

# Progetto Programmazione Avanzata

Sandro Wu

A.A 22-23

## Indice

<b>1</b>	<b>Panoramica</b>	<b>2</b>
1.1	Idea del progetto . . . . .	2
1.2	Inizializzazione Database . . . . .	2
<b>2</b>	<b>Applicazione</b>	<b>3</b>
2.1	Schermate dell'applicazione . . . . .	3
2.1.1	Login Page . . . . .	3
2.1.2	Home Page . . . . .	3
2.1.3	Services e Subscription . . . . .	4
2.1.4	All Services . . . . .	5
2.1.5	Activity Log . . . . .	5
2.1.6	Cost Center . . . . .	6
2.2	Interazione con Server . . . . .	6
2.3	Struttura e scelte progettuali . . . . .	6
<b>3</b>	<b>Server</b>	<b>7</b>
3.1	Sessione e Token . . . . .	7
3.2	Spring JPA . . . . .	7
3.3	Lista API . . . . .	7

# 1 Panoramica

## 1.1 Idea del progetto

L'idea dell'applicazione è un Manager di servizi, in questo caso Servizi Cloud (ispirazione da Microsoft Azure). In breve l'applicazione offre la possibilità per l'utente di visualizzare, creare, eliminare i propri servizi e modificarne lo stato, visualizzare i log delle azioni eseguite ed i costi sostenuti per i servizi.

L'applicazione necessita di dati forniti dall'esterno. Per esempio i dati dei servizi offerti dalla piattaforma cloud, i dati dei costi fatturati all'utente.

## 1.2 Inizializzazione Database

Il progetto è composto da due repository Applicazione e Server.

Creare prima il database dall'Applicazione. Solo dopo ciò sarà possibile avviare il Server.

Quindi, per inizializzare il database:

- All'avvio dell'Applicazione, comparirà la schermata di login.
- E' presente un bottone in basso a destra "Initialize database".
- Cliccandoci si crea (e sovrascrive) il database e si popola con i dati predefiniti.

L'Applicazione si connette direttamente a MySQL con parametri:

```
IP: 127.0.0.1
Porta: 3306
Database: <Matricola>
Username : root
Password : root
```

I file coinvolti sono:

```
> src/main/java/progetto/applicazione/InitDB.java
> src/main/resources/progetto/applicazione/data.json
> src/main/resources/progetto/applicazione/create_table.sql
```

Il database creato ha per semplicità tabelle legati con dei "falsi" foreign keys, quindi di fatto indipendenti tra loro. Lo scopo è usare così Spring JPA solo a livello di base, senza entrare a livelli troppo tecnici, poiché fuori dall'obiettivo del progetto.

## 2 Applicazione

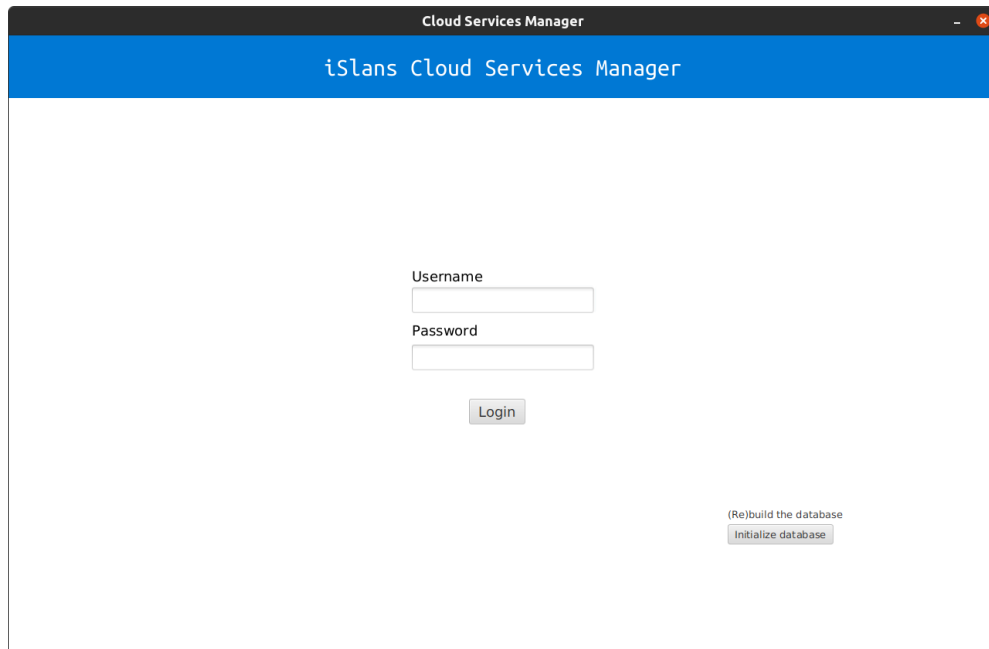
### 2.1 Schermate dell'applicazione

#### 2.1.1 Login Page

All'avvio dell'applicazione è richiesto il login. Non è prevista la registrazione di nuovi utenti dall'applicazione. L'account di test è

