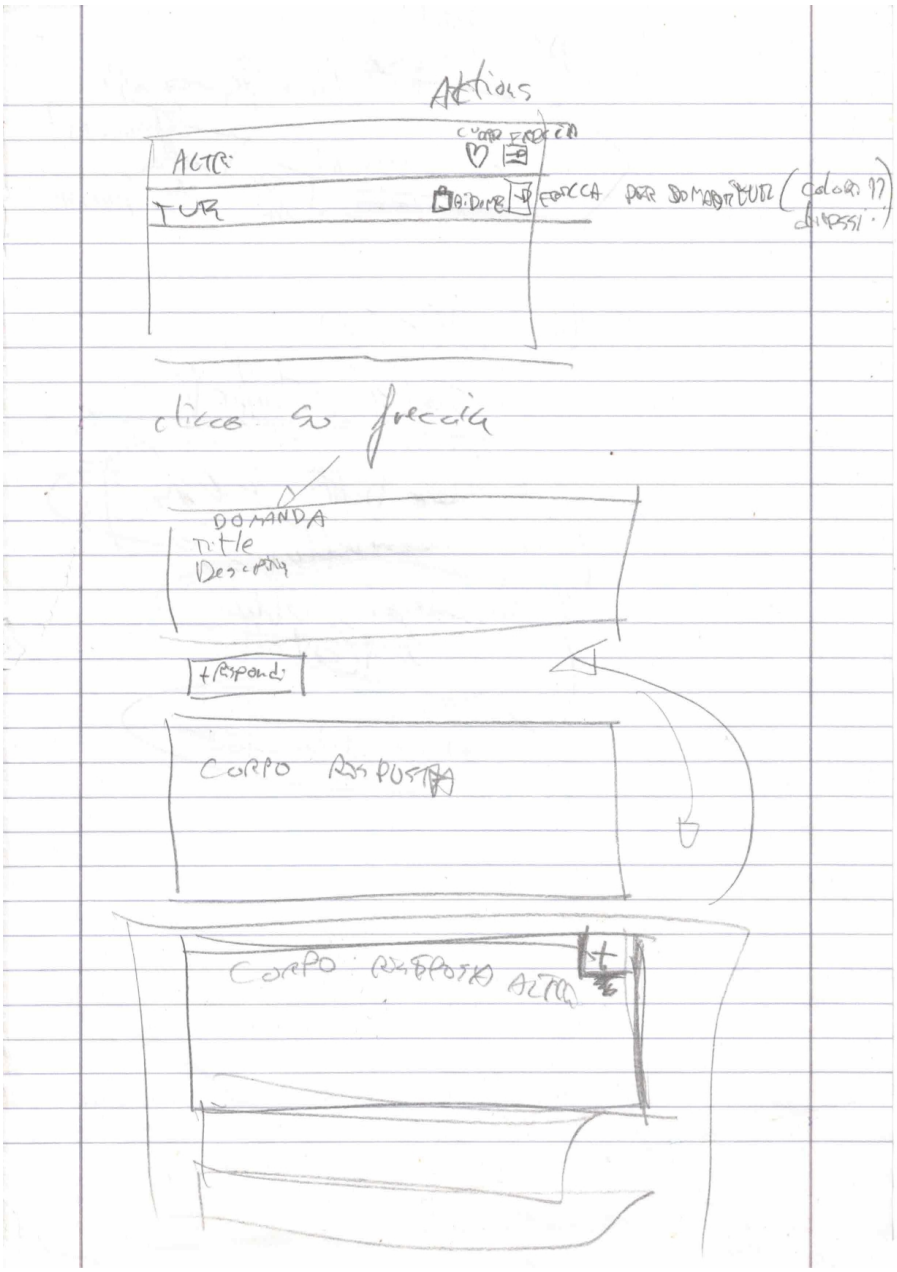

Figura 3.2: Sketch 2

3.3 Interfacce

Il sito si compone principalmente delle schermate:

- *Login (che funge anche da Logout)*
- *SignUp*
- *Home*
- *Domanda scelta*
- *Area scelta* (ad esempio ART, MATH, ...)

