



UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA

DIMES

Corso di Laurea Magistrale in
Ingegneria Informatica

Progetto

Sistemi Distribuiti e Cloud Computing

Professori

Prof. Domenico Talia

Prof. Loris Belcastro

Candidato

Gianluca Massara

Matricola: 235159

Anno Accademico 2021/2022

Sommario

1. Introduzione	3
1.1 Traccia	3
2. Analisi dei Requisiti	3
2.1 Requisiti Funzionali	4
2.2 Requisiti non Funzionali	4
3. Progettazione e Sviluppo	5
3.1 Architettura	5
3.2 Database – MySQL	7
3.3 Backend – Spring	8
3.3.1 DAO Layer	9
3.3.2 Service Layer	10
3.3.3 Controller Layer	12
3.4 Frontend – Flutter	15
4. Azure – Distribuzione WebApp	16
4.1 Configurazione del servizio Azure	17
4.2 Accesso, trasferimento file ed Deploy sulla VM	19
5. MyLibrary WebApp – Risultato Finale	21

1. Introduzione

Il progetto si propone di creare un sistema distribuito che consenta agli utenti di caricare su un servizio di cloud immagini, note e commenti relativi ai libri che hanno letto e desiderano condividere informazioni su di essi. Fondamentalmente, si tratta di un sistema di recensioni dei libri. L'accesso a tale sistema avviene attraverso un'interfaccia web che consente di eseguire le operazioni menzionate in precedenza e di effettuare ricerche su vari libri per leggere le note lasciate da altri utenti o per lasciarne una propria.

Inoltre, il sistema deve sfruttare le risorse di calcolo, archiviazione e virtualizzazione offerte da Microsoft Azure. Questo documento si concentra inizialmente sull'analisi dei requisiti del sistema. Successivamente, vengono descritte le fasi che hanno portato allo sviluppo della piattaforma web, le scelte di progettazione principali, l'architettura del sistema e i servizi offerti agli utenti. Infine, vengono analizzate le diverse componenti utilizzate per la distribuzione su Azure.

1.1 Traccia

Lo sviluppo segue le indicazioni della seguente traccia:

Implementare un sistema distribuito per la creazione e la condivisione di contenuti relativi a libri letti tra un insieme di persone (ad esempio, amici, familiari o appassionati della lettura che vogliono condividere opinioni sui libri letti o informazioni su presentazioni di libri). Il sistema deve consentire agli utenti di creare e condividere note, commenti, file o altro, anche attraverso filtri che permettono di classificare i contenuti condivisi. Creare anche un front end web in un linguaggio a scelta per consentire di operare sul sistema.

2. Analisi dei Requisiti

La prima fase del progetto si è concentrata sull'analisi dei requisiti. Questo è un passaggio fondamentale nello sviluppo di un sistema software, poiché il suo obiettivo è definire le funzionalità che il nuovo prodotto deve offrire. Sono stati identificati i requisiti funzionali, ovvero le specifiche delle azioni che il sistema deve essere in grado di svolgere, come il caricamento di immagini, note e commenti, la ricerca di libri e la visualizzazione delle recensioni degli altri utenti. Inoltre, sono stati identificati anche i requisiti non funzionali, come le prestazioni del sistema, la sicurezza dei dati e

l'usabilità dell'interfaccia web. L'analisi dei requisiti ha fornito una base solida per la fase successiva dello sviluppo del sistema.

2.1 Requisiti Funzionali

Dalla traccia del progetto emerge che il sistema deve soddisfare i seguenti requisiti:

- Utilizzare le risorse di calcolo, storage e virtualizzazione offerte da Microsoft Azure.
- Permettere il caricamento di immagini sul Cloud.
- Consentire l'inserimento di note (recensioni con descrizione e titolo) sul Cloud.
- Accesso al sistema tramite Front-end Web per il caricamento e la ricerca dei contenuti.

In effetti, basandoci sulle specifiche fornite, è possibile dedurre ulteriori requisiti funzionali che possono essere ragionevoli per il sistema. Alcuni esempi potrebbero includere:

- Registrazione degli utenti: Il sistema dovrebbe consentire agli utenti di registrarsi per creare un account personale, fornendo le informazioni necessarie come nome, indirizzo e-mail e password.
- Creazione di recensioni: Oltre al caricamento di immagini, note e commenti, il sistema potrebbe consentire agli utenti di scrivere recensioni strutturate dei libri, includendo valutazioni nel formato di stelle (da 1 a 5).
- Filtri di ricerca avanzati: Oltre alla ricerca di libri per titolo o autore, il sistema potrebbe offrire opzioni di ricerca avanzate, come la ricerca per genere, anno di pubblicazione, valutazione media o altri attributi specifici.

Questi sono solo alcuni esempi di requisiti funzionali aggiuntivi che potrebbero essere considerati in base alle specifiche fornite. L'analisi completa dei requisiti consentirebbe di identificare ulteriori dettagli e funzionalità specifiche richieste dal sistema.

2.2 Requisiti non Funzionali

I requisiti non funzionali sono caratteristiche del software che non sono esplicitamente richieste dal cliente, ma influenzano il lavoro degli sviluppatori e descrivono come il sistema è in grado di eseguire determinati compiti. Per il progetto in questione, sono stati identificati i seguenti requisiti non funzionali:

- **Usabilità:** Un software è considerato usabile se è facile da utilizzare per gli utenti. L'usabilità è una qualità soggettiva che è fortemente influenzata dall'interfaccia grafica. Nel caso specifico, l'usabilità del sistema dipenderà dal front-end web che verrà realizzato, assicurando un'interfaccia intuitiva e user-friendly.
- **Prestazioni:** Le prestazioni sono qualità influenzate dall'efficienza del sistema. Un software è considerato efficiente se sfrutta in modo ottimale le risorse di calcolo disponibili. Per garantire questo requisito, l'utilizzo delle risorse di calcolo e virtualizzazione offerte da Azure risulta essere ottimale. Ciò consente di utilizzare risorse che possono essere allocate dinamicamente in base al carico di lavoro richiesto, assicurando prestazioni ottimali del sistema.
- **Affidabilità:** Un software è considerato affidabile se funziona come ci si aspetta e mantiene le sue funzionalità nel tempo. L'affidabilità può essere definita come la probabilità che il software si comporti come previsto per un certo intervallo di tempo. Nell'ambito del progetto, l'utilizzo delle risorse di cloud computing consente di aumentare l'affidabilità del sistema, fornendo una maggiore disponibilità e tolleranza ai guasti.
- **Robustezza:** Un sistema è considerato robusto se è in grado di gestire situazioni impreviste o eccezionali e di comportarsi in modo accettabile nonostante tali circostanze. Ad esempio, il sistema dovrebbe essere in grado di gestire correttamente il caricamento di dati non corretti o inattesi, senza causare malfunzionamenti o errori critici.

