Relazione progetto Reti 2

# Specifiche funzionali

Per la realizzazione del progetto riguardante una palestra domotica, il nostro gruppo si è occupato del monitoraggio ed analisi dell’attività dei nuotatori. A tale scopo è stata creata un applicazione per cellulare mediante la quale è possibile iscriversi ai corsi in piscina come nuotatori o come coach. Una volta iscritti e loggati come istruttori è possibile creare un allenamento, questo sarà composto da quattro esercizi (composti da stile di nuoto e numero di vasche) e la data in cui dovrà essere svolto. Se invece si effettua il login come nuotatore potranno essere effettuate le seguenti azioni:

* Visualizzare tutti gli allenamenti che sono stati creati dai diversi coach, con data maggiore o uguale a quella odierna.
* Possibilità di iscriversi agli allenamenti in data odierna.
* Collegare un fitbit, sul quale verrà caricato l’allenamento a cui ci si intende iscrivere.
* Visualizzare tutti gli allenamenti effettuati nel passato con le relative statistiche calcolate dal fitbit.
* Analisi dei dati dell’ultimo allenamento attraverso il confronto con delle costanti.

Il progetto inoltre utilizza un “non server”, ovvero una macchina che svolge la funzione di server anche se effettivamente non lo è, qui i dati vengono salvati nel file system interno, ogni utente ha una propria cartella dedicata in cui risiedono gli allenamenti effettuati o creati nel caso in cui sia rispettivamente nuotatore o coach.

<foto file system>

Per simulare il fitbit è stato creato un processo in java, il quale genera numeri pseudocasuali al fine di simulare lo svolgersi dell'attività sportiva, alla fine della simulazione viene fatta la media dei dati dell'allenamento. Ogni qualvolta che si decide di attivare il processo fitbit viene generato un numero casuale di 5 cifre che simboleggia il numero di serie univoco, il quale dovrebbe essere presente sul retro di un eventuale braccialetto reale.

Sono presenti due sensori all'inizio e alla fine della piscina, che trasmettono un segnale ogni volta che un nuotatore termina una vasca, in questo modo vengono ridotti il numero di esercizi da fare e quando questi arrivano a zero il fitbit smette di registrare i dati.

Per lo scambio di messaggi si è scelto di utilizzare un message broker con mqtt ed ogni utente in fase di login sottoscrive un il sottoalbero sul quale vengono pubblicati i propri dati.

# Analisi della tecnologia

### IoT Hub e broker

I message broker e gli IoT Hub sono nati per fronteggiare l'esigenza di avere un sistema più flessibile rispetto ai sistemi basati sul protocollo STUN. L’applicazione IoT è infatti a basso traffico e i dati sono caratterizzati in tipologie, offrendo quindi qualcosa adatto alla situazione e che offra servizi aggiuntivi. L’IoT Hub è quindi paragonabile a un insieme di caselle postali mediante le quali è possibile interfacciarsi col server. La comunicazione è vista come una comunicazione a memoria comune con la differenza che lo scambio di dati non avviene all’interno di un sistema operativo ma in rete, il traffico viene gestito da un sistema distribuito di server poiché deve gestire grosse quantità di dati. Quando viene attivato un processo, esso si associa a una delle caselle postali e vengono avvisati tutti gli user che hanno a che hanno sottoscritto il servizio. Questi ultimi invieranno dei messaggi che verranno letti dal processo stesso, tutto questo viene implementato attraverso delle chiamate send() e receive(). Sono presenti tuttavia anche delle criticità: in presenza di un sistema avente migliaia di processi che comunicano tra loro, la (dis)assegnazione di una casella è obsoleta e difficile da realizzare. Per questo motivo sono stati introdotti i message broker: una struttura simile a una bacheca in cui tutte le novità vengono segnalate a tutti i device dell’applicazione IoT che, se interessati, possono visualizzare. Il paradigma utilizzato è di tipo publish-subscribe, il messaggio viene quindi pubblicato in un topic della bacheca e tutti quelli registrati a esso riceveranno una notifica. A differenza dell’IoT Hub (molto simile alla comunicazione tra processi), il message broker è più simile alla memoria condivisa, *quindi vengono gestiti tutti gli host allo stesso modo con dovute precauzioni*, esso è preferito al primo perché la gestione di milioni di utenti durante lo scambio di messaggi è più gestibile (anche se più naturale), tuttavia non è adatto allo streaming *dato che la sua velocità non lo permette*.

