Bonoldi Enrico, Classe 5AI

ELABORATO INDIVIDUALE DI INFORMATICA E SISTEMI E RETI

Descrizione

> Un'agenzia matrimoniale vuole gestire e pubblicare con un'applicazione web le informazioni per la ricerca della propria anima gemella in un dato territorio.

INFORMATICA

> *tutto il progretto è dispobnibile qui*

1. Analisi

Soggetto : Agenzia Matrimoniale
Clienti : Chiunque (vedi supposizioni...)

L'applicativo deve fornire l'accesso alle azioni di ricerca in modo semplice e veloce.

> *Entrando in un ambito così personale è necessario tener conto della sicurezza relativa ai dati sensibili.*

Soluzione proposta

La scelta dell'abiente è ricaduta su nodeJS e expressJS data la loro immediatezza e velocità di sviluppo. Offrono stumenti (validator,session manager,...) che permettono di gestire in sicurezza le parti piu critiche della applicazione quali inserimento di dati e autenticazione.

Tecnologie utilizzate

lato server

- NodeJS
- ExpressJS
    - express-session
    - EJS (templating)
    - pg

lato client

- HTML
- CSS
    - Bootstrap
- Javascript
    - Leaflet (mappe)
    - JQuery

Struttura

La struttura del progetto è la seguente:

- **Accesso**
    - **Registrazione**
    - **Login/Logout**
- **Ricerca**
- Gestione profilo

in grassetto i servizi implementati nella soluzione finale.

Supposizioni

1. gli iscritti al sistema siano maggiorenni.

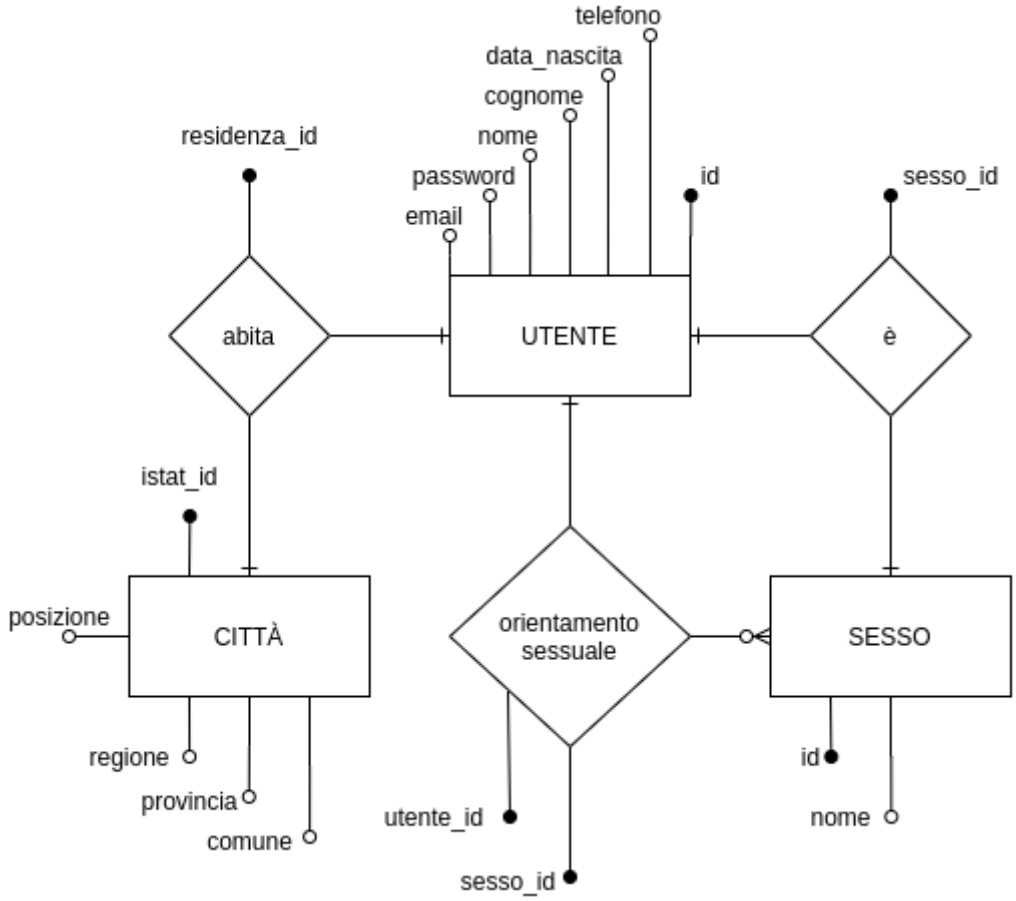
2. l'orientamento sessuale di ogni utente può essere maschio e/o femmina.
3. ricerca
    1. basata sulla distanza tra i comuni di residenza degli utenti dal quello dell'utente che ricerca.
    2. basata sul orientamento sessuale (maschio e/o femmina)