Login

Il Login è la schermata iniziale del sito. Un utente prima di poter andare nella Home o nelle aree deve prima loggarsi. Mentre ci si logga si può decidere di mostrare la password o meno con l'apposito pulsante.
Nel caso l'utente non si fosse ancora registrato deve prima registrarsi.

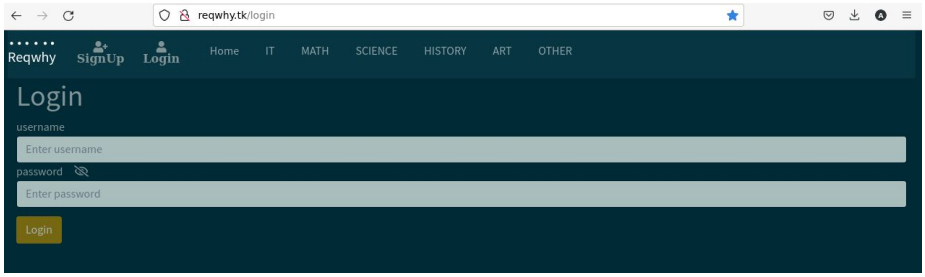

Figura 3.3: Login

Nel caso l'utente provasse ad andare nella schermata di Login quando è già loggato, verrebbe mostrato un bottone di Logout. Nello stesso caso, se l'utente provasse ad andare nella schermata di SignUp, verrebbe mandato nella schermata di Login.

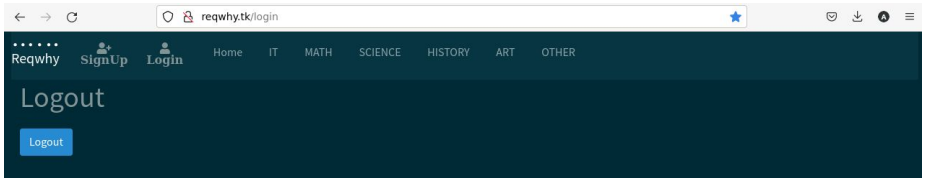

Figura 3.4: Logout

SignUp

Il SignUp è la schermata dove l'utente si registra. Mentre ci si registra si può decidere di mostrare la password o meno con l'apposito pulsante.
Un utente può decidere di registrarsi come insegnante.

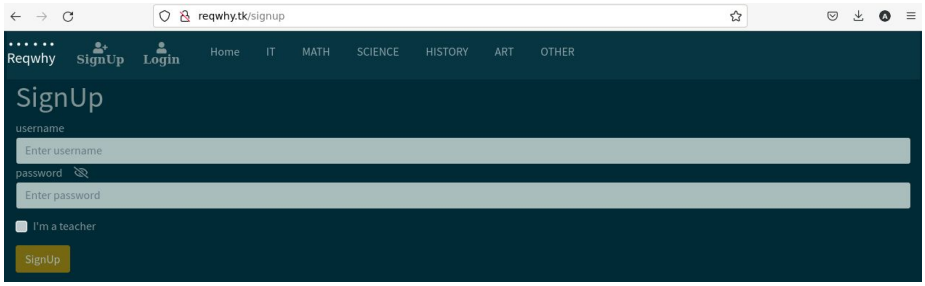

Figura 3.5: SignUp

Home

La Home è la schermata principale del sistema. Qui si possono vedere tutte le domande inserite.
Tramite il bottone "Add Question" è possibile aggiungere una domanda al sistema.
Con il bottone "Submit" si invia la domanda al sistema per essere aggiunta.
Con il bottone "Cancel" si annulla l'azione.
Tramite l'icona a forma di cuore si puo' mettere o togliere il "love" alle domande che più piacciono all'utente.
Con l'icona a forma di freccia verso destra l'utente apre la pagina relativa alla domanda scelta.
E' possibile inoltre, tramite l'apposita icona a forma di cestino, eliminare la domanda scelta.