### Principali classi di applicazione:

* Smart Home: gestione della casa a livello di luci, elettrodomestici e allarme, rendendola intelligente;
* Dispositivi indossabili;
* Connected cars: vetture e trasporti connessi;
* Industrial Internet: monitoraggio e controllo degli apparati industriali;
* Smart Cities: come smart home ma a livello cittadino;
* Agriculture: monitoraggio e controllo delle colture;
* Smart Retail: supporto alla vendita al dettaglio;
* Energy Engagement: gestione della rete di distribuzione;
* Healthcare: monitoraggio e cura dei pazienti distribuiti nel territorio;
* Allevamento di animali.

### Implementazione di un broker

I message broker vengono implementati utilizzando MQTT (Message Broker Telemetry Transport) (Message Broker Telemetry Transport), un protocollo proposto a IBM tutt’oggi adottato in molte piattaforme tra cui Azure, la sua natura leggera permette infatti di orientarsi facilmente ai message broker, basandosi sul modello publish-subscribe: i processi comunicano tra loro tramite il loro argomento senza esporre loro stessi direttamente, infatti la comunicazione non è diretta ma avviene modificando dati comuni organizzati in una certa gerarchia.

### Funzionamento

L’architettura di un broker è ad albero le cui foglie rappresentano i dati, essi sono descritti dai nodi ai livelli superiori, *permettendo così di prenderli gestendo l’intero albero oppure dei sottoalberi*. I processi client possono essere:

* Publisher: dopo la connessione al broker, comunicano con un thread locale a esso che gestisce la memoria condivisa, tramite esso è possibile pubblicare i dati;
* Subscriber: alla loro connessione al broker (mantenuta dal subscriber stesso attraverso un token) allocano un thread e, attraverso quest’ultimo, si colloquia con un thread consumatore locale al broker, il suo compito è quello di recapitare i dati richiesti.

La comunicazione col broker avviene utilizzando i socket, i processi client attendono su essi fino alla fine dell’operazione, la connessione viene effettuata utilizzando TCP dal momento che UDP non è reliable.

#### Formato del messaggio

Il messaggio utilizzato per la comunicazione tramite message broker è binario, l’unica parte non binaria è quella dei nomi e dei pathname, i quali utilizzano lo standard UTF-8, *permettendo così messaggi di dimensioni ridotte*. Un messaggio è composto dai seguenti campi:

* Header di controllo: permette di capire il tipo di messaggio inviato, in tutto sono 16;
* Lunghezza;
* Variabili della lunghezza: blocco di dati variabile che dipende dal tipo di messaggio;
* Payload: dati effettivi.

Tra i tipi di messaggi vi sono pingReq e pingResp, necessari per far circolare il token e quindi mantenere viva la connessione.

#### Qualità del servizio

La qualità del servizio del broker riguarda il recapito dei messaggi e non la velocità, ve ne sono di tre tipi:

* At Most One (Best-effort): fa un tentativo per consegnare il messaggio al broker, se funziona bene, altrimenti si accontenta. In questo tipo di QoS l’ACK non viene mandato *dato che non si preoccupa della ricezione dei dati;*
* At Least One: Il messaggio deve essere ricevuto almeno una volta, tutte le altre non alterano il comportamento per via dell’idempotenza. La conferma di ricezione avviene attraverso un ACK, se questo viene perduto, il mittente attende un intervallo e dopo ritrasmette il messaggio;
* Exactly One: il messaggio deve essere ricevuto una e una sola volta. Questo meccanismo è il più complesso, infatti per ricevere un messaggio una sola volta si procede nel seguente modo: viene inviato un messaggio al broker e quest’ultimo risponde con un PUBREC, se quest’ultimo non viene ricevuto, si ritrasmette il messaggio con un flag DUP. Vedendo che c’è il flag DUP, il broker scarta il messaggio e invia un PUBREC, quando ricevuto il mittente invia un PUBREL al broker e quest’ultimo risponderà con un PUBCOMP. Si fa quindi una sorta di three way handshake in cui *è possibile capire dove è possibile riprendere da dove ci si è fermati in caso di interruzioni*.

Il tipo di qualità del servizio viene scelto alla registrazione e ce n’è uno per ogni topic. Il problema sorge quando publisher e thread dispongono di QoS differenti, bisogna quindi fare in modo che la qualità del servizio del thread sia maggiore o uguale a quella di chi si è registrato *per evitare perdite di pacchetti durante la trasmissione*.