Possibili sviluppi futuri

1. Ampliamento profilo e ricerca (carattere, aspetto,...)
2. Messaggistica istantanea.

2. Modello concettuale e logico

Modello ER



Modello logico

CITTA(<u>istat_id</u>,comune,provincia,regione,posizione)
CAP(*<u>istat_id</u>*,<u>CAP</u>)

SESSO(<u>id</u>,nome)
UTENTE(<u>id</u>,*sesso_id*,*residenza_id*,email,password,nome,cognome,data_nascita,telefono)

ORIENTAMENTO(*<u>utente_id</u>*, *<u>sesso_id</u>*)

note di realizzazione

- La tabella CAP è attualmente inutilizzata ma è stata inserita per completezza.
- E' stata inserita una vista per facilitare l'accesso ai dati utente.

3. Implementazione database

DDL disponibile qui
app/db/agenzia_matrimoniale.ddl.sql

DUMP disponibile qui
app/db/agenzia_matrimoniale.dump.sql

4. Query significative

Ricerca dei partner

app/libs/user.js

> $1 - id utente
> $2 - distanza dall'utente corrente (in metri)

```
SELECT
   *,
   ST_AsText(posizione) as posizione_coordinate,
   ST_Distance(
      info_utente.posizione,
      (
         SELECT
            posizione
         from
            info_utente
         where
            info_utente.id = $1
      )
   ) as distance_between
from
   info_utente
where
   (
      ST_Distance(
         info_utente.posizione,
         (
            SELECT
               posizione
            from
               info_utente
            where
               info_utente.id = $1
         )
      ) < $2
      AND position(
         (
            SELECT
               sesso
            from
```

```
                info_utente
            where
                info_utente.id = $1
        ) in info_utente.orientamento_aggregato
    ) > 0
    AND position(
        info_utente.sesso in (
            SELECT
                orientamento_aggregato
            from
                info_utente
            where
                info_utente.id = $1
        )
    ) > 0
)
OR info_utente.id = $1
```

Orientamento sessuale

app/db/agenzia_matrimoniale.ddl.sql

utilizzata nella costruzione della view `info_utente`

```
SELECT
    orientamento.utente_id,
    array_to_string(array_agg(sesso.nome), ',' :: text)
    AS orientamento_aggregato
FROM
    orientamento,
    sesso sesso
WHERE
    orientamento.sesso_id = sesso.id
GROUP BY
    orientamento.utente_id
```

5. Applicazione Web (implementazione completa)

path : app

folders:

- **libs** : utilities per accesso al db (registrazione,login,...)
- **db** : file per il setup e popolamento del db
- **middlewares**
- **public** : directory per servire file statici (js,css,assets,...)
    - **views** : *templates* e sections per *EJS*

