RELAZIONE SIMONE PERFUMO PISSIR MATRICOLA 20032859

SPECIFICHE FUNZIONALI

Il progetto consiste nella realizzazione di una WEB-APP per la gestione di parcheggi.

Il progetto è formato da parcheggi che sono composti da tre dispositivi IOT (entrata, uscita, pagamento)

Inoltre, sono presenti amministratori che gestiscono i parcheggi potendo fare alcune azioni come cambiare lo stato da aperto a chiuso e viceversa, modificare, eliminare e aggiungere un parcheggio.

E poi ci sono gli utenti che possono entrare in un parcheggio ottenendo un ticket oppure pagare il ticket ed uscire se sono già dentro.

ANALISI DELLA TECNOLOGIA E APPROCCIO MQTT CON MOSQUITTO

Nel progetto ho utilizzato il protocollo che è adatto per la trasmissione dei dati prodotti dai dispositivi IoT ovvero **MQTT.**

Le caratteristiche principali di MQTT sono quelle di essere un protocollo semplice e leggero per lo scambio di messaggi, nonché di minimizzare il traffico sulle reti e richiedere poche risorse ai dispositivi per la sua gestione.

MQTT segue un paradigma di pubblicazione e sottoscrizione asincrono che è il publish and subscribe in cui gestisce i parcheggi, inoltre, MQTT prevede lo scambio di messaggi tramite un apposito Broker che si occupa di ricevere i messaggi e renderli disponibili sulla porta 1883.

Il broker, dunque, si occupa di consegnare i messaggi mqtt mandati dai tre dispositivi IOT entrata uscita e pagamento sui topic: parcheggio/id/entrata, parcheggio/id/uscita e parcheggio/id/pagamento.

Per sfruttare al massimo le caratteristiche di mqtt ovvero minimizzare il traffico sulle reti e richiedere poche risorse ai dispositivi per la sua gestione utilizzo thread per ogni volta che gli utenti utilizzano i sensori per eseguire delle azioni che poi verranno pubblicate sul publisher tramite stampe su terminale.

Inoltre, per le azioni dell'utente ho utilizzato i metodi delle API REST ovvero post get put e delete

Che lavorano sul database parcheggi creato con sqlite.

IL DB parcheggi aveva in origine le tabelle PARCHEGGI TICKET E AMMINISTRATIRI ma successivamente ho implementato il login per gli amministratori con github con il protocollo di oauth2 presente negli strumenti di sviluppo di github.

Ho utilizzato Il framework Spring Boot sia per creare il progetto con le dipendenze di oauth2 e di springweb ma anche per gli url delle pagine ovvero le funzioni dentro al package controller sono presenti i @requestmapping che aprono le pagine html dentro il package html dove sono presenti script in javascript dove si faranno le fetch per richiedere a parking manager le funzioni che prendono i dati dal database parcheggi.db tramite le query nella classe database.java.

Per ultimo ho reso sicuro il canale di comunicazione con un certificato parcheggio.p12 navigando su https://localhost:8000

Architettura del Software

il progetto è costituito da una web app, un database e un broker mqtt con iot device

iot device: Iot_entrata, Iot_uscita, Iot_pagamento che utilizzano sensori che, se cliccati, mandano messaggi sul broker. Ad esempio, se un utente entra con una macchina il messaggio sarà macchina entrata e ticket stampato

-WebApp: La webApp è una interfaccia che consente agli amministratori di gestire i parcheggi in modo intuitivo e agli utenti comuni di visualizzare lo stato dei parcheggi. Gli amministratori possono accedere alla loro pagina dedicata loggandosi con github utilizzando il protocollo oauth2 utilizzando il bottone login.

Gli utenti possono vedere i parcheggi,i posti totali ,iposti disponibili e lo stato(se aperti o chiusi)

Mentre gli amministratori possono creare,modificare o chiudere un parcheggio oltre che consultare la tabella dei parcheggi.