3. Progettazione e Sviluppo

L'applicazione, considerandone lo scopo, è stata denominata **MyLibrary**.



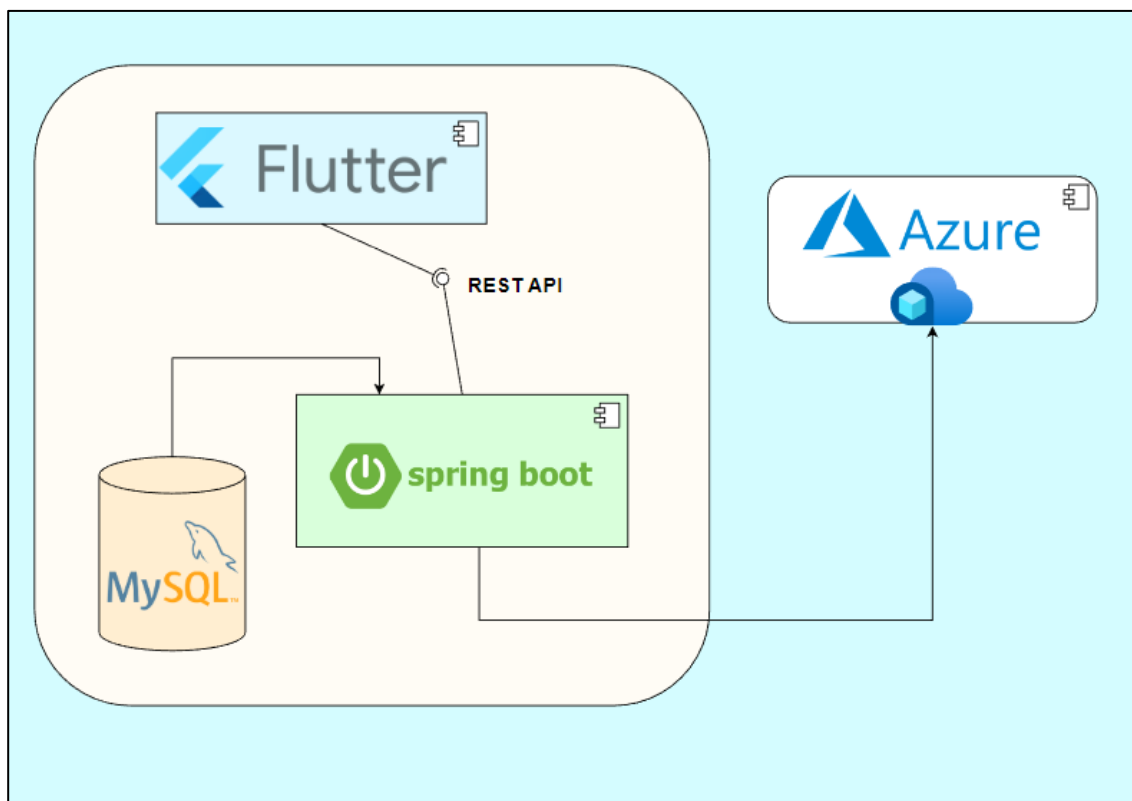
3.1 Architettura

In base al tipo di applicazione da sviluppare, possono essere adottate diverse metodologie di sviluppo. Ad esempio, se l'applicazione è relativamente semplice e deve gestire dati locali al

dispositivo senza condividerli con altri utenti in rete, è possibile sviluppare l'applicazione utilizzando metodi di archiviazione dati integrati all'interno dell'app stessa. Tuttavia, molto spesso è necessario sviluppare applicazioni più complesse, note come applicazioni enterprise, che interagiscono con database esterni al dispositivo e memorizzano dati condivisi tra molti utenti, come nel caso di un'e-commerce. In questo contesto, è buona prassi separare il front-end dal back-end per gestirli in modo indipendente.

Complessivamente l'architettura è suddivisa in 4 macro-componenti:

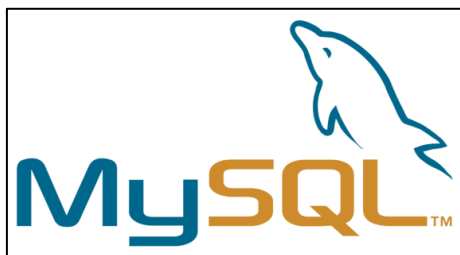
- Backend Server (Spring Framework)
- Frontend Server (Flutter Framework)
- Database (MySQL)
- Servizi Cloud (Azure) [di cui si parlerà in dettaglio in un capitolo dedicato]



Per quanto riguarda MyLibrary nello specifico, è stato utilizzato il framework open-source **Spring**, basato su Java, per lo sviluppo del back-end; e il framework open-source **Flutter**, basato sul linguaggio Dart, per la creazione del front-end. Inoltre, mentre per quanto riguarda la persistenza dei dati è stato utilizzato il RDBMS **MySQL**.

3.2 Database – MySQL

La scelta di utilizzare MySQL come RDBMS (Relational Database Management System) per l'implementazione dell'applicazione è stata motivata dalle sue caratteristiche principali e dalla sua ampia diffusione nel mondo dell'IT.