# Scelta dell’approccio

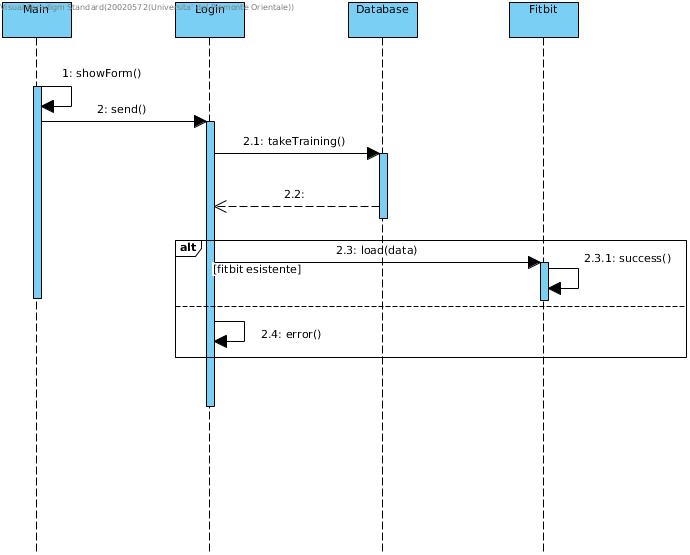
All'interno del progetto sono scelte le seguenti tecniche per l'implementazione delle varie componenti.

