Progetto - Sistemi Aperti e Distribuiti

Tasso Tennis - Prenotazioni Campi da tennis

STUDENTI:

Riommi Maria [315912]
Vescera Nicolò [301838]

Obiettivi

Ideare e realizzare la progettazione di un Web Service che permetta di gestire in modo interattivo la prenotazione, da parte di un utente Player, e l'inserimento, da parte di un utente Latifondista, di campi da tennis usando le tecnologie SOAP/REST su una piattaforma a scelta e un database per l'archiviazione dei dati.

TECNOLOGIE UTILIZZATE

Tecnologie utilizzate

- Python + Flask
- REST/Json
- Bootstrap
- JQuery + AJAX
- Database SQLite

Perchè Python ?

- Interpretato
- Largamente Supportato
- Altissimo Livello
- Multi-Paradigma
- Adatto allo sviluppo di Applicazioni Distribuite
- Facile e Divertente

Python >>>>> tutto il resto

Perchè REST/JSON ?

REST vs SOAP

- Flessibilità
- API ottimizzate
- Semplicità
- Ampiamente utilizzato

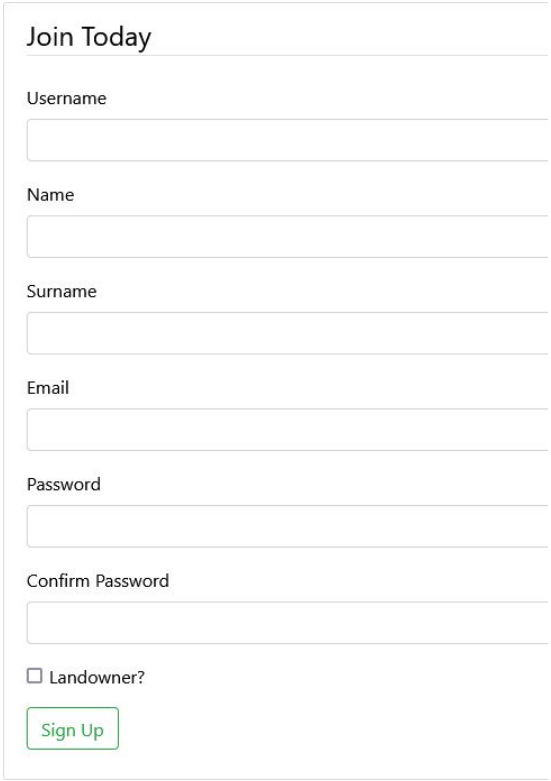
JSON vs XML

- Semplice da scrivere/leggere
- Supporta tipi di dato
- Supporto degli Array
- Libreria Python semplice e intuitiva

REALIZZAZIONE 

Funzionalità Comuni

- Homepage iniziale
- About
- Registrazione
- Login
- Visualizzazione del profilo



Utenti

- **Latifondista**
  - Inserire, modificare ed eliminare i campi
  - Visualizzare le prenotazioni effettuate dai Player per i suoi campi

- **Player**
  - Effettuare/eliminare una prenotazione
  - Controllare le prenotazioni effettuate

API 

Convenzioni utilizzate

- GET:
  - Successo `{'rep': [dati_richiesti]}`
  - Errore `{'error': 'Messaggio di errore'}`

- POST, PUT, DELETE:
  - Successo `{'message': 'Messaggio di successo'}`
  - Errore `{'error': 'Messaggio di errore'}`

/api/fields

- **GET :** Ritorna i campi dell'utente
- **POST :** Inserisce un nuovo campo nel database
- **DELETE :** Elimina un campo dal database
- **PUT :** Aggiorna i dati di un campo

/api/prenotations

- **GET :** Ritorna tutte le prenotazioni dell' utente
- **POST :** Inserisce una nuova prenotazione nel database
- **DELETE :** Elimina una prenotazione dal database

/api/prenotations - DELETE

```python
@app.route('/api/prenotations/<int:id>', methods=['DELETE'])
@login_required
def delete_prenotation(id):
    # controlla che l'utente sia un utente normale
    if current_user.is_authenticated and current_user.landowner == False:
        user_id = current_user.id

        # elimina la prenotazione
        to_delete = Prenotation.query.filter_by(id=id, player_id=user_id).first()
        if to_delete:
            db.session.delete(to_delete)
            db.session.commit()

            return json.dumps({'message': f'Prenotazione eliminata con successo !'}), 200

        return json.dumps({'error': 'Prenotazione inesistente !'}), 400

    return json.dumps({'error': 'Login first !'}), 401
```

Chiamata AJAX per /api/prenotations - DELETE

```
$.ajax({
        url: '/api/prenotations/'+id,
        type: 'DELETE',

        success: function(response) {
            risposta = JSON.parse(response);    // converte la risposta in JSON
            alert(risposta.message);            // avvisa dell'avvenuto successo
            location.reload();                  // aggiorna la pagina
        },

        error: function(error) {
            errore = JSON.parse(error.responseText)     // converte la risposta in JSON
            alert(errore.error);                        // avvisa dell'errore
        }
    });
```

DATABASE 💾

Tabella User

```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(30), nullable=False, unique=True)
    name = db.Column(db.String(30), nullable=False)
    surname = db.Column(db.String(40), nullable=False)
    email = db.Column(db.String(40), unique=True, nullable=False)
    password = db.Column(db.String(20), nullable=False)
    landowner = db.Column(db.Boolean(), default=False)
```

Tabella Field

```
class Field(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    name = db.Column(db.String(20), nullable=False, unique=True)
    description = db.Column(db.Text)
    address = db.Column(db.String(60), nullable=False)
    available_from = db.Column(db.Time, nullable=False)
    available_to = db.Column(db.Time, nullable=False)
    price_h = db.Column(db.Float, nullable=False)
    landowner_id = db.Column(db.String(30), db.ForeignKey('user.id'),default=False)
```

Tabella Prenotation

```
class Prenotation(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    field_id = db.Column(db.Integer, db.ForeignKey('field.id', ondelete='CASCADE'))
    player_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    date = db.Column(db.Date, nullable=False)
    start = db.Column(db.Time, nullable=False)
    end = db.Column(db.Time, nullable=False)
    price = db.Column(db.Float, nullable=False)
    __table_args__ = (db.UniqueConstraint(
                'field_id', 'date', 'start', 'end', name='_prenotation_uc'),
      )
```

FINE 👋