Diario di lavoro

|  |  |
| --- | --- |
| Luogo | Trevano |
| Data | 12.02.2019 |

|  |
| --- |
| Lavori svolti |
| Oggi è stata una giornata veramente poco produttiva rispetto a quanto sarebbe dovuto essere. Il motivo è l’enorme perdita di tempo dovuta, ancora una volta, all’hosting. Dopo molto tempo usato nel cercare di collegarmi direttamente al web hosting su myd messomi a disposizione, ho chiesto al mio responsabile chiarimenti. Così, ho scoperto che il 99% degli hosting presenti su internet permette l’utilizzo del proprio spazio per i database solamente da localhost, e quindi dalla piattaforma stessa (in questo caso quindi, infomaniak). Questo si è rivelato un grande imprevisto, dato che chiaramente lavorando in Java avevamo già scartato tempo fa l’utilizzo di infomaniak. La mia richiesta dello spazio si basava dunque sull’ignoranza di questo dettaglio, mentre il mio responsabile ha semplicemente fornito quanto richiesto senza porsi o pormi domande.  Il restante tempo della mattinata è stato dunque impiegato nel provare ad implementare la versione MySQL di Heroku, ovvero il web hosting che utilizzo al momento. Purtroppo, non sono andato lontano. Proseguendo nei passaggi trovati sul sito stesso, avrei dovuto installare un build-app presente, CleanDB Heroku. Per installare una qualsiasi applicazione, per quanto gratuita come in questo caso, a quanto pare è necessario inserire la propria carta di credito. Dopo aver provato ad inserirne una falsa creata tramite un sito, mi ha segnalato errore. Probabilmente si rende conto quando questa sia falsata.  Ho quindi cominciato a ricercare nuovamente un sistema di web hosting che potesse ospitare sia un applicativo Java che un database MySQL.  Ho trovato qualcosa. Il primo è stato , il quale però richiedeva una professionalità fin troppo alta, con richiesta di documenti e iscrizioni alla propria community. Ho quindi ripiegato su uno che avevo conosciuto ai tempi ma che allora era chiuso per questione di spazi limitati: HelioHost.  Una volta iscritto e provato ad aprire il cPanel tramite il quale gestire il caricamento e l’utilizzo della mia applicazione, dà una pagina d’errore, senza che io ne riesca a capire il motivo.  Al momento penso tuttavia di aver perso fin troppo tempo, quindi nel pomeriggio proseguirò con il codice in sé, possibilmente testando e sistemando quanto creato la scorsa giornata.  Ho quindi preso in considerazione un’opzione che avrei preferito evitare: inserire la mia carta di credito per il tempo necessario all’installazione dell’applicativo gratuito di Heroku per gestire i db MySQL. Nel corso delle prossime lezioni valuterò se sarà il caso o meno di attuare la cosa.  Ho quindi continuato l’implementazione del codice, strutturando la classe di Subscription (con attributi uguali a quelli della sua tabella, fatta eccezione per l’utente che include l’oggetto e lo stesso l’evento, invece di essere scomposto rispettivamente in 3 e 2 campi).  Il Mapper di Subscription non ha niente di particolare rispetto alla logica di quello User, fatta eccezione per gli attributi indicati sopra (User e Event), i quali tramite le colonne corrispondenti viene creato l’oggetto, senza tuttavia controllarne l’effettiva presenza nel database. Questo è stato fatto dopo aver testato l’effettivo controllo e aver capito che come controllo non solo era chiaramente ridondante, ma anche inutile e dispendioso, dovendo importare i rispettivi Dao all’interno di un semplice Mapper.  Per quanto riguarda il Dao, ho deciso di non crearne uno nuovo ma di inserire i metodi all’interno dello UserDao. La scelta deriva dal fatto che molti metodi dell’utente sono necessari alla struttura dell’iscrizione e ripeterli non avrebbe avuto molto senso.  Ho quindi iniziato a creare la pagina per la visualizzazione dei dati.  L’utente viene visualizzato tramite la strutturazione dell’url:  localhost:8080/{action/{link}  Il link rappresenta l’utente sul quale si vuole eseguire l’azione, link che è chiaramente univoco e che, a differenza di quanto detto in precedenza, non verrà cancellato una volta usato per la registrazione ma verrà tenuto come valore unico dal quale ricavare gli utenti.  Action si divide in più categorie: modify è un’azione che sarà permessa solo all’utente rappresentato nel link. View è un’azione permessa solamente a consiglieri, assistenti e chiaramente amministratori, anche se questi hanno anche l’azione action, che consiste in una versione migliorata della sola lettura, potendo accettare i parametri dell’utente nel caso questi siano ancora non confermati da nessun altro amministratore.  Sulla base di queste possibilità, ho deciso di strutturare la pagina tramite due semplici variabili che rendono completamente diversa la funzione della pagina: write e checked.    