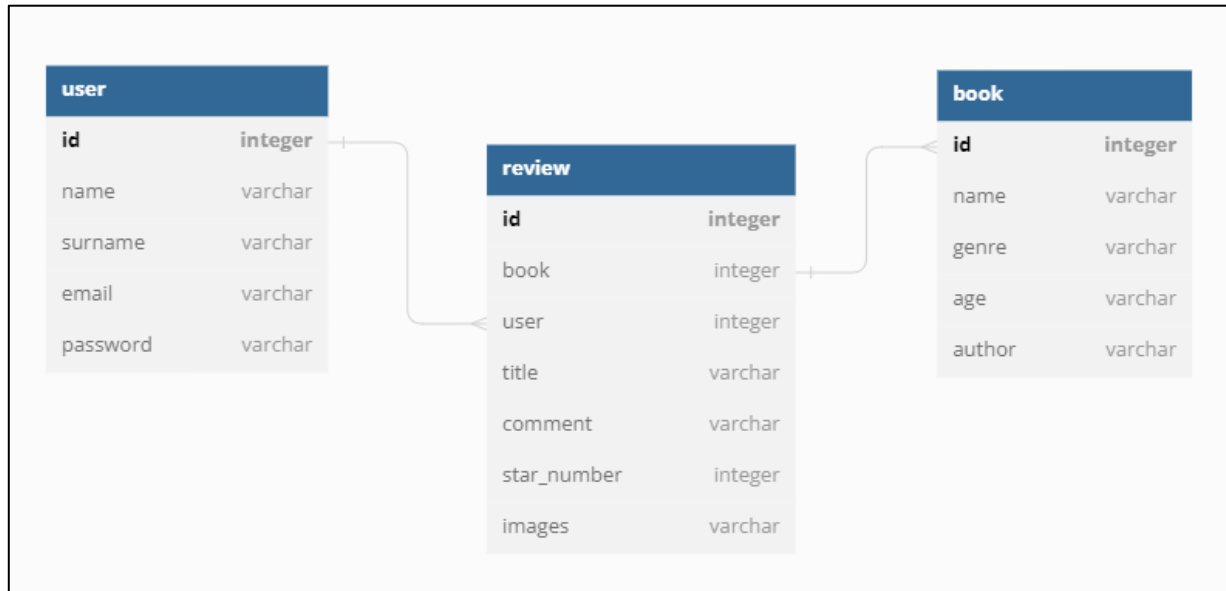


MySQL è un sistema di gestione di database relazionali open source e libero, che ha visto la luce nel 1996 come evoluzione di un preesistente DBMS relazionale chiamato mSQL. Nel corso degli anni, MySQL si è affermato come una delle tecnologie più riconosciute e utilizzate nel settore IT, trovando applicazione in numerosi software e siti web.

La scelta di MySQL per lo sviluppo dell'applicazione è stata influenzata da diversi fattori:

- **Alta efficienza:** Nonostante la gestione di grandi quantità di dati, MySQL è noto per la sua elevata efficienza nell'elaborazione delle query e nel recupero delle informazioni. Questo rende il database adatto a gestire carichi di lavoro impegnativi e ad affrontare situazioni di traffico elevato.
- **Funzionalità avanzate:** MySQL offre una vasta gamma di funzionalità tipiche dei migliori DBMS, come indici per l'ottimizzazione delle query, trigger per l'esecuzione di azioni automatiche in risposta a eventi specifici e stored procedure per l'esecuzione di logiche complesse direttamente all'interno del database. Queste funzionalità permettono di implementare soluzioni sofisticate e personalizzate.
- **Ampia integrazione:** MySQL è altamente integrabile con i principali linguaggi di programmazione e con gli ambienti di sviluppo più diffusi. Ciò consente agli sviluppatori di utilizzare le loro competenze esistenti e di integrare facilmente il database nelle applicazioni esistenti o in quelle in fase di sviluppo.

Complessivamente, la scelta di MySQL come RDBMS per l'applicazione offre vantaggi come l'efficienza nell'elaborazione dei dati, la disponibilità di funzionalità avanzate e l'integrazione agevole con le tecnologie esistenti.



Come si può vedere in figura, lo schema del database è molto semplice, si compone di sole 3 tabelle che rappresentano gli **Utenti**, i **Libri** e le **Recensioni** (una per ogni coppia Utente-Libro).

3.3 Backend – Spring



Il framework Spring Boot Web MVC fornisce un'architettura Model-View-Controller (MVC) preconfigurata che consente di separare gli aspetti distinti dell'applicazione, come la logica dei dati, la logica di business e la logica di input.

- **DAO Layer:** DAO, acronimo di Data Access Object, si interfaccia con una risorsa dati e fornisce un'interfaccia generica per accedervi. Questo strato consente di modificare i meccanismi di accesso ai dati senza influire sul codice che li utilizza.
- **Service Layer:** Questo strato incapsula la logica di business dell'applicazione e definisce le funzionalità offerte indipendentemente dalla modalità di accesso.

- Controller Layer: Questo strato espone le funzionalità dell'applicazione all'esterno attraverso un'interfaccia RESTful.

3.3.1 DAO Layer

In questo livello vengono definiti gli oggetti con i quali ci si interfaccia a livello fisico, si tratta delle tabelle del database ed anche di come queste vengono accedute. Spring, infatti, offre delle librerie per automatizzare l'esecuzione di query sul database che prendono il nome di JPA Repository.

Gli oggetti vengono rappresentati attraverso una classe di tipo Entity per ciascuna tabella:

```
@Entity
@Table(name = "user", schema = "my_library")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private int id;

    @Basic
    @Column(name = "name")
    private String name;

    @Basic
    @Column(name = "surname")
    private String surname;

    @Basic
    @Column(name = "email")
    private String email;

    @Basic
    @Column(name = "password")
    private String password;

    @OneToMany(mappedBy = "user")
    @JsonIgnore
    @ToString.Exclude
    private List<Review> reviews;
}
```

Come si può notare, questa classe rispecchia la esatta definizione della tabella vista nello schema di Database nell'immagine precedente.

La struttura è la stessa per le altre 2 tabelle. E poi sono presenti le classi relative alle Repository, che è dove vengono dichiarate le query che si vogliono poter eseguire sul Database.