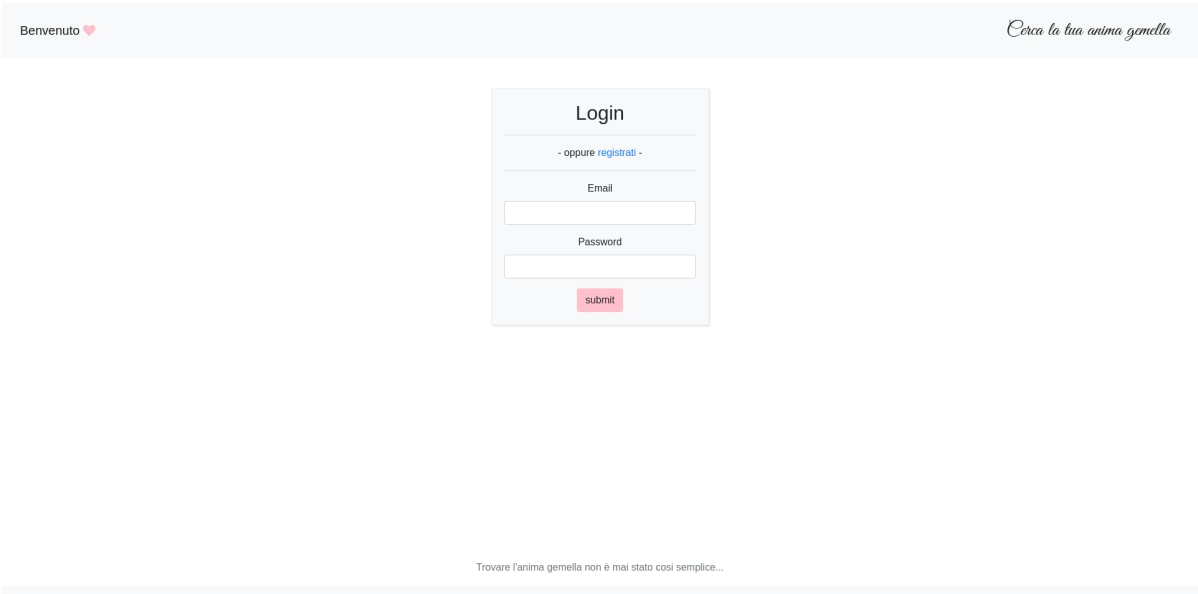
entry file **index.js**:

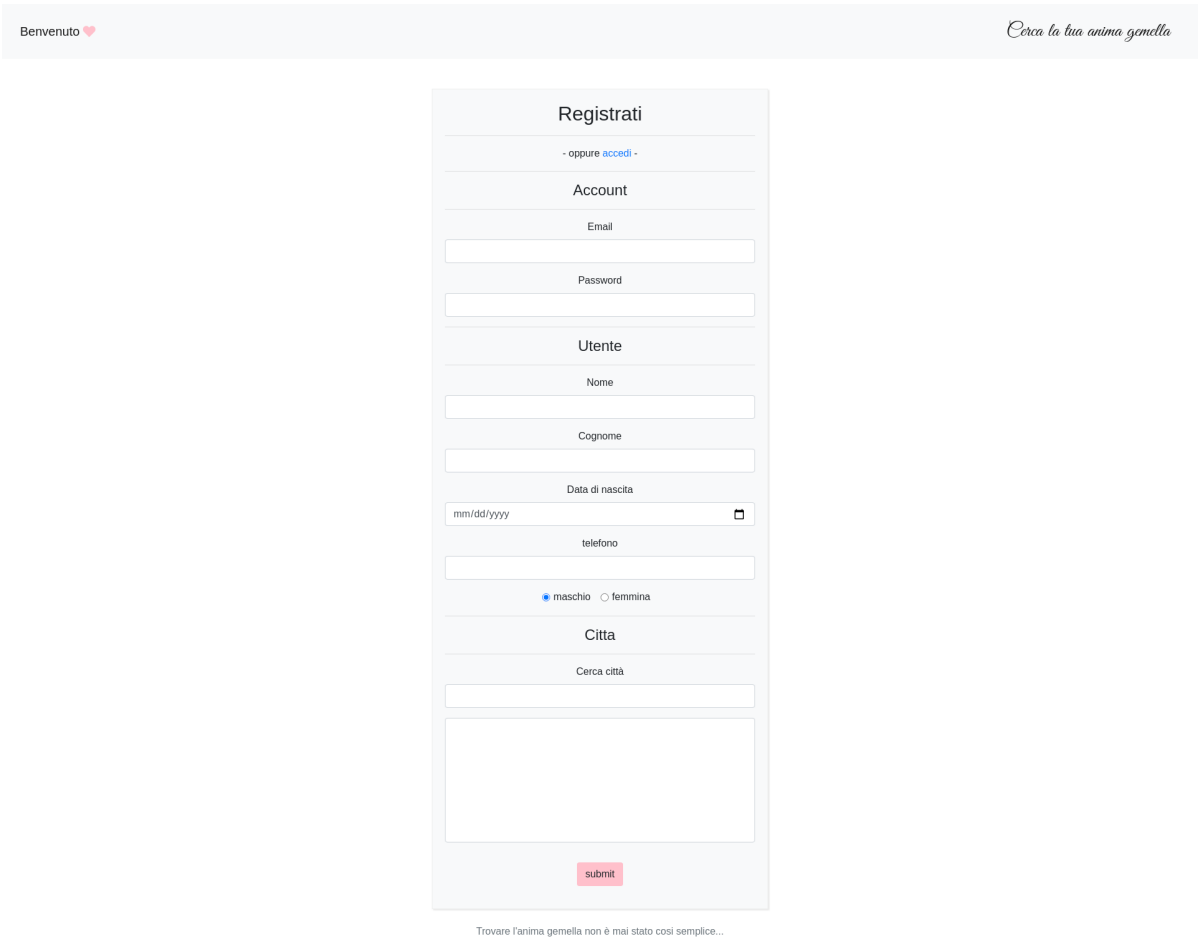
- setup di expressJS e registrazione routes.