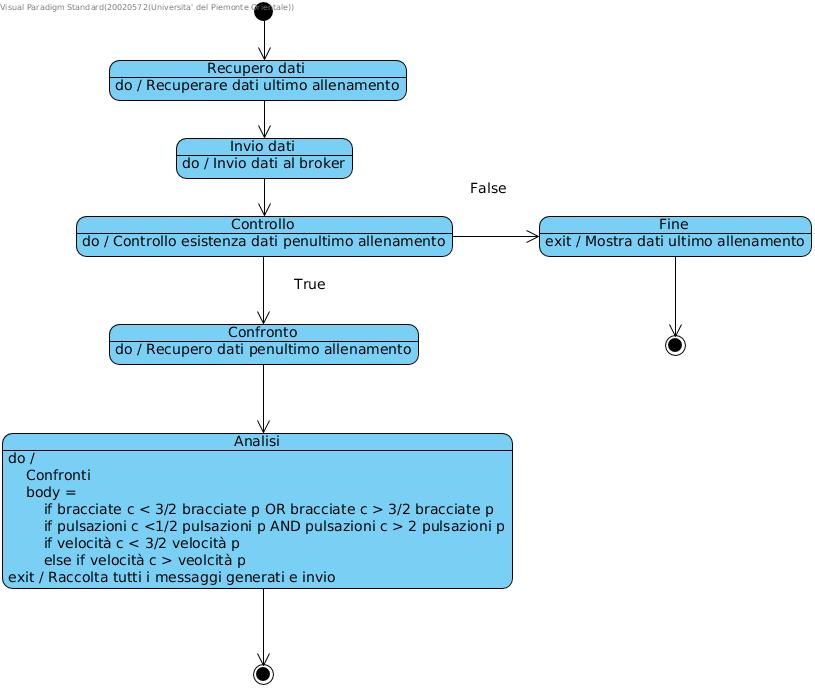
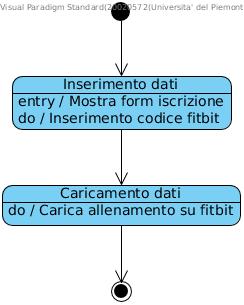
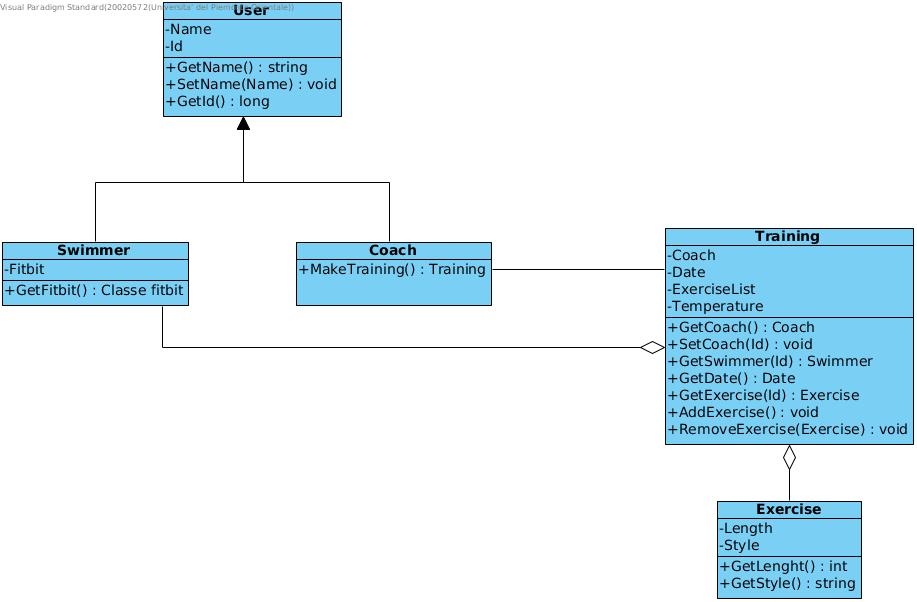
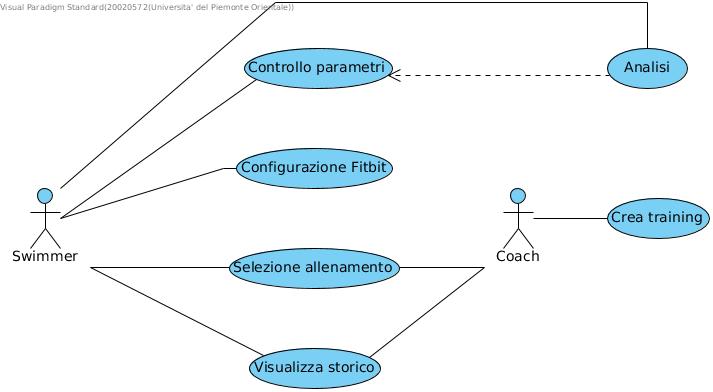
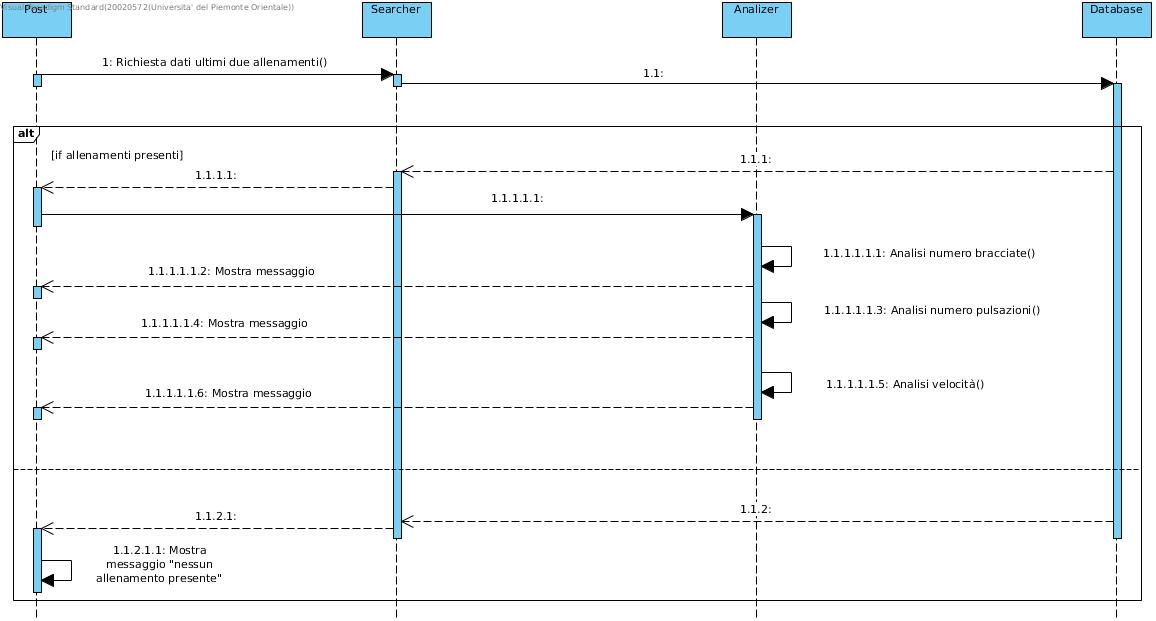
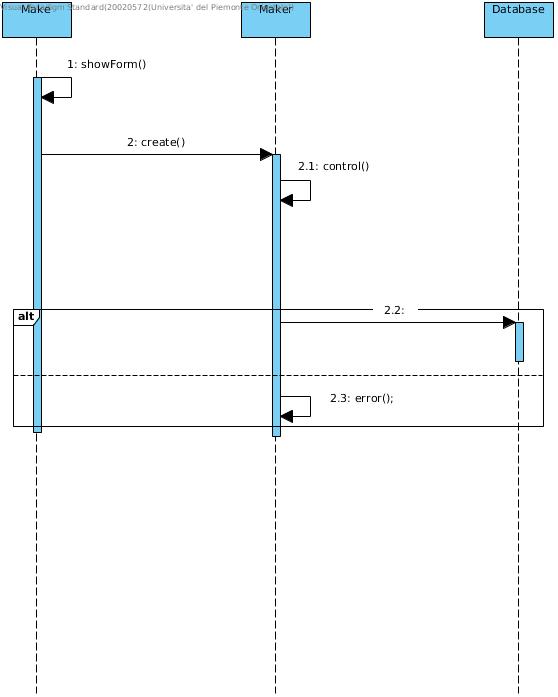
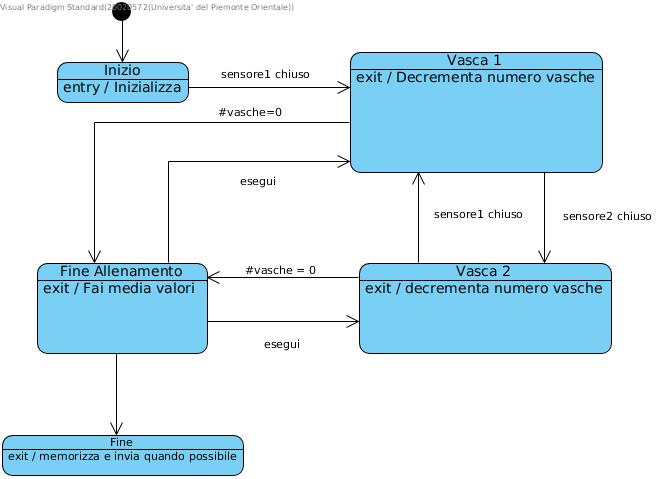
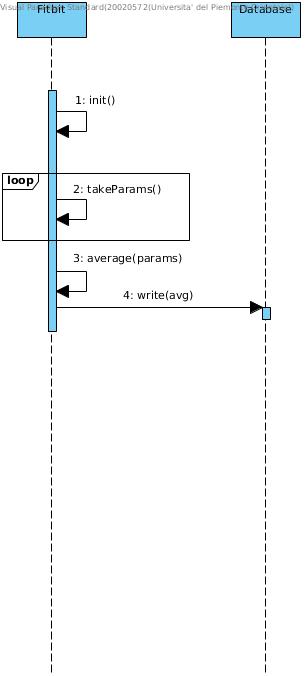