Review Repository:

```
@Repository
public interface ReviewRepository extends JpaRepository<Review, Integer> {

    Gianluca
    List<Review> findByUserAndBook(User user, Book book);

    Gianluca
    List<Review> findAllByBookOrderByStarNumberDesc(Book book);

    Gianluca
    List<Review> findAllByBookAndCommentContainingOrderByStarNumberDesc(Book book, String keyword);

    Gianluca
    List<Review> findAllByBookAndStarNumberOrderByStarNumberDesc(Book book, int starNumber);

    Gianluca
    List<Review> findAllByBookAndStarNumberAndCommentContainingOrderByStarNumberDesc(Book book, int starNumber, String keyword);
}
```

User Repository:

```
@Repository
public interface UserRepository extends JpaRepository<User, Integer> {

    User findUserByEmailAndPassword(String email, String password);

    boolean existsUserByEmail(String email);
}
```

3.3.2 Service Layer

Il service layer implementa la logica di business dell'applicazione. Riceve le richieste dal controller layer, recupera i dati necessari dal DAO layer e si interfaccia con i servizi esterni. Per garantire una migliore modularità e coesione interna, il service layer è stato suddiviso in diversi componenti, ognuno con responsabilità specifiche. I principali componenti sono i seguenti:

- UserService: Questo componente si occupa della registrazione degli utenti nel sistema e dell'autenticazione degli stessi.

```
@Service
@Transactional(readOnly = true)
public class UserService {

    @Autowired
    private UserRepository userRepository;

    public User showUserByPassword(String email, String hashedPassword) {
        return userRepository.findUserByEmailAndPassword(email, hashedPassword);
    }

    @Transactional
    public User registerUser(User user) {
        if (userRepository.existsUserByEmail(user.getEmail()))
            return null;
        return userRepository.save(user);
    }
}
```

Come si può notare, viene ricevuto solamente il valore HASH della password, il quale viene poi salvato sul database.

- BookService: Gestisce tutte le operazioni relative ai libri, tra cui la ricerca (per nome, autore, genere, età).

- ReviewService: Gestisce tutte le operazioni di ricerca delle recensioni relative ad uno specifico libro ed anche la creazione delle stesse.

```

@Transactional
public Review saveReview(String jsonReview, String jsonFiles) throws JsonProcessingException {
    ObjectMapper mapper = new ObjectMapper();
    Review review = mapper.readValue(jsonReview, Review.class);
    byte[][] files = mapper.readValue(jsonFiles, byte[][].class);
    String[] images = review.getImages().split(regex: " ");

    if(files.length!=0)
        try {
            FileOutputStream fos;
            for(int i=0;i<files.length;i++) {
                fos = new FileOutputStream( name: path + images[i]);
                fos.write(files[i]);
                fos.close();
            }
        } catch (IOException e) {
            System.out.println("exception saving local file");
            return null;
        }

    return reviewRepository.save(review);
}

public List<Review> showByUserAndBook(int idUser, int idBook){
    Book book = new Book();
    User user = new User();
    book.setId(idBook);
    user.setId(idUser);
    return reviewRepository.findByUserAndBook(user,book);
}

public List<Review> showAllByBook(Book book) { return reviewRepository.findAllByBookOrderByStarNumberDesc(book); }
public List<Review> showAllByBookAndCommentContaining(Book book, String keyword){
    return reviewRepository.findAllByBookAndCommentContainingOrderByStarNumberDesc(book, keyword);
}

public List<Review> showAllByBookAndStarNumber(Book book, int starNumber){
    return reviewRepository.findAllByBookAndStarNumberOrderByStarNumberDesc(book, starNumber);
}

public List<Review> showAllByBookAndStarNumberAndCommentContaining(Book book, int starNumber, String keyword){
    return reviewRepository.findAllByBookAndStarNumberAndCommentContainingOrderByStarNumberDesc(book, starNumber, keyword);
}

```

Questa è probabilmente la classe più complessa in quanto al suo interno è realizzato il salvataggio dei file associati alla creazione di una nuova recensione. Il tutto è implementato tramite l'invio dei file in forma di array di bytes, i quali vengono rielaborati lato Backend e poi salvati sul filesystem così da poter essere visualizzati sul Frontend in un momento successivo.

3.3.3 Controller Layer

Questa è la classe in cui vengono esposti gli strumenti di comunicazione con il Backend, ovvero le interfacce che espongono le API ai client (nel nostro caso API REST). Sono presenti 3 diverse classi di Controller, una per ogni specifico oggetto Entity:

- BookController: è presente un solo metodo GET che riceve 4 parametri: nome, lista di età, lista di generi e lista di autori; viene effettuata la verifica all'interno del Database e vengono restituiti tutti quei libri il cui nome contiene la stringa "name" e il cui autore, genere ed età consigliata sono contenuti nelle relative liste.

```
@RestController
@RequestMapping(value = "/books", produces = "application/json;charset=UTF-8")
public class BookController {

    @Autowired
    private BookService bookService;

    @PostMapping()
    public List<Book> getByName(@RequestParam String name, String listAges, String listGenres, String listAuthors){
        try {
            return bookService.showAllBook(name, listAges, listGenres, listAuthors);
        } catch (JsonProcessingException e) {
            System.out.println("exception JSON book controller");
            throw new RuntimeException(e);
        }
    }
}
```

- ReviewController: contiene svariati metodi per la ricerca di una recensione tramite nome, oppure per numero di stelle valutazione (o entrambi), inoltre è presente un metodo POST tramite il quale vengono ricevuti i dati relativi a nuove recensioni appena create.

```
@RestController
@RequestMapping(value = "/reviews", produces = "application/json;charset=UTF-8")
public class ReviewController {

    @Autowired
    private ReviewService reviewService;

    @PostMapping(value = "/own")
    public List<Review> getByUserAndBook(int idUser, int idBook){
        return reviewService.showByUserAndBook(idUser,idBook);
    }

    @PostMapping(value = "/save")
    public Review saveReview(@RequestParam String jsonReview, @RequestBody String jsonFiles) {
        try {
            return reviewService.saveReview(jsonReview, jsonFiles);
        } catch (JsonProcessingException e) {
            System.out.println("Json exception in saveReview controller");
            return null;
        }
    }

    @PostMapping(value = "/all")
    public List<Review> getAllByBook(@RequestBody Book book){
        return reviewService.showAllByBook(book);
    }

    @PostMapping(value = "/keyword")
    public List<Review> getAllByBookAndCommentContaining(@RequestBody Book book, @RequestParam String keyword){
        return reviewService.showAllByBookAndCommentContaining(book, keyword);
    }

    @PostMapping(value = "/star")
    public List<Review> getAllByBookAndStarNumber(@RequestBody Book book, @RequestParam int starNumber){
        return reviewService.showAllByBookAndStarNumber(book, starNumber);
    }

    @PostMapping(value = "/star_keyword")
    public List<Review> getAllByBookAndStarNumberAndCommentContaining(@RequestBody Book book, @RequestParam int starNumber, @RequestParam String keyword){
        return reviewService.showAllByBookAndStarNumberAndCommentContaining(book, starNumber, keyword);
    }
}
```