User: Test

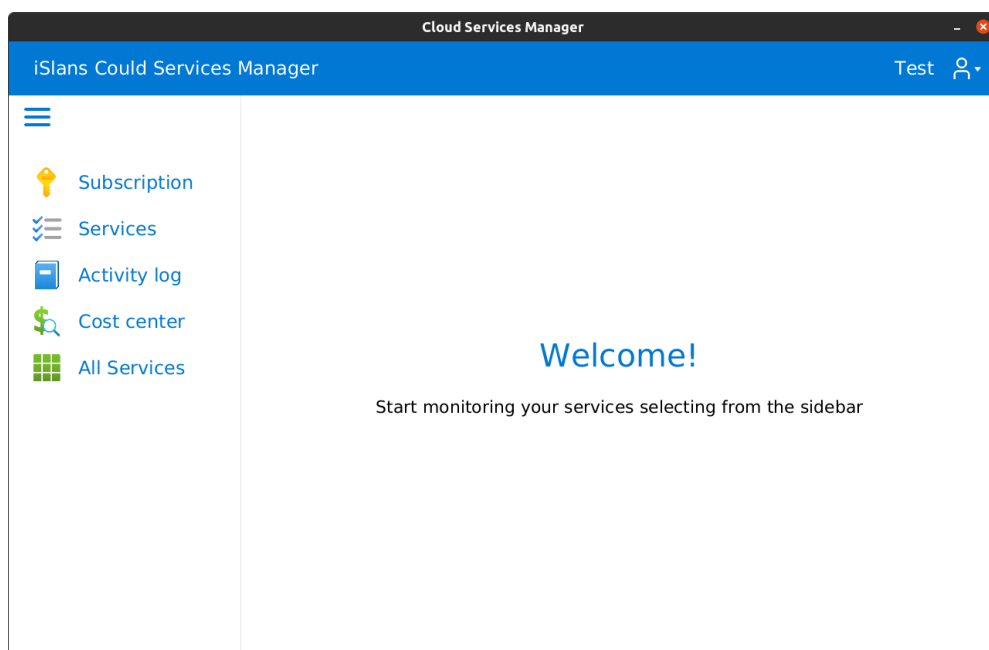
Pass: test



#### 2.1.2 Home Page

L'interfaccia è composta da:

- TopBar, la barra in alto. Mostra a destra l'username e un'icona. L'icona contiene un menù a scomparsa con l'opzione di logout.
- SideBar, a sinistra. Contiene un menù ad hamburger per espandersi/chiudersi, ed i vari link per le pagine dell'applicazione
- MainBody è tutta la parte restante, che viene caricato dinamicamente in base alla pagina selezionata nel SideBar



2.1.3 Services e Subscription

Qui vi sono i servizi e sottoscrizioni dell'utente. Attraverso il menù a scomparsa, si può

- attivare / fermare / eliminare i servizi
- attivare / fermare le sottoscrizioni

Cloud Services Manager

iSlans Could Services Manager

Test

ID	Name	Service	Category	Subscri...	Plan	Status
1	Ubuntu 18.04	Virtual Machi...	Compute	Base subscri...	Basic plan	Active
2	Device DB	SQL Database	Databases	Base subscri...	Basic plan	Active
3	device API	Functions	Web	Base subscri...	Basic plan	Active
4	Storage	Storage Acc...	Storage	Base subscri...	Basic plan	Active

Cloud Services Manager

iSlans Could Services Manager

Test

ID	Name	Status
1	Base subscription	Active

### 2.1.4 All Services

Sono mostrati tutti i servizi offerti attualmente dalla piattaforma Cloud. Dal menù a scomparsa, l'utente può attivare un nuovo servizio, apparirà un Dialog con le impostazioni del nuovo servizio da attivare.