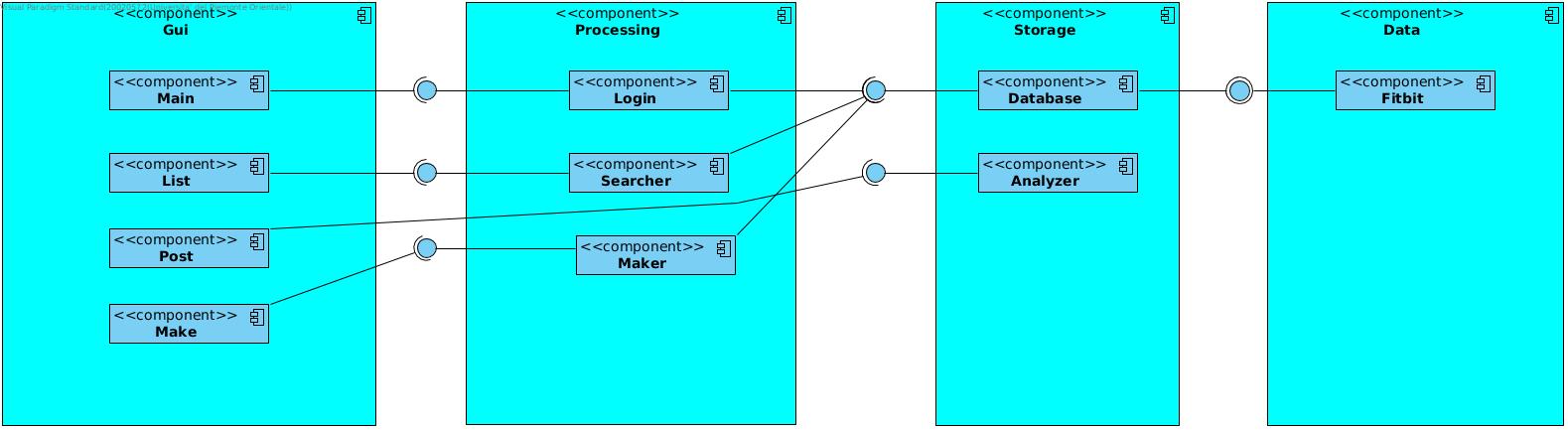
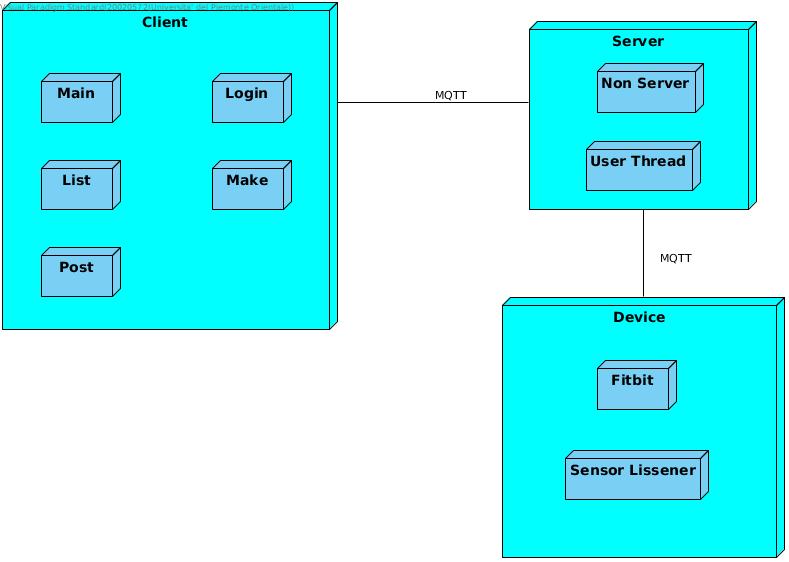
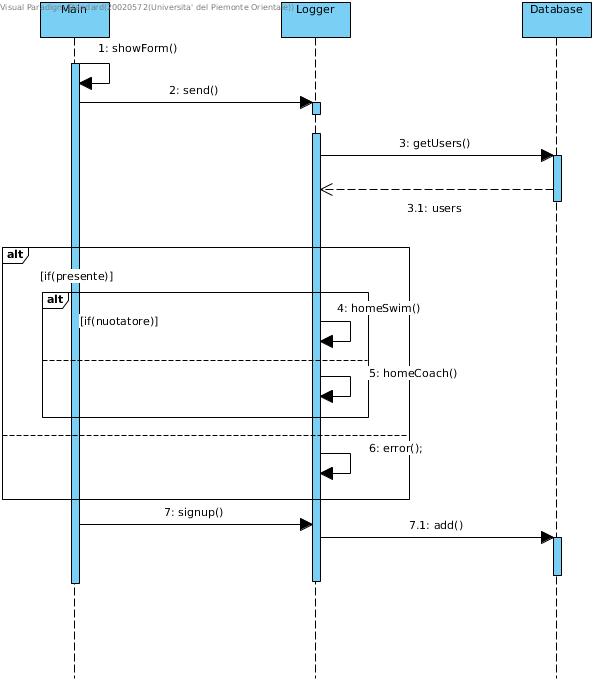
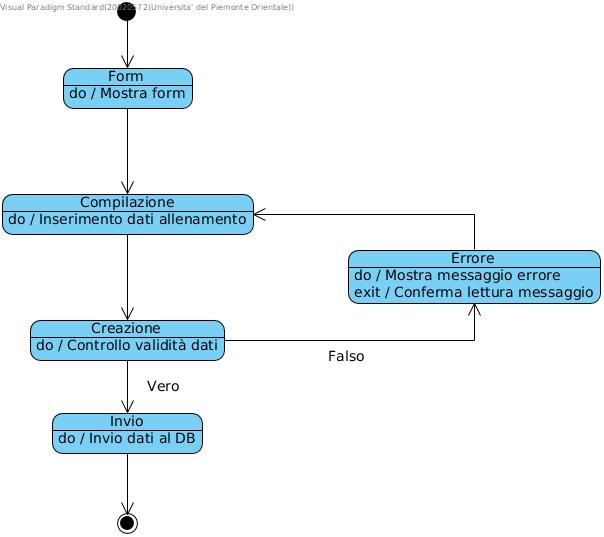
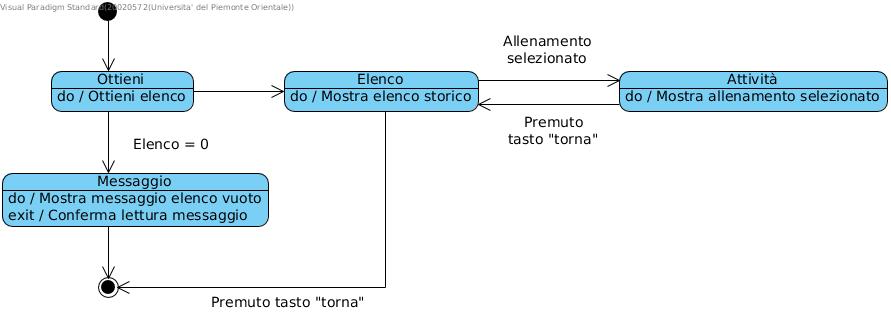
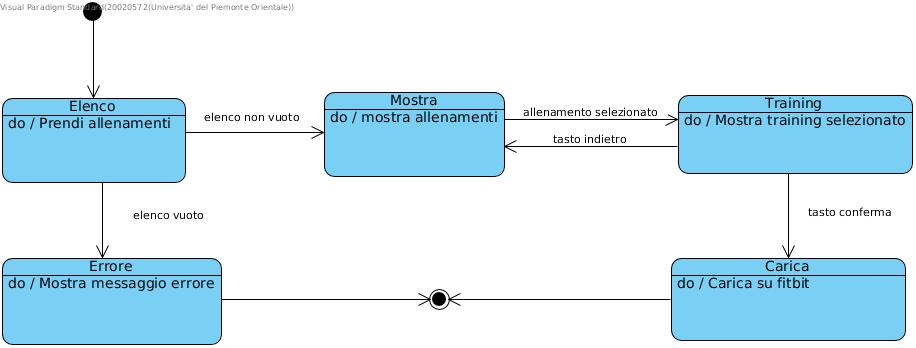
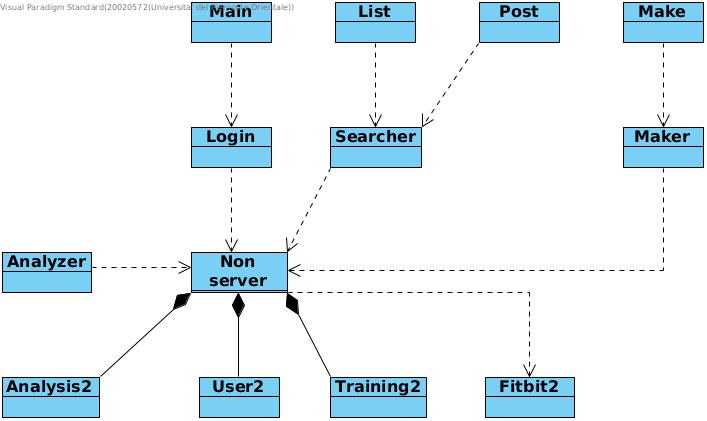