- UserController: contiene i metodi per effettuare il login o la registrazione di un nuovo utente.

```
@RestController
@RequestMapping(value = "/user", produces = "application/json;charset=UTF-8")
public class UserController {

    @Autowired
    private UserService userService;

    @PostMapping("/login")
    public User loginUser(String email, String password) { return userService.showUserByPassword(email, password); }

    @PostMapping("/register")
    public User registerUser(@RequestBody User user) { return userService.registerUser(user); }
}
```

3.4 Frontend – Flutter

Una volta completato lo sviluppo del back-end, è stato affrontato il front-end, che rappresenta sostanzialmente la parte visibile e interattiva del sistema per l'utente. Il front-end è responsabile dell'acquisizione dei dati in ingresso e della loro elaborazione in conformità con specifiche predefinite, al fine di renderli utilizzabili dal back-end.



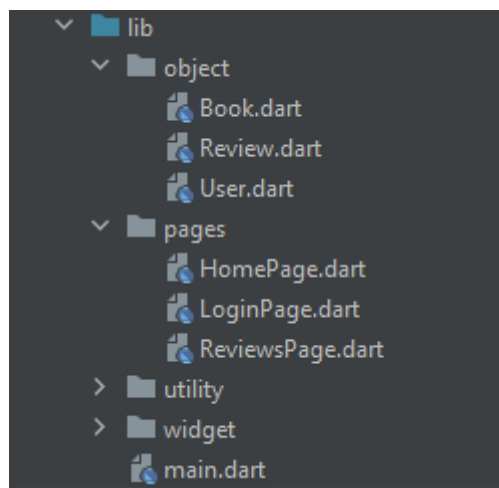
Nel caso specifico, il front-end è stato sviluppato utilizzando Flutter, un framework che consente la realizzazione di applicazioni mobili, desktop e web utilizzando un unico codebase. Sono state diverse le ragioni che hanno portato all'adozione di Flutter per lo sviluppo dell'App MyLibrary:

- Velocità: il codice Flutter viene compilato in codice macchina ARM o Intel, consentendo prestazioni veloci su diversi dispositivi.
- Produttività: Flutter permette di sviluppare in modo rapido e semplice, consentendo l'aggiornamento istantaneo del codice senza perdita di tempo.
- Flessibilità: Flutter consente il controllo di ogni pixel per creare applicazioni personalizzate e design adattivi che si adattano a qualsiasi schermo.
- Supporto: Essendo sviluppato e utilizzato da Google, Flutter gode di una vasta community a livello globale che fornisce supporto e risorse.

Rispetto ad altri framework, Flutter è stato costruito da zero, consentendo lo sviluppo di applicazioni semplici e ad alte prestazioni. Per raggiungere questo obiettivo, Flutter viene eseguito nativamente su ogni piattaforma utilizzando la compilazione AOT (Ahead-of-Time), mentre durante lo sviluppo viene utilizzata la compilazione JIT (Just-in-Time) per accelerare il processo di test. Ogni applicazione basata su Flutter è composta da Widget, che possono essere contenitori, testi, immagini e altro ancora. Ogni widget viene interpretato e rappresentato su una "Canvas" gestita dal motore grafico Skia. La piattaforma mostra il widget finale all'utente e intercetta gli eventi generati dall'interazione utente, inoltrando tali eventi all'applicazione.

La struttura del progetto Flutter è la seguente:

- **Objects:** Definisce i modelli dei dati. Mappa gli oggetti Json (che si ricevono dal back-end) in oggetti Dart.
- **Utility:** Contiene classi di utilità, un'interfaccia per l'inoltro delle richieste al back-end Spring. Contiene il RestManager che raccoglie le richieste da parte dell'ApiController, le incapsula in richieste HTTP e le inoltra al back-end.
- **Pages:** Definisce la User Interface dell'applicazione, con le varie schermata principali che vengono visualizzate.
- **Widget:** come dice la parola stessa, contiene delle particolari strutture grafiche che vengono riutilizzate all'interno di più pagine.



4. Azure – Distribuzione WebApp



Nella fase finale del progetto, la Web-App è stata distribuita su Azure utilizzando il servizio di calcolo Azure Virtual Machine. Questo servizio consente di sfruttare le risorse di calcolo e virtualizzazione di Azure per eseguire un sistema operativo sul quale esportare la Web-App. Il servizio supporta sia

Windows che Linux, fornisce sistemi di auto-scaling per gestire in modo dinamico le risorse in base al carico di lavoro e offre diverse opzioni per il deployment.

In aggiunta al servizio Azure Virtual Machine, è stato utilizzato un server Nginx. Nginx è un server web ad alte prestazioni che agisce come reverse proxy e gestisce la distribuzione del traffico verso l'applicazione web. L'aggiunta di un server Nginx consente di migliorare le prestazioni e la sicurezza dell'applicazione distribuita.

In sintesi, la distribuzione della Web-App su Azure è stata effettuata utilizzando il servizio Azure Virtual Machine, che offre un ambiente virtuale completo, e con l'aggiunta di un server Nginx per ottimizzare le prestazioni e la sicurezza dell'applicazione.

4.1 Configurazione del servizio Azure

La sottoscrizione Azure for Student mette a disposizione quasi tutti i servizi disponibili ed un credito iniziale di 100\$, per questo motivo si è deciso di optare per la configurazione di un'unica macchina virtuale di tipo **B2** dato il prezzo non troppo elevato e un quantitativo di risorse adeguato allo svolgimento del progetto.