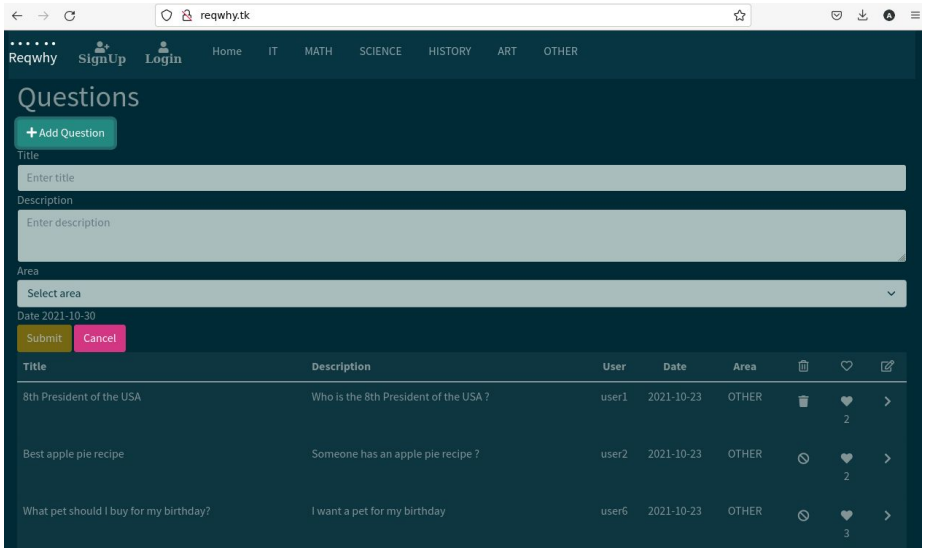

Figura 3.6: Home

Domanda scelta

In questa schermata è possibile vedere le risposte date alla domanda scelta. Nella parte alta vengono mostrate delle informazioni relative alla domanda. Subito sotto sono visualizzare le risposte con le relative azioni.
Un'azione è quella di poter mettere o togliere l'"up" alle risposte che piacciono di più all'utente, usando l'apposita icona.
Nel caso la domanda sia stata posta dall'utente che l'ha aperta, si ha la possibilità di segnare la risposta migliore tramite l'apposito bottone "set as BEST". Un utente professore può segnare la risposta migliore anche a domande che non ha posto.
Con il bottone "Add Answer" si ha la possibilità di inserire una risposta alla domanda.
Con il bottone "Submit" si invia la risposta al sistema, con il bottone "Cancel" si annulla l'azione.

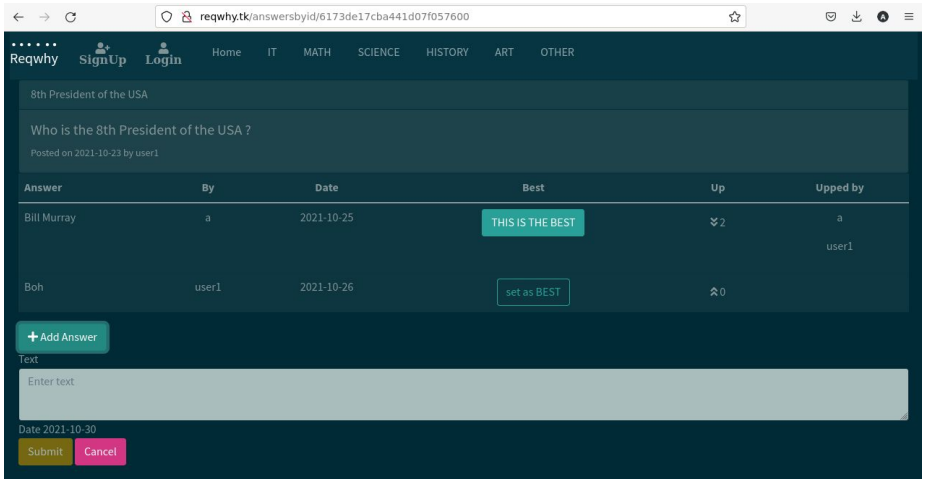

Figura 3.7: Domanda e risposte di un utente

Area scelta

La schermata di un'area è la schermata in cui vengono visualizzate le domande relative a quell'area. E' equivalente alla schermata "Home" a parte per il bottone "Add Question" che permette di aggiungere domande.