The screenshot shows the 'Cloud Services Manager' application window. The title bar says 'Cloud Services Manager'. The main header is 'iSlans Could Services Manager' with a 'Test' button and a user icon. Below the header is a table with columns 'ID', 'Name', and 'Category'. The table lists 16 services, including Virtual Machines, Disks, Kubernetes Services, Virtual Machine Scale Sets, Virtual Machines (Classic), Service Fabric Clusters, Batch Accounts, App Services, App Service Plans, Functions, Continuous WebJobs, Triggered WebJobs, API Management Services, App Services, App Service Plans, and CDN Profiles. A dialog box titled 'Create a new resource' is open over the 'App Services' row. The dialog shows the service '(ID: 14) App Services [Web]', a text field for 'Resource Name' with the placeholder 'Insert the resource name', a dropdown for 'Subscription' with '[1] Base subscription' selected, and a dropdown for 'Plan' with '[1] Basic Plan' selected. There are 'Cancel' and 'Create' buttons at the bottom of the dialog.

ID	Name	Category
1	Virtual Machines	Compute
2	Disks	Compute
3	Kubernetes Services	Compute
4	Virtual Machine Scale Sets	Compute
5	Virtual Machines (Classic)	Compute
6	Service Fabric Clusters	Compute
7	Batch Accounts	Compute
8	App Services	Compute
9	App Service Plans	Compute
10	Functions	Web
11	Continuous WebJobs	Web
12	Triggered WebJobs	Web
13	API Management Services	Web
14	App Services	Web
15	App Service Plans	Web
16	CDN Profiles	Web

**Create a new resource**

Service: (ID: 14) App Services [Web]

Resource Name:

Subscription: [1] Base subscription

Plan: [1] Basic Plan

other settings...

Cancel Create

### 2.1.5 Activity Log

Vi è lo storico delle azioni eseguite dall'utente.

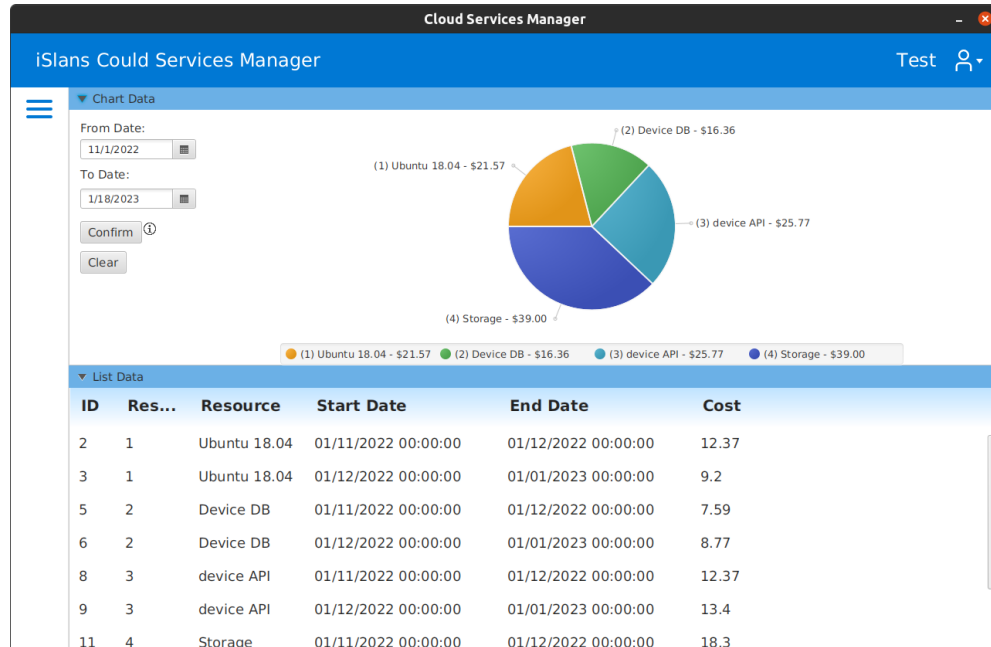
The screenshot shows the 'Cloud Services Manager' application window. The title bar says 'Cloud Services Manager'. The main header is 'iSlans Could Services Manager' with a 'Test' button and a user icon. Below the header is a table with columns 'ID', 'Time', and 'Log'. The table shows 10 log entries, including creating and stopping resources for Virtual Machines, SQL Database, Functions, and Storage Accounts. The log entries are as follows:

ID	Time	Log
1	02/10/2022 09:32:55	[12080] Created resource - Virtual Machines - Ubuntu 18.04
2	11/10/2022 11:03:30	[12080] Created resource - SQL Database - Device DB
3	03/10/2022 15:23:12	[12080] Created resource - Functions - device API
4	17/10/2022 10:21:40	[12080] Created resource - Storage Accounts - Storage
97	15/01/2023 19:08:45	[30102] Stopped resource - Functions - device API
98	15/01/2023 19:08:49	[30102] Stopped resource - Storage Accounts - Storage
99	15/01/2023 19:08:54	[30110] Activated resource - Functions - device API
100	15/01/2023 19:08:56	[30110] Activated resource - Storage Accounts - Storage

### 2.1.6 Cost Center

Sono mostrati i costi dei servizi usufruiti. Di default, sono prelevati i dati degli ultimi tre mesi. La pagina ha due sezioni: Char Data e List Data

- In Chart Data, viene mostrato nel grafico i costi sostenuti per servizio, nel periodo di tempo selezionato.
- In List Data sono mostrati i singoli costi.