Dimensioni macchina virtuale ↑↓	Tipo ↑↓	CP... ↑↓	RA... ↑↓	Numer... ↑↓	Arch... ↑↓	Disco Premium ↑↓	Costo/mese ↑↓
<div> <div>></div> <div>Più usate dagli utenti di Azure</div> </div>				Dimensioni più usate dagli utenti in Azure			
<div> <div>></div> <div>Serie D v4</div> </div>				Dimensioni della famiglia D di quarta generazione per le esigenze di lavoro che richiedono prestazioni della serie D			
<div> <div>✓</div> <div>Serie B</div> </div>				Ideale per carichi di lavoro che non richiedono prestazioni della serie D			
B1ls	Utilizzo generico	1	0.5	2 320	4	Supportata	4,82 USD
B1s	Utilizzo generico	1	1	2 320	4	Supportata	9,64 USD
B1ms	Utilizzo generico	1	2	2 640	4	Supportata	19,27 USD
B2s	Utilizzo generico	2	4	4 1280	8	Supportata	38,54 USD
B2ms	Utilizzo generico	2	8	4 1920	16	Supportata	77,38 USD

Come si può notare dall'immagine successiva, la creazione di una risorsa di tipo VM con le specifiche e le impostazioni necessarie alla distribuzione della WebApp, implica la creazione delle seguenti risorse:

Tutti i servizi >

Tutte le risorse

Università della Calabria (studenti.unical.it)

+ Crea | Gestisci visualizzazione | Aggiorna | Esporta in CSV | Apri query | Assegna tag | Elimina

Filtra per qualsiasi cam... | Sottoscrizione uguale a **tutte le** | Gruppo di risorse uguale a **tutte le** | Tipo uguale a **tutte le** | Località uguale a **tutte le** | Aggiungi filtro


1 Risorse non protette | 1 Elementi consigliati

<input type="checkbox"/> Nome ↑↓	Tipo ↑↓	Gruppo di risorse ↑↓	Località ↑↓	Sottoscrizione ↑↓
<input type="checkbox"/> MyLibraryApp	Macchina virtuale	CloudProject	Norway East	Azure for Students
<input type="checkbox"/> MyLibraryApp-ip	Indirizzo IP pubblico	CloudProject	Norway East	Azure for Students
<input type="checkbox"/> MyLibraryApp-nsg	Gruppo di sicurezza di rete	CloudProject	Norway East	Azure for Students
<input type="checkbox"/> mylibraryapp362_z1	Interfaccia di rete	CloudProject	Norway East	Azure for Students
<input type="checkbox"/> MyLibraryApp_OsDisk_1_b9b601e7a83945b7a91ddb1037accb7e	Disco	CLOUDPROJECT	Norway East	Azure for Students
<input type="checkbox"/> myVPN	Rete virtuale	CloudProject	Norway East	Azure for Students
<input type="checkbox"/> NetworkWatcher_norwayeast	Network Watcher	NetworkWatcherRG	Norway East	Azure for Students

Tra cui, ovviamente, un indirizzo IP pubblico, tramite il quale accedere alla porta 80 esposta sulla macchina stessa sulla quale è hostata la WebApp tramite server Nginx.

La VM creata ospita un sistema operativo Linux (distribuzione Ubuntu 20), il quale è stato scelto per la semplicità con quale è possibile avviare servizi e gestire la macchina tramite Terminale. Infatti, per potere accedere alla suddetta macchina, è stato necessario settare un nome utente ed una password per consentire il collegamento tramite protocollo SSH (molto noto ed usato, soprattutto in ambiente Linux per l'accesso remoto in sicurezza alle proprie macchine).

Le risorse a disposizione sulla macchina sono modeste, come anticipato, ma sufficienti allo scopo:

 Dimensioni	
Dimensioni	Standard B2s
CPU virtuali	2
RAM	4 GiB

4.2 Accesso, trasferimento file ed Deploy sulla VM

Come detto in precedenza, l'accesso alla risorsa in Cloud è possibile tramite il servizio SSH che è preinstallato nella maggior parte delle distribuzioni Linux, in seguito al collegamento, i vari step per il completamento del Deploy sono stati eseguiti sul terminale Linux.

Innanzitutto, è stato necessario aggiornare il sistema per essere sicuro che le dipendenze e le librerie fossero presenti e compatibili, quindi sono stati eseguiti i comandi:

- **sudo apt update**
- **sudo apt upgrade**

Successivamente sono stati installati tutti i software necessari al proseguimento:

- **Java JDK:** necessaria ad avviare l'eseguibile JAR che è stato ottenuto tramite il processo di packaging standard che è fornito dal gestore di dipendenze Maven integrato nel Framework Spring con il quale è stato sviluppato il Backend.

```
Gianluca@MyLibraryApp:~$ java --version
openjdk 17.0.7 2023-04-18
OpenJDK Runtime Environment (build 17.0.7+7-Ubuntu-0ubuntu120.04)
OpenJDK 64-Bit Server VM (build 17.0.7+7-Ubuntu-0ubuntu120.04, mixed mode, sharing)
Gianluca@MyLibraryApp:~$
```

- **Server Nginx:** necessario ad ospitare la WebApp sulla porta 80. Il suo funzionamento è molto semplice, una volta installato è sufficiente creare un file di configurazione di questo tipo:

```
Gianluca@MyLibraryApp:~$ cat /etc/nginx/sites-enabled/my_library.conf
# Default server configuration
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/my_library/web;
    index index.html;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

In cui viene specificato la porta da esporre e la cartella root del progetto Frontend che si vuole Deployare. La cartella in questione è denominata **my_library/web** ed è ottenuta tramite il processo di Web-Build fornito dal Framework Flutter.

- **MySQL server:** necessario per mantenere la persistenza dei dati. Il software è stato configurato ancora una volta tramite comandi sul terminale, tra cui **CREATE** ed **INSERT** tramite i quali sono state replicate le tabelle User, Review e Book viste in precedenza, e popolate con righe relative ad un insieme di Libri, Recensioni ed Utenti che sono stati generati tramite semplici script o reperite in internet in formato di Excel Sheets.