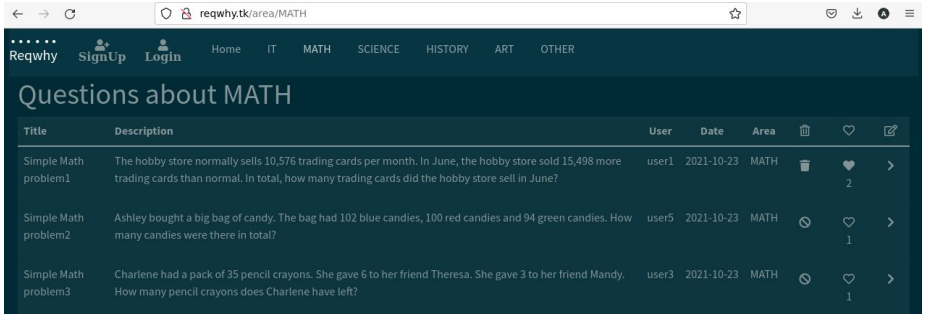

Figura 3.8: Domande relative ad area MATH

3.4 Architettura

I componenti (cioè component di Vue) principalmente utilizzati sono:

- *Navbar*
- *Login*
- *SignUp*

- *Questions*
- *AnswersByQuestionId*

Navbar

"Navbar" consiste nella barra di navigazione, presente in tutte le schermate del sistema.

Login

"Login" è il component utilizzato per visualizzare la schermata di Login o altrimenti di Logout.

SignUp

"SignUp" è il component utilizzato per visualizzare la schermata di SignUp.

Questions

"Questions" è il component utilizzato sia per visualizzare la Home (che contiene tutte le domande) che le schermate relative alle diverse aree (ad esempio MATH).

AnswersByQuestionId

"AnswersByQuestionId" è il component utilizzato per visualizzare una domanda e le relative risposte.

3.5 Database

Di seguito la rappresentazione grafica del DB.

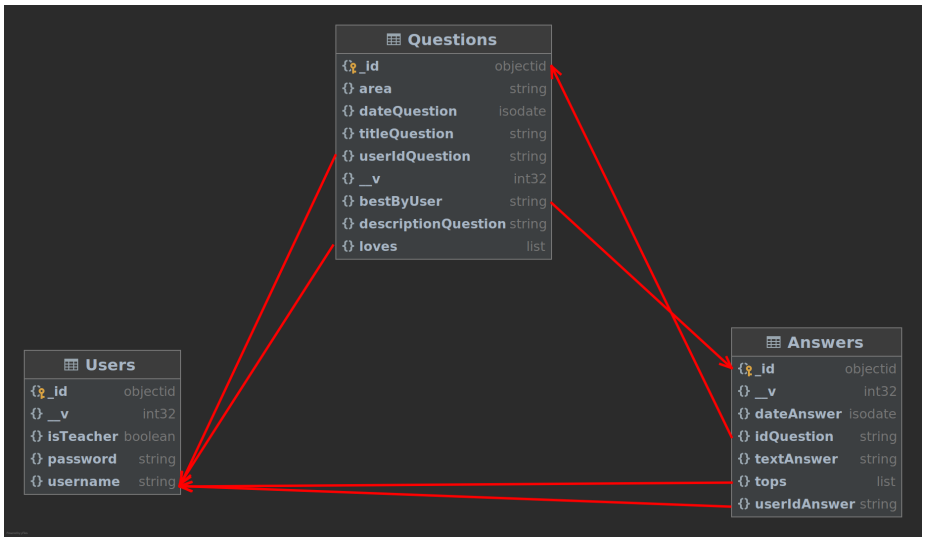

Figura 3.9: Rappresentazione grafica del DB

Il database è stato modellato con 3 model. Uno relativo alle domande (Questions), uno relativo alle risposte (Answers) e uno relativo agli utenti(Users).

Questions

- *titleQuestion*: titolo della domanda;
- *descriptionQuestion*: descrizione della domanda;
- *dateQuestion*: data (completa) della domanda;
- *area*: area della domanda;
- *userIdQuestion*: username dell'utente che ha inserito la domanda;
- *bestbyUser*: _id della risposta migliore alla domanda;
- *loves*: lista degli username degli utenti che hanno messo "love" alla domanda.

Answers

- *idQuestion*: _id della domanda collegata alla risposta;
- *userIdAnswer*: username dell'utente che ha inserito la risposta;
- *textAnswer*: testo della risposta;
- *dateAnswer*: data (completa) della risposta;
- *tops*: lista degli username degli utenti che hanno messo "up" alla risposta.