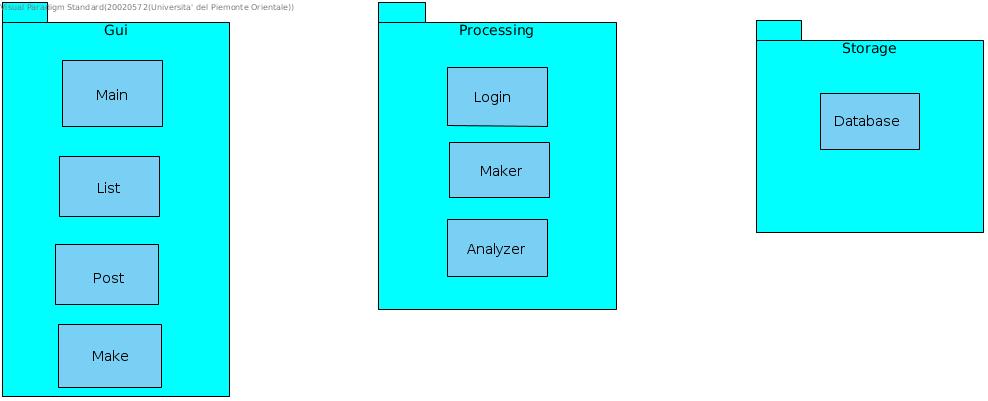
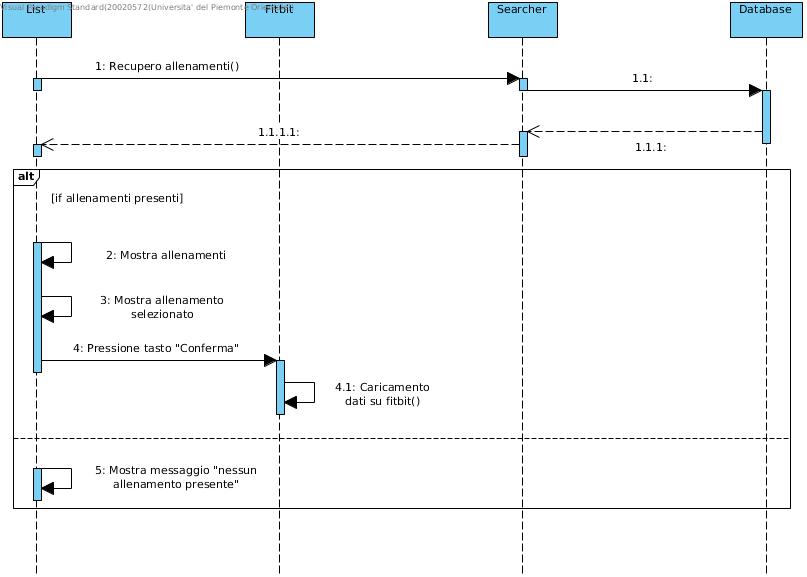
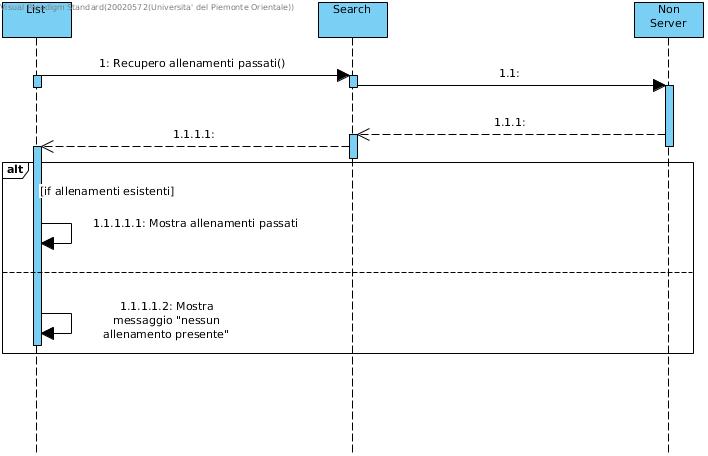
Mosquitto per lo scambio di messaggi, questa scelta è stata espressa dal committente (il professore). Mosquitto è un protocollo leggero del progetto Eclipse e sponsorizzato da IBM fornendo apposite librerie, l’organizzazione dei subject è ad albero e con esso è possibile fare Bridging, ovvero che si possono creare reti di broker con infrastruttura locale e nel cloud. I Subject possono essere cercati attraverso + e #, il primo indica un nome di un qualunque subject mentre il secondo indica un intero sottoalbero.