```
mysql> describe user;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id     | int           | NO   | PRI | NULL    | auto_increment |
| name   | varchar(45)   | NO   |     | NULL    |                |
| surname | varchar(45)   | NO   |     | NULL    |                |
| email  | varchar(60)   | NO   |     | NULL    |                |
| password | varchar(200) | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> describe book;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id     | int           | NO   | PRI | NULL    | auto_increment |
| name   | varchar(100)  | NO   |     | NULL    |                |
| genre  | varchar(45)   | NO   |     | NULL    |                |
| age    | varchar(50)   | NO   |     | NULL    |                |
| author | varchar(60)   | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> █
```

I file necessari, tra cui il JAR eseguibile del Backend, la cartella root Web/ e gli script SQL di generazione del Database sono stati trasferiti sulla VM tramite il seguente comando:

```
scp ./FileArchive.rar Gianluca@51.120.0.147:/home/Gianluca/CloudProject
```

Cioè si è utilizzato SCP (Secure Copy), un protocollo di rete basato su SSH (Secure Shell) utilizzato per il trasferimento sicuro di file tra un computer locale e un server remoto.

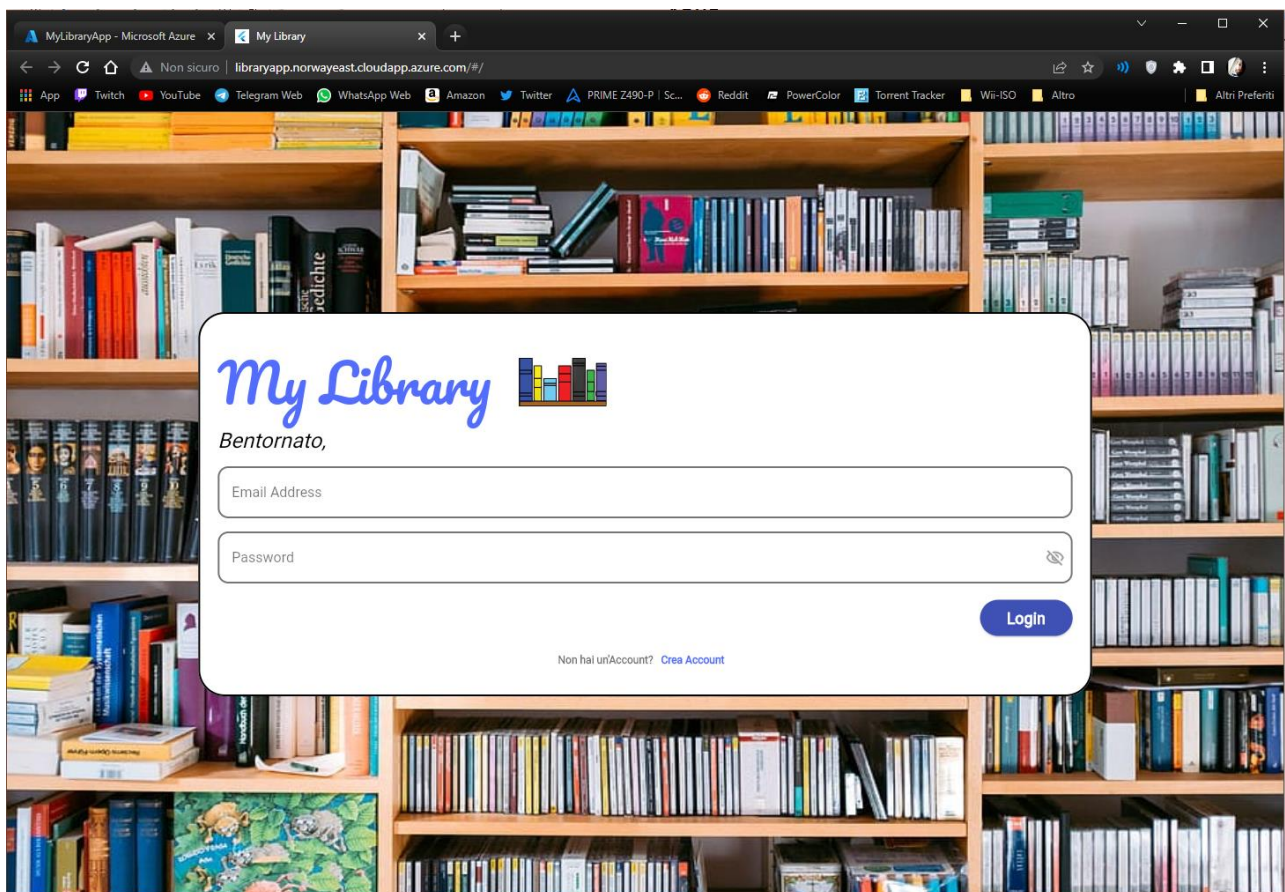
Per concludere l'intera procedura di Deploy, e per rendere automatico il processo di esecuzione dei vari servizi, tra cui Backend Server e MySQL server, è stato creato uno script, il quale è stato impostato per essere eseguito ogni volta che la VM Azure viene avviata.

5. MyLibrary WebApp – Risultato Finale

Per potere testare il prodotto finale, una volta avviata la VM è possibile collegarsi tramite qualsiasi browser all'indirizzo IP della stessa (51.120.0.147), o meglio ancora, digitando il suo DNS (Domain Name Service):

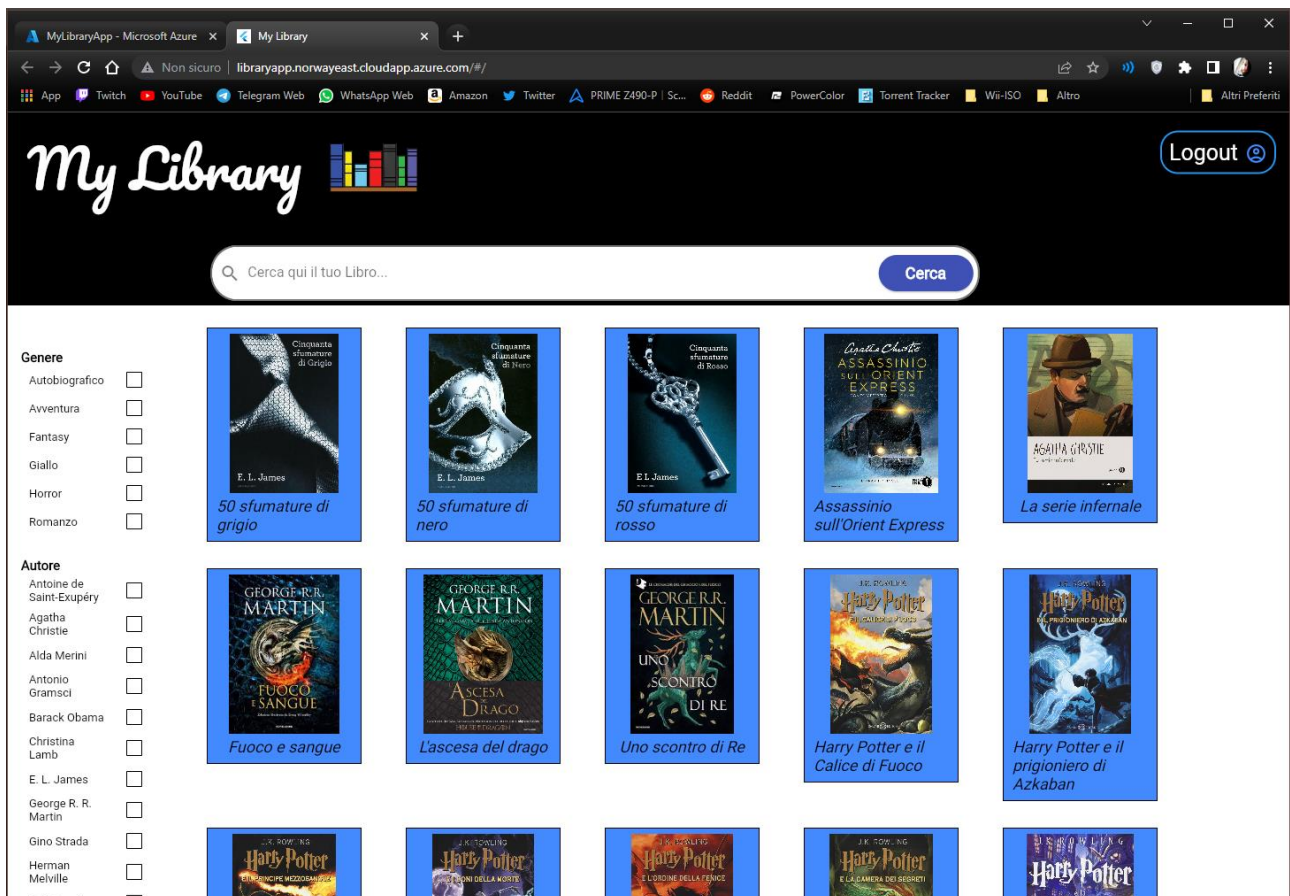
<http://libraryapp.norwayeast.cloudapp.azure.com>

A questo verrà visualizzata la pagina iniziale, cioè quella di Login:



Da qui è possibile effettuare il Login, ovviamente, oppure creare un nuovo Account.

Una volta eseguita l'autenticazione, si verrà trasferiti sulla Home Page:



Qui vengono visualizzati un elenco di Libri di cui è possibile leggere le recensioni di altri utenti. I libri stessi possono essere cercati tramite Titolo e filtrati in base al Genere, Autore o Età, come è possibile vedere nel menù sulla sinistra.