Users

- *username*: username dell'utente;
- *password*: password dell'utente;
- *isTeacher*: valore per indicare se l'utente è un insegnante o no.

Capitolo 4

Tecnologie

Lo stack utilizzato per questo progetto è MEVN, per cui sono stati utilizzati:

- *MongoDB*: MongoDB è un database NoSQL, che viene spesso utilizzato nei siti moderni al posto di database relazionali come MySQL;
- *Express*: è un framework per applicazioni Node;
- *Vue*: è un framework Javascript frontend, al suo posto è possibile utilizzare React (MERN) o Angular (MEAN);
- *Node*: è un framework Javascript backend.

La scelta di usare MEVN è stata data principalmente dal fatto che, considerando anche React e Angular, Vue è quello con la curva di apprendimento più bassa[1]. Sembra inoltre tra i tre quello più semplice da utilizzare.
Anche essendo tra i tre il framwork più moderno, sta acquisendo molta popolarità[2].

[1] `https://www.codeinwp.com/blog/angular-vs-vue-vs-react/`
[2] `https://www.codeinwp.com/blog/angular-vs-vue-vs-react/`

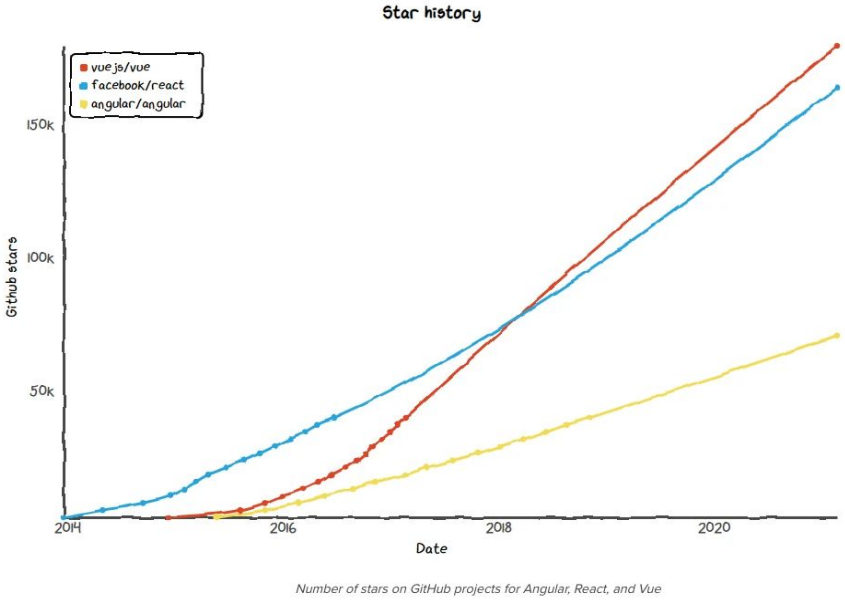

Figura 4.1: Confronto Vue, React e Angular

Bisogna ricordare comunque che Vue è guidato da una community open source, mentre invece Angular e React hanno dei dipendenti di Google e di Facebook che contribuiscono attivamente allo sviluppo.
Nella scelta d'uso di uno di questi tre framework, ritengo la stesura del codice più ordinata rispetto ad uno stack XAMP.
Oltre allo stack MEVN ho utilizzato anche altre tecnologie:

- *Bootstrap*: Ho usato un ottimo tema[3] basato su "Bootstrap", che ho poi scaricato e caricato nel mio progetto;
- *local-storage*: Ho usato il pacchetto npm "local-storage"[4] per settare e poi recuperare l'username di un utente. In questo modo sono riuscito a salvare l'username dell'utente che si è loggato in modo semplice e veloce;
- *bcryptjs*: Ho usato il pacchetto npm "bcryptjs"[5] per criptare e descriptare le password salvate nel database. In questo modo le password salvate nel database non sono visibili in chiaro, nemmeno all'amministratore del sistema.

[3] `https://bootswatch.com/solar/`
[4] `https://www.npmjs.com/package/local-storage`
[5] `https://www.npmjs.com/package/bcryptjs`

Capitolo 5

Codice

Ho deciso di inserire di seguito alcune parti di codice che ritengo rilevanti.

