Cartella sanitaria

Progetto universitario d’informatica – linguaggio Java

Specifiche Architetturali tecniche

# Definizione architetturale

## Gestione input/output

Per il progetto è richiesto una semplice interazione tramite console di comando, per questo motivo il software sarà un unico eseguibile in linguaggio Java.

L’eseguibile verrà incapsulato in un executable jar standard eseguibile da una JRE installata sul sistema operativo, questo consentirà di avere un eseguibile cross-platform compatibile con diversi SO.

### Referenze

Definizione e spiegazione dei JAR files.

<https://docs.oracle.com/en/java/javase/21/docs/specs/jar/jar.html>

## Gestione dei dati

Il progetto richiede un salvataggio e un caricamento di dati da un sistema di persistenza e specifica che deve essere un file su file system. Per questo motivo non verranno usati DB, ma semplici scritture e letture su unico file. Il sistema leggerà i dati all’avvio e li manterrà in sessione, finita una transazione funzionale il file verrà aggiornato rimpiazzando il precedente.

I dati verranno scritti serializzandoli in un formato JSON standard.

I dati non saranno criptati in quanto il progetto non richiede questo tipo di sicurezza.

### Referenze

Definizione e spiegazione dei JSON

<https://datatracker.ietf.org/doc/html/rfc8259>

## Date

Le date usate nel software avranno il seguente formato: *yyyy-MM-dd'T'HH:mm:ss.SSSZ*

Esempio: *2001-07-04T12:08:56.235-0700*

Si utilizzerà questo per rispettare lo standard ISO 8601 con adattamento Java secondo RFC 822.

### Referenze

<https://en.wikipedia.org/wiki/ISO_8601>

<https://www.ietf.org/rfc/rfc0822.txt>

## Librerie

Come linguaggio principale verrà usato Java versione 17.

Verranno utilizzate librerie open source o con licenze free.

Come spiegato in questo articolo, l’utilizzo delle librerie non è solo fortemente consigliato, ma anche una buona regola di programmazione al fine di ottimizzare, migliorare, rendere più sicuro e diminuire i costi del software sviluppato.

<https://www.cleveroad.com/blog/check-out-the-benefits-of-open-source-library-and-open-source-software/>

### Gestore librerie

Come gestione di librerie verrà usato lo strumento Maven. Questo, oltre a offrire una gestione centralizzata e automatizzata delle librerie, offre molte funzionalità come compilare e generare l’eseguibile ed eseguire gli unit tests.

<https://maven.apache.org/>

## Repository del codice

Il codice è depositato come open source su GitHub®.

<https://github.com/Cartella-Sanitaria-Project>

## Licenza

Il progetto viene esposto con licenza GNU GPL 3.0

<https://www.gnu.org/licenses/gpl-3.0.html>

## Logging e Metriche

Non verranno inseriti metriche di utilizzo, logging e altri software di troubleshooting in quanto non richiesto dal progetto e dato che il software è indicato all’utilizzo di un singolo utente tramite console di comando. Eventuali errori o segnalazioni saranno riportate direttamente in console.

# Impostazione preliminare del progetto

## Creazione classe di main

La classe di avvio si chiamerà Main, avrà un metodo main statico e sarà l’avvio dell’applicazione.

Questa classe avrà il compito di inizializzare la sessione, caricare dati da file locale e avviare la prima richiesta all’utente, ovvero un menù contestuale per richiedere la funzionalità da eseguire.

## Creazione della classe di sessione

La classe di sessione si chiamerà ApplicationSession, sarà una classe con costruttore privato e quindi accessibile alla classe stessa.

La classe conterrà un'unica istanza statica di sé stessa creata una sola volta e ogni interazione dalle classi esterne avverrà caricando la classe tramite metodo statico esposto.

Questo tipo di soluzione è definita Singleton:

<https://www.oracle.com/technical-resources/articles/java/singleton.html>

La sessione conterrà le informazioni dell’utente che serviranno anche per aggiornare il file.