## 2.2 Interazione con Server

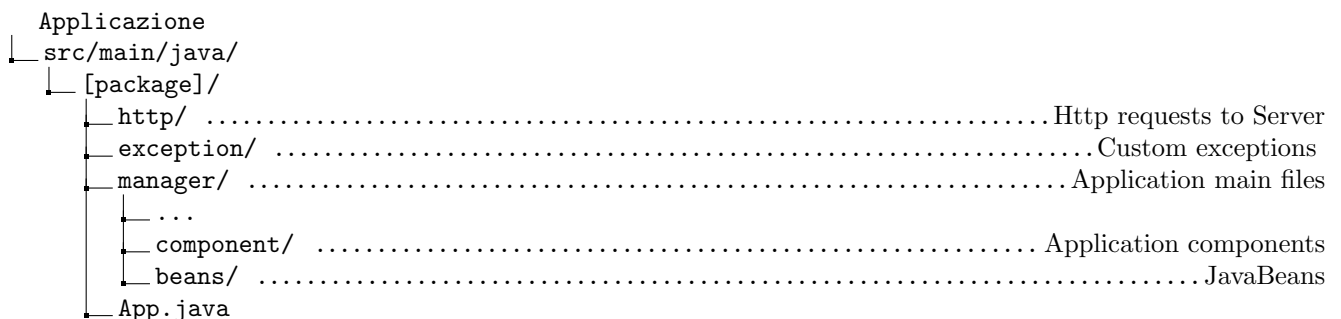
Per le richieste http nell'applicazione, è stato usato la libreria Java Http Client (java.net.http) Vi è una classe `Request.java`<sup>1</sup> che utilizza Http Client ed espone i metodi utilizzati dall'applicazione per la comunicare con il Server.

Le risposte del Server sono in formato json. Sono decodificate, a seconda delle situazioni, con deserializzazione in classi specifiche o in JsonObject generico.

Se le richieste http non vanno a buon fine (e.g. Server non disponibile), comparirà un relativo Alert Dialog per avvisare l'utente.

## 2.3 Struttura e scelte progettuali

Dato il numero non basso di file nel progetto, i file java sono organizzati in più sottocartelle (subpackage)



Una scelta progettuale di interesse può essere riguardo le diverse schermate dell'applicazione mostrate in precedenza. Questi hanno due componenti in comune tra loro, TopBar e SideBar; mentre il MainBody cambia in base alla schermata. Questa struttura modulare si riflette nei file fxml ed i relativi Controller. Vi è un `manager.fxml`<sup>2</sup> come schermata di base, che include i componenti fxml necessari, caricati staticamente fin dall'inizio o dinamicamente durante il funzionamento.

<sup>1</sup>src/main/java/progetto/applicazione/http/Request.java

<sup>2</sup>src/main/resources/progetto/applicazione/manager.fxml

## 3 Server

Il Server si avvia solo se il database è stato creato.

Le API esposte sono con prefisso `/api/v1`

Le risposte delle API sono in formato json

### 3.1 Sessione e Token

Dopo la prima autenticazione con username e password, il server genera e restituisce un Token. Questo sarà necessario al client per accedere alle API che interagiscono sulle risorse dell'utente.

Il Token generato ha una scadenza, dopo la quale non sarà più valido.

Se un API riceve un Token non valido, risponderà con lo stato 401 Unauthorized.

### 3.2 Spring JPA

Per usare Spring JPA, sono state mappate le principali tabelle del database in Entities <sup>3</sup> ed usato i JPA Repository <sup>4</sup>

Per le query semplici sono utilizzate i derived Query di Spring JPA. Mentre quelle non immediate con join di tabelle, per semplicità sono create usando nativeQuery.

### 3.3 Lista API

Riporto qui brevemente la lista delle API esposte, senza entrare nei dettagli sui dati in input richiesti e le risposte date.

```
GET  /user/costs
GET  /user/activity
POST /user/login

GET  /resource/user
POST /resource/create
POST /resource/activate
POST /resource/stop
DELETE /resource

GET  /subscription/user
POST /subscription/activate
POST /subscription/stop

GET  /service/all
GET  /service/plans
```

---

<sup>3</sup>src/main/java/progetto/server/database/table/...

<sup>4</sup>src/main/java/progetto/server/database/...