Json per il formato dei messaggi scambiati, questa scelta è stata fatta poiché gli sviluppatori hanno imparato il linguaggio durante il periodo di stage. JSON (JavaScript Object Notation) è un semplice formato per lo scambio di dati ed è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni comuni ai più usati linguaggi di programmazione. JSON è basato su un insieme di coppie nome/valore come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.

Per l'interfaccia utente si è deciso di utilizzare un applicazione che gira su android anziché implementare un web server al fine di valorizzare le conoscenze apprese durante il periodo di studio e stage.

# Architettura del Software





# Validazione

Quando l’app sul cellulare viene lanciata, sono possibili due operazioni: login e signup. Per poter accedere è necessario inserire email e password negli appositi campi. I dati così inseriti saranno confrontati con quelli presenti nel file degli utenti con un confronto tra stringhe, se non dovesse esserci alcuna corrispondenza verrà restituito un messaggio di errore che verrà mostrato a schermo sul cellulare. Nel caso ci si volesse registrare come nuovo utente, bisognerà compilare l’apposito form, se l’el’email usata per la nuova iscrizione dovesse essere già presente nel file utenti apparirà un messaggio di errore che chiederà all’utente di inserirne una nuova.a.

Loggandosi con credenziali da coach si ha la possibilità di creare un nuovo allenamento che i nuotatori potranno scegliere di eseguire, poiché ogni allenamento viene salvato come un file che utilizza la data come nome, per evitare conflitti e sovrascrizioni prima della creazione viene effettuato un controllo per accertare che non esista già un allenamento in tale data, nel caso si già presente non viene creato e appare un messaggio di errore a schermo.

Loggandosi con credenziali da nuotatore è possibile vedere tutti gli allenamenti creati dai vari coach che hanno data maggiore o uguale a quella odierna. Tale data viene reperita dal sistema operativo e confrontata con quella presente in ogni allenamento mediante un ciclo che seleziona tutti i file, se il confronto da esito positivo viene visualizzato a schermo il training altrimenti si passa al successivo.

Per poter caricare sul fitbit un allenamento è necessario che sia uno in data odierna, usando la logica spiegata prima se viene selezionato un training valido, verrà visualizzato un bottone che permetterà l’associazione con il bracciale.