## Creazione della classe di menù

La classe di menù si chiamerà Menu, sarà una classe con un metodo pubblico non statico senza argomenti e void. Questo stamperà in console le varie funzioni disponibili con un numero associato e richiederà in input all’utente il numero della funzione.

La richiesta di input deve essere contenuta in un ciclo while che finirà quando l’utente inserisce un numero corretto della lista.

Una volta ricevuto il numero, la classe di menù chiamerà una classe esterna in base alla funzionalità scelta.

In più nella lista deve essere aggiunto un campo “0 – Termina applicazione” per consentire di terminare il software. La chiusura dell’applicazione avverrà tramite semplice “System.exit(0)”.

# Requisito A - Memorizzazione e gestione delle informazioni personali dell’utente

## Dati di persistenza

Tabella 1:

|  |  |  |
| --- | --- | --- |
| Nome campo | Tipologia campo | Descrizione |
| nome | Stringa | Nome del paziente |
| cognome | Stringa | Cognome del paziente |
| indirizzo | Stringa | Indirizzo del paziente |
| email | Stringa | Email del paziente |
| telefoni | Array di Stringhe | Uno o più numeri di telefono |
| dataNascita | Stringa | Data in formato descritto nel punto 1.3 |
| luogoNascita | Stringa | Luogo di nascita del paziente |
| genere | Stringa | Maschio o Femmina |
| codiceFiscale | Stringa | Composta: 6 caratteri [A-Z], 2 cifre [0-9], 1 carattere [A-Z], 2 cifre [0-9], 1 carattere [A-Z], 3 cifre [0-9], 1 carattere [A-Z] |
| codiceSanitario | Stringa | Codice univoco, dettagli al punto 3.3 |
| gruppoSanguigno | Object | Ref. GruppoSanguigno |

Tabella 2:

|  |  |  |
| --- | --- | --- |
| Nome campo | Tipologia campo | Descrizione |
| fattoreRh | Stringa | Positivo o Negativo |
| gruppo | Stringa | 0, A, B, AB |

## Classi di persistenza

È necessario creare le seguenti classi di persistenza per gestire i dati, le classi implementeranno l’interfaccia Serializable per essere serializzate come JSON.

### Classe del paziente

La classe si chiamerà Paziente e conterrà le informazioni della tabella 1, le variabili saranno private ed esposte tramite getter e setter.

### Classe del gruppo sanguigno

La classe si chiamerà GruppoSanguigno e conterrà le informazioni della tabella 1, le variabili saranno private ed esposte tramite getter e setter.

## Generazione codice sanitario

La creazione avverrà tramite UUID senza dash (-), questo genererà un codice univoco e casuale.

Questo compito sarà affidato ad una classe di utils che userà il metodo *UUID.randomUUID()* e farà il replace dei trattini tramite *.replace("-", "")*.

<https://it.wikipedia.org/wiki/Universally_unique_identifier>

## Aggiornamento della classe di Menù

La classe Menu deve essere aggiornata aggiungendo nel messaggio della lista delle funzionalità la funzione di aggiornamento dei dati.

Inoltre, bisogna implementare la chiamata ad una nuova classe di aggiornamento dei dati in caso l’utente scelga quella voce del menù.

## Classe di aggiornamento dati

La classe si chiamerà AggiornaDati, esporrà un metodo principale che stamperà la lista dei dati con a fianco il relativo numero e chiederà all’utente di inserire il numero del campo da aggiornare, in più nella lista deve essere aggiunto un campo “0 – Uscita, torna al menù principale” per consentire di tornare al menù principale.

### Aggiornamento del dato

Il metodo principale, una volta capito il campo da aggiornare, chiamerà uno dei vari e specifici metodi della stessa classe per l’aggiornamento del campo, questo chiederà un input all’utente e controllerà la bontà del dato in base al campo scelto. In caso il valore inserito non sia corretto chiederà di nuovo l’input all’utente.

Prima della richiesta del campo deve essere stampato anche un messaggio che informa dicendo: “Per tornare al menù dei campi inserisci: 0-exit” e in caso l’input sia quello il software deve tornare al menù di aggiornamento dati.