5.1 ReqwhyRoutes

Ho inserito qui le parti di *reqwhyRoutes.js* che contengono le diverse API create.

```
app.route('/api/questions')
        .get(questionsController.list_questions)
         .post(questionsController.create_question);

app.route('/api/questions/:id')
  .get(questionsController.read_question)
  .put(questionsController.update_question)
  .delete(questionsController.delete_question);

app.route('/api/questionsbyarea/:area')
        .get(questionsController.list_questions_by_area);

app.route('/api/answersbyquestionid/:question')
        .get(answersController.list_answers_by_question_id)
        .delete(answersController.delete_answers_by_question_id);

app.route('/api/answers')
        .post(answersController.create_answer);

app.route('/api/answers/:id')
        .put(answersController.update_answer);

app.route('/api/typeofuser/:username')
        .get(usersController.read_type_of_user);
```

```
        app.route('/api/signup')
                .post(usersController.create_user);

        app.route('/api/login')
                .post(usersController.verify_user);
```

5.2 UsersController

Ho inserito qui due funzioni di *usersController.js*, da notare l'uso delle funzioni di criptaggio e decriptaggio delle password.

Funzione per creare un utente

```
exports.create_user = function(req, res) {
    var new_user = new User(req.body);
    new_user.password = bcrypt.hashSync(req.body.password, 10);
    new_user.save(function(err, user) {
            if (err)
                res.send(err);
            else{
                    res.status(201).send(true);
                }
    });
};
```

Funzione per verificare un utente

```
exports.verify_user = function(req, res) {
    User.findOne({username:  req.body.username}, req.body, function(err, user) {
        if (err)
            res.send(err);
        else{
            if(user==null){
                res.send(false);
            }
            else{
                if(bcrypt.compareSync(req.body.password, user.password)) {
                    res.send(true);
                }
                else{
                    res.send(false);
                }
            }
        }
    });
};
```

5.3 Router

Ho inserito qui il contenuto di *router.js* in cui si può vedere la gestione delle rotte del sistema, usando il componente *VueRouter* chiamato "router".

```
const router = new VueRouter({
    mode: 'history',
    routes: [
      { path: '/', name: "Home", component: Questions},
      { path: '/area/:area', name: "QuestionsByArea", component: Questions},
      { path: '/answersbyid/:question', name: "AnswersByQuestionId", component: AnswersByQue
      { path: '/login', name: "Login", component: Login},
      { path: '/signup', name: "SignUp", component: SignUp},
      { path: '/404', component: NotFound },
      { path: '*', redirect: '/404' },
    ]
})
```

Capitolo 6

Deployment

Per quanto riguarda il deploy del progetto ho deciso di realizzarlo sia locale che reale.
Il sito in locale è utilizzabile digitando i seguendi comandi.
Per clonare la cartella del progetto:

```
git clone https://github.com/AlbertoDonati/Reqwhy.git
```

Per installare le dipendenze:

```
npm install
```

Per far partire il server:

```
node index.js
```

Per quanto riguarda invece il deploy reale ho descritto indicativamente nei passaggi successivi i passi compiuti.
Ho creato una macchina virtuale Azure, in particolare una macchina Ubuntu 18.04-LTS.