Tramite questi due semplici attributi passati con il controller alla pagina tramite Spring, indico quali funzioni deve seguire. Nel caso write sia vera, tutti i campi input sono scrivibili. In caso contrario, non lo saranno, tramite l’impostazione di una variabile scritta in ogni campo input come “disabled”.  Circa allo stesso modo funziona checked. Nel caso sia vera, verranno mostrati dei campi input checkbox per permettere ad un amministratore di accettare o meno i dati inseriti. Nel caso sia false, la pagina verrà mostrata normalmente.        Ho diviso le due variabili in quanto gli assistenti e i consiglieri avranno la possibilità di lettura ma non di accettare i campi, gli utenti potranno solo scrivere e modificare i propri dati mentre gli amministratori al momento possono solo accettare i parametri, con eventualmente la possibilità in futuro di renderli capaci anche di modificare dati (pensavo più a eventuali errori di battitura e simili, per il momento non chiudo la possibilità ma non la tengo abilitata). |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| Ho avuto problemi per tutta la mattinata, potremmo dire.  Avendo ricevuto uno spazio esterno per inserire il mio database, il quale è sempre rimasto in locale, ho cominciato ad apportare le modifiche, le quali erano veramente semplici, a detta mia. Pensavo fosse sufficiente risolvere il collegamento tramite JDBC, in questo modo:      La prima versione è quella che utilizzavo fino ad ora, mentre la seconda è la versione corretta per il collegamento. O così credevo. Infatti, collegandomi tramite quel DataSource il risultato è il seguente:  “org.springframework.security.authentication.InternalAuthenticationServiceException: Failed to obtain JDBC Connection; nested exception is com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link failure”  Per verificare il funzionamento del collegamento del tipo di connessione tramite quei parametri, ho provato a collegarmi tramite l’utilizzo di HeidiSQL, il quale però mi risultava questo errore:    Ho quindi cominciato a cercare possibili soluzioni al problema, che sembrava collegato e mio. Ho consultato senza successo i seguenti siti:  <https://stackoverflow.com/questions/2983248/com-mysql-jdbc-exceptions-jdbc4-communicationsexception-communications-link-fai>  <https://serverfault.com/questions/89955/unable-to-connect-to-mysql-through-jdbc-connector-through-tomcat-or-externally>  <https://www.onlinetutorialspoint.com/spring-boot/spring-boot-h2-database-jdbc-example.html>  Ho quindi cominciato a cercare alternative della strutturazione, ma con scarso successo.  <https://stackoverflow.com/questions/50837960/spring-boot-connection-to-mysql-remote-database>  <https://springframework.guru/configuring-spring-boot-for-mysql/>  Alla fine, conscio del fatto che il problema potesse non essere di programmazione ma di logica, ho chiesto al mio responsabile consiglio, il quale mi ha spiegato che per utilizzare il web hosting che mi aveva fornito era obbligatorio l’uso di infomaniak. Chiaramente, questo ha cambiato le carte in tavola, rendendo non solo il lavoro svolto nella mattinata inutile, ma l’intera richiesta e conseguente apertura di uno spazio per contenere il database.  Alla fine, ho provato a cercare alternative definitive, ovvero un remote hosting che funzionasse con qualsiasi applicazione esterna. Dopo qualche tentativo tuttavia che ritornava gli stessi errori che dava infomaniak, ho capito che non ne valeva la pena.  <https://remotemysql.com/>  Ho avuto anche un piccolo problema in seguito con la programmazione riguardante il metodo split delle stringhe. Era necessario per dividere l’id dell’utente e utilizzare i 3 parametri in modo separato per inserirli nell’iscrizione, ma purtroppo non funzionava a dovere.  Questo è quello che controllavo:    Questo è quello che ritornava:    Dopo qualche test e controllo, ho capito che il problema riguardava la regular expression per cui dividevo. Facendo qualche ricerca ho confermato la mia ipotesi, vedendo che per quanto riguarda il carattere ‘.’ Va trattato come carattere speciale, e quindi scritto ‘\\.’. (sito con soluzione: <https://stackoverflow.com/questions/7935858/the-split-method-in-java-does-not-work-on-a-dot> |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| Ho perso veramente molto tempo oggi, riuscendo a recuperare solamente in parte nel pomeriggio. Tuttavia, avendo ancora due settimane di margine a fine progetto, e conscio del fatto che prima o poi dovrei recuperare le ore perse per malattia in qualche modo, sono fiducioso nel farcela. Per le prossime lezioni dovrei implementare la funzione multilingua, la quale ho già ben in chiaro la struttura e penso di risolverla velocemente. |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| Continuare la pagina dell’account, implementando la modifica dei campi e l’accettazione di un admin di questi ultimi già inseriti. |