env file **env.json**

Login/Registrazione

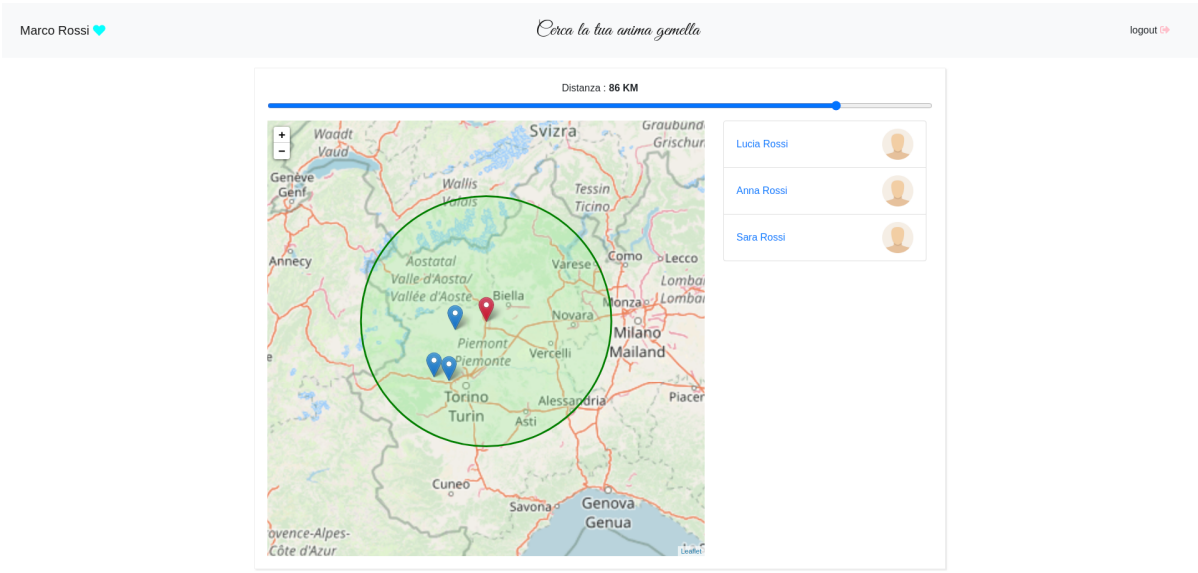
- Login e inizializzazione di una nuova sessione
- Controllo accesso via middleware (app/middlewares/auth.js)
- Registrazione