Cliccando su uno dei tanti libri elencati, è possibile accedere alla pagina delle recensioni:

Recensioni

Hai letto questo libro? Lascia qui una recensione!

Valutazione complessiva:

☆☆☆☆☆

Aggiungi una foto alla tua recensione!

Titolo:

Scegli un titolo per la tua recensione..

Recensione:

Scrivi qui la tua recensione..

Salva

Cerca per parola chiave... **Cerca**

Filtra per:

Tutte le stelle ★

TSENG BONDUR

★★★★★

Personaggi indimenticabili

Questo libro è un capolavoro dell'immaginazione. L'autore ha creato un mondo affascinante e ricco di dettagli, che cattura immediatamente l'attenzione del lettore. La trama è avvincente e piena di colpi di scena, mantenendo costantemente alta la suspense. I personaggi sono ben


Come si può notare, qui sono presenti le funzionalità descritte prima, e cioè la possibilità di fornire una valutazione al libro attraverso un sistema a stelle (da 0 a 5), inserire un Titolo alla propria “recensione” ed una nota che descriva la propria opinione sul libro. Insieme a tutto questo è possibile selezionare delle foto da aggiungere al contenuto che si vuole condividere:

Nella stessa pagina è ovviamente possibile scorrere fra i contenuti condivisi da altre persone, che anche possibile cercare tramite parole chiave o filtrare in base al gradimento (espresso in numero di stelle):

Cerca

Filtra per:




Tutte le stelle ★


 MURPHY DIANE

★★★★★

Un viaggio di piacere e dolore

"50 sfumature di nero" è una lettura **intrigante** e appassionante che porta il lettore in un viaggio di piacere e dolore. Il secondo libro della trilogia "50 sfumature" non delude le aspettative. È pieno di scene piccanti e sorprese inaspettate.




 BONDUR LOUI

★★★★★

Scrittura fluida ma a volte troppo melodramma

"50 sfumature di nero" è un'esperienza emozionante che ti fa venire voglia di continuare a leggere la trilogia. Ho trovato il secondo libro un po' troppo simile al primo, ma comunque una lettura **intrigante** per gli amanti del genere.

 FIRRELLI JEFF

★☆☆☆☆

Scene piccanti e sorprese inaspettate

Se ti è piaciuto il primo libro, "50 sfumature di nero" ti catturerà ancora di più con una trama più profonda e personaggi ancora più interessanti. Non ho trovato "50 sfumature di nero" altrettanto coinvolgente del primo libro, ma comunque una lettura **intrigante** per gli amanti del genere.