La webApp utilizza connessioni SSL e HTTPS sulla porta 8000 per garantire la sicurezza delle comunicazioni.

Database: parcheggi.db creato con sqlitebrowser ed è formato da 2 tabelle PARCHEGGI TICKET

PARCHEGGI:ID nomeparcheggio numPosti postidisponibili stato per eseguire le query degli amministratori (creazione modifica e e eliminazione) e per prendere i parcheggi per la consultazione sull'interfaccia

TICKET: ID stato ID_parcheggio. Per vedere se il ticket è pagato o no

Descrizione delle classi

Applicataion.java: da qui si attiva springboot e porta all'indirizzo https://8000

Package controller userAmmController security config AmministratoreController HomeController: Dopo che abbiamo attivato application.java vediamo package in cui ci sono le impostazioni di sicurezza ovvero che se vogliamo andare nella pagina amministratori bisogna prima loggarsi con github altrimenti si può solo andare alla home

AmministratoreController e HomeController richiamano utilizzando springboot @GetMapping le pagine html in cui si faranno le varie fetch che andranno nel userAmmController che richiamerà le funzioni del parking manager e che a sua volta richiamerà le funzioni sul database.

Package database:contiene database e query che svolgono operazione su di esso

Package Model:le classi nel package model contengono modelli degli oggetti utilizzati nel progetto, ossia oggetto di tipo ticket contente i suoi attributi, funzioni di set e get e i costruttori e lo stesso per percheggio . le classi Iot entrata pagamento e uscita gestiscono attraverso i thread un invio di messaggio dopo entrata uscita di una macchina o il pagamento di un ticket

manager contiene la classa parkingManager dove abbiamo il broker, reso sicuro sulla porta tcp://localhost:1883, che serve a gestire i messaggi mqtt in arrivo dalla classe sensori iot parcheggio, sui topic entrata uscita e pagamento, per poi andare a elaborare questi messaggi avvelendosi delle funzioni del database e andando a effettuare modifiche nel database. inoltre, gestisce le funzioni per l'amministratore, per creare e modificare un parcheggio richiamate dal controller amministratore, il quale gestisce le fetch della pagina web. Inoltre, abbiamo il broker, reso

sicuro sulla porta tcp://localhost:1883, che gestisce l’arrivo e i l’invio dei messaggi

sensori è package che contiente l'iot parcheggio, serve a simulare attraverso l'uso del frame aEventsense l'entrata e uscita da un parcheggio o il pagamento di un ticket, nel frame sono presenti 3 bottoni per le operazioni nel parcheggio. i bottoni vengono disabilitati se il colore è rosso, il controllo del' funzionalità dei bottoni è gestito da una funzione che controlla ad ogni operazione se vi ancora sono posti disponibili per l'entrata nel parcheggio, altrimenti se il ticket è pagato per l'uscito dal parcheggio.

Poi fuori dal package principale abbiamole resources con il package key store che contiene il cerificato pacheggio.p12 ,le pagine html e le application properties con il client id e il client secret per loggarsi su github e le informazioni sul certificato

Validazione

L’interfaccia utente si attiva sulla classe application.java che ci porta alla porta https://localhost:8000 dove abbiamo l’utente che può solo visionare i vari parcheggi poi sempre nella pagina di home si può accedere alla pagina amministratori /amministratori

Dove chi si è loggato con github può eliminare modificare e aggiungere un parcheggio e ha un bottone di logout per tornare a visionare la parte dell’utente.

Credenziali github:

20032859@studenti.uniupo.it

Password: +Simonepissir1

per poter visionare i sensori si deve utilizzare il terminale dell’ide iotParcheggio e parking manager dove in iot parcheggio chiamando AEventSense ci fa vedere i tre bottoni entrata uscita e pagamento e mentre il parking manager si mette in ascolto sui tre topic: topic/+/entrata, topic/+/uscita e topic/+/pagamento.