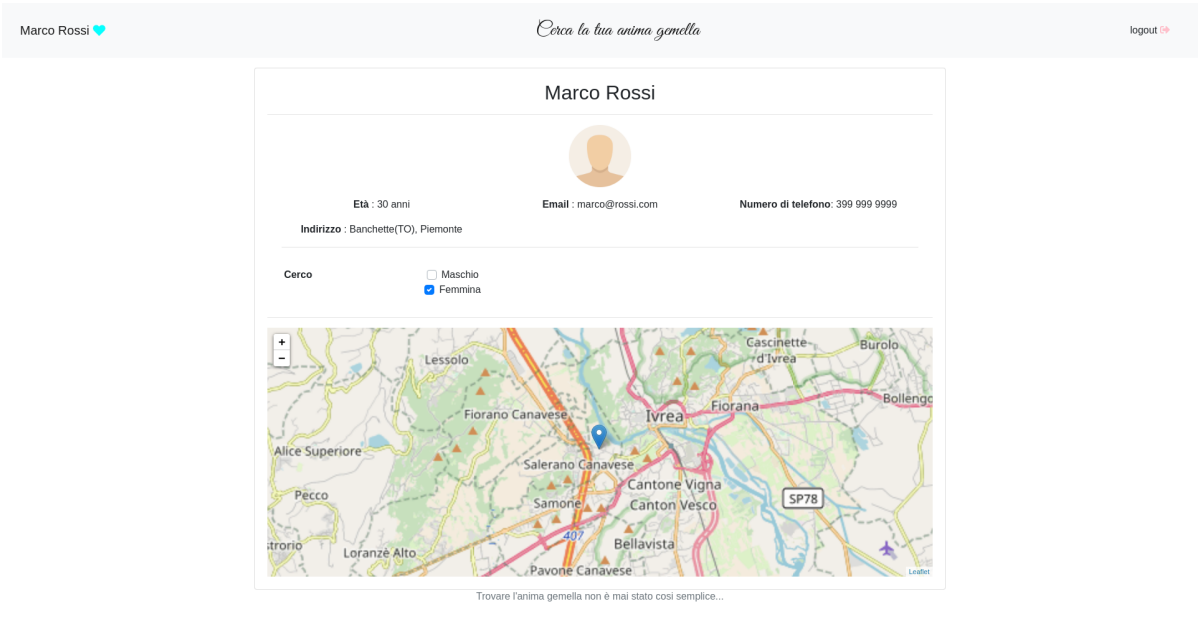
Ricerca

- Ricerca dinamica dei possibili partner in un certo range (in km)
- l'endpoint per la ricerca è `/api/find/{range in metri}`
    - ritornerà una lista di utenti compatibili



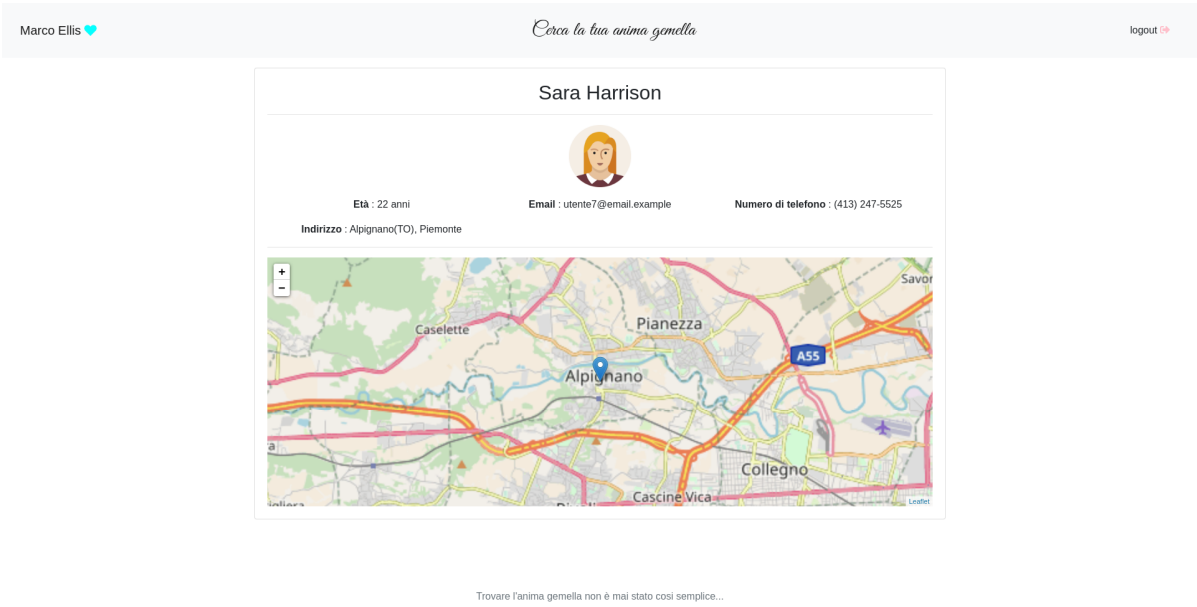
Profilo account

- Visionare i dati personali del proprio account
- Modificare l'immagine profilo (cliccando sull'immagine corrente)
    - l'endpoint per la modifica dell'immagine profilo è `/users/img`



Profilo utente

- Visionare i dati dell'account di un utente iscritto alla piattaforma



Proxy immagini profilo

Per ritornare una immagine profilo di default è stata implementata una **route "proxy"**

6.a NoSql

Una alternativa efficace a **postgres~postgis** è **mongodb~geoJSON**.