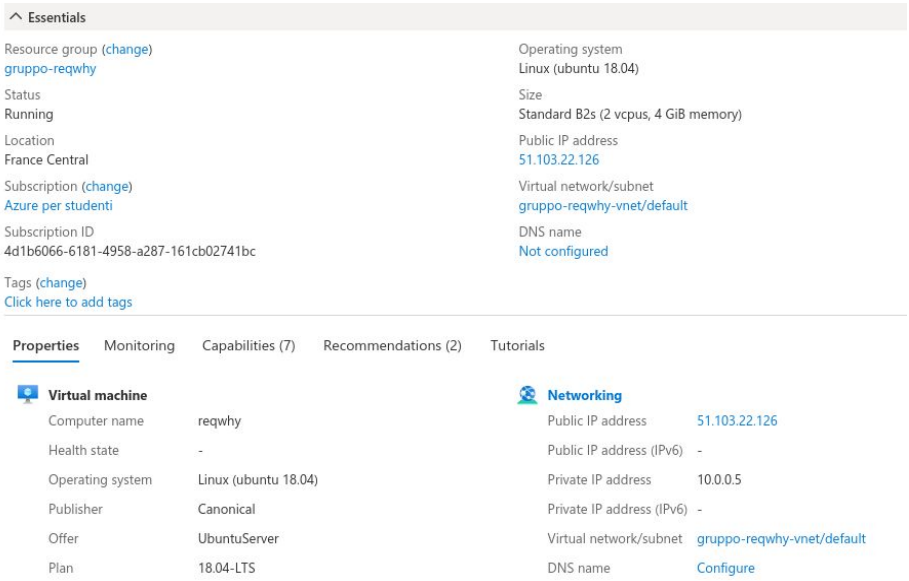

Figura 6.1: Immagine presa da Azure

In questa macchina ho lasciato aperte, sia per il corretto utilizzo del sito che per comodità, diverse porte.
Inoltre, seguendo le istruzioni date dalla documentazione di Azure[1] sono riuscito a connettermi alla macchina tramite XRDP. Questo mi ha permesso di connettermi comodamente alla macchina utilizzando anche l'interfaccia grafica. All'interno della macchina è possibile far partire il progetto usando i comandi locali (vedi sopra).
Bisogna prestare attenzione al fatto che il progetto è stato creato per connettersi alla porta 3000 e non 80.
Per fare il forwarding della porta mi è stato utile questo comando[2]:

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 3000
```

L'indirizzo IP statico della vm è stato collegato a un dominio da me creato. Ho utilizzato per la creazione del dominio il servizio Freenom[3]. Ho collegato il dominio alla macchina creata grazie alla gestione DNS.

[1] https://docs.microsoft.com/en-us/azure/virtual-machines/linux/use-remote-desktop

[2] https://snikt.net/blog/2012/11/18/linux-how-to-forward-port-3000-to-port-80/

[3] https://my.freenom.com/

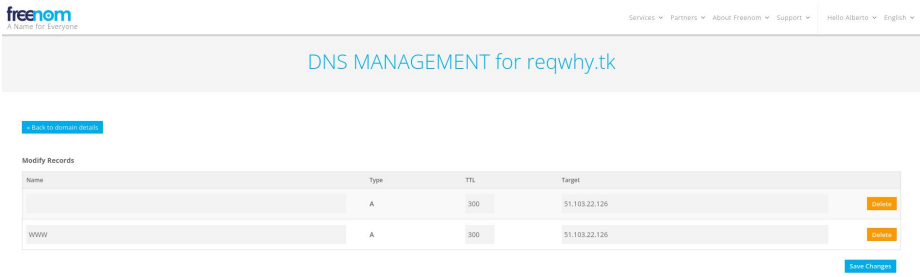

Figura 6.2: Immagine presa da Freenom

Il progetto è stato così reso raggiungibile all'indirizzo: $http://reqwhy.tk$.

Capitolo 7

Test

Ho testato personalmente il progetto sia in locale sia nella versione hostata su Azure.
L'ho inoltre fatto provare a diverse altre persone. I test mi hanno permesso di accorgermi del fatto che alcuni elementi non erano intuitivi, e quindi ho cercato di migliorarli. Ho inoltre cambiato il tema e fatto altre piccole modifiche al codice.
Facendo testare il progetto mi sono accorto, oltre alla non intuitività di certi elementi, della necessità di fare le giuste domande riguardo il progetto.
Parte delle risposte che mi sono state date riguardo i test che ho fatto svolgere, sono state date in maniera errata. Questo anche perchè l'utente non aveva ben compreso alcune domande e/o non le aveva lette attentamente.

Capitolo 8

Conclusioni

Grazie al progetto ho potuto mettere in pratica l'utilizzo dell'architettura MEVN. Avendo svolto il progetto non con lo stack XAMP ma MEVN, ho così utilizzato un approccio diverso e a mio parere decisamente più ordinato nell'organizzazione del codice.
Oltre a ciò ho conosciuto, trovato ed utilizzato i pacchetti npm, che trovo tutt'ora molto utili per diversi ambiti.
Inoltre, in questo progetto ho creato ed utilizzato delle API da me definite.
Infine, il progetto mi ha permesso di scoprire la comodità d'uso della developer mode del browser, utile specialmente per testare il funzionamento del sistema.

Elenco delle figure