GeoJSON (RFC 7946) è una specifica che introduce nel tipo di file **JSON** i riferimenti geospaziali (Point,LineString,...) e il sistema di coordinate.

6.b Sviluppo

Per il mantenimento del codice è stato usato **GIT**

SISTEMI E RETI

Descrizione

> Esporre una possibile soluzione tecnologica, giustificandola, che consenta all'agenzia di offrire ai clienti la possibilità di accedere al servizio offerto in totale sicurezza e riservatezza. Illustrare in particolare gli aspetti di sicurezza delle comunicazioni a base web.

Navigazione web

Quando navigiamo nella rete percorriamo "strade" non protette e gestite da terzi; questo solleva un problema di sicurezza visto che la nostra applicazione dovrà distribuire dati personali.

Soluzione HTTP Secure

L'applicazione web usa **HTTP** e questo non garantisce nessuna sicurezza per quanto riguarda la riservatezza; non è presente **nessun meccanismo di crittografia** e i dati viaggiano in chiaro tra client e server.

Per mettere in sicurezza HTTP si utilizza SSL(secure socket layer, **TLS** nella forma aggiornata) che si avvale di un meccanismo di certificati e crittografia asimmetrica per garantire la sicurezza.

Funzionamento

La crittografia a **chiave asimmetrica** ci garantisce la riservatezza della comunicazione ma dal momento che questa modalità ha bisogno di distribuire la chiave pubblica sorge un problema di autenticità che è risolto mediante l'utilizzo di **Enti certificatori** che garantiscono l'autenticità della chiave pubblica utilizzata per la comunicazione.

> Questo tipo di autenticazione può essere one-way nel caso sia solo il server ad autenticarsi, two-way (mutual) nel caso sia anche il client a doversi autenticare; nel secondo caso il procedimento rimane lo stesso sia per il client che per il server.

1. **Client Hello** In questa fase il client richiede al serve la connessione protetta tramite TLS e invia una serie di informazioni (epoch time,session ID, chipers, server name)
2. **Server Hello** Contiene la risposta al client con informazioni relative a:
    1. Certificato
    2. Algoritmi ci cifratura
    3. Conferma di autenticazione (one-way o two-way)
3. **Controllo certificato** Il client (il browser) verifica l'autenticità del certificato affidandosi agli enti certificatori e controlla l'impronta e l'hostname. Inoltre registra gli algoritmi accettati dal server.
4. Da qui in poi la comunicazione passa a livello applicazione in cui tutto il traffico HTTP viene criptato/decriptato da TLS.

Installazione

1. Generiamo una chiave asimmetrica (RSA)
    - `openssl genrsa -des3 -passout pass:admin -out agenzia_matrimoniale.key 2048`
2. Generiamo la **CSR** (Certificate Signing Request), in questo modo chiediamo all'ente di approvare la nostra richiesta per la generazione di un certificato. In questo vanno inseriti una serie di informazioni circa il servizio che vogliamo certificare.
    - `openssl req -new -key agenzia_matrimoniale.key -out agenzia_matrimoniale.csr`
3. La nostra richiesta di certificato viene inviata all'ente e in caso di approvazione ottenia il certificato vero e proprio (`agenzia_matrimoniale.crt`).
   Nel nostro caso non avendo a disposizione un ente certificatore la csr è stata auto verificata con `openssl` con una durata di 365 giorni.
   NOTA : non essendo noi un ente certificatore non possiamo dare nessuna garanzia sul certificato, che rimane comunque utilizzabile ma non "sicuro"
    - `openssl x509 -req -days 365 -in agenzia_matrimoniale.csr -signkey agenzia_matrimoniale.key -out`

```
agenzia_matrimoniale.crt
```

Questo è il risultato finale