Università degli Studi di Salerno

Corso di Ingegneria del Software

Time-Platform
System Design Document
Versione 2.0

Data: 30/11/2015

Progetto: Time-Platform	**Versione** : 2.0
Documento: System Design Document	**Data** : 03/12/2015

Partecipanti:

Nome	Matricola
Angellotti Fabio [AF]	0512103314
Angellotti Francesco [AFR]	0512101988
Milito Mariano [MM]	0512101430

Scritto da:	Angellotti Fabio

Revision History

Data	Versione	Descrizione	Autore
21/11/2015	1.0	Creazione documento e bozza *introduzione*	[AF]
22/11/2015	1.1	Revisione *Introduzione*	[AFR],[MM]
23/11/2015	1.2	Bozza *architettura software proposta*	[AF],[AFR], [MM]
24/11/2015	1.3	Revisione paragrafo *architettura software proposta*	[AF],[AFR], [MM]
25/11/2015	1.4	Servizi dei sottosistemi	[AFR]
01/12/2015	1.5	Revisione del documento	[AF],[AFR], [MM]
02/12/2015	2.0	Revisione del documento	[AF],[AFR], [MM]
03/12/2015	2.0	Revisione diagrammi	[AF],[AFR], [MM]

Indice

1. Introduzione

1.1 *Purpose of the System*

Il sistema che l'azienda ha richiesto è una piattaforma web-based per la gestione dei turni di lavoro dei propri dipendenti, che in particolare permetta ad ogni utente del servizio di accedere al sistema,previa autenticazione,e gli permetta di controllare i turni e gli orari lavorativi.Così facendo si ottiene la possibilità di interagire instantanemente con l'amministrazione affinchè vi sia una corretta e veloce gestione dei turni onde evitare perdite di tempo e probabili gap amministrativi.

1.2 *Design Goals*

Gli obiettivi di qualità che il sistema dovrà rispettare sono:

- **Prestazioni**
    - **Tempo di risposta:** Ci si aspetta che una richiesta possa essere soddisfatta in meno di 2 secondi. Oltre questo limite il sistema è da considerarsi scarso.
    - **Throughput:**. Questo criterio è trascurabile, perché l'applicazione effettua poco lavoro computazionale.

- **Criteri di Affidabilità**
    - **Robustezza:** il sistema software dovrà evitare malfunzionamenti a seguito dei difetti nel codice. La maggior parte dei problemi riguardanti il codice dovrà essere individuata nella fase di testing. Inoltre il sistema dovrà sopravvivere agli input non validi immessi dall'utente.
    - **Affidabilità:** L'affidabilità è definita come la probabilità che il sistema funzioni senza errori per un dato intervallo di tempo, in un dato ambiente e per un determinato scopo. Useremo il numero di difetti (bug) che verranno scoperti durante la fase di test per predire l'affidabilità e sarà migliorata rimuovendo i fault (comportamento run-time inaspettato (ed errato) osservato da un utente del sistema) che compaiono nelle parti del sistema che sono più frequentemente usate.
    - **Disponibilità:** Il sistema sarà sempre disponibile a compiere nuove operazioni,in quanto il costo computazionale delle operazioni è molto basso.
    - **Tolleranza ai guasti:** Capacità di operare sotto condizioni di errore , affinché il sistema funzioni correttamente sotto tali condizioni viene utilizzata la tecnica della ridondanza. Questa tecnica si basa sulla presenza degli stessi dati su diversi server , così che se uno di questi risulti non raggiungibile viene sostituito.
    - **Sicurezza del sistema:** Capacità di resistere ad attacchi da parte di malintenzionati. La possibilità di resistere ad attacchi da parte di malintenzionati sarà garantita da un apposito sistema di gestione degli

account. Inoltre un'ulteriore sicurezza dei dati è garantita anche dal sistema DBMS. Per motivi legati alla sicurezza ,l'area utente per la gestione dei dipendenti è accessibile solo in locale.

- **Usabilità**: Time-Platform rende ogni funzione di semplice uso, garantendo all'utente un'ottima esperienza di utilizzo del sistema; grazie ad interfacce grafiche intuitive.

- **Compromessi di design**
    - **Spazio vs. velocità:** Se il software non rispetta i requisiti di tempo di risposta, è possibile utilizzare più memoria per velocizzare il sistema.
    - **Tempo di rilascio vs. funzionalità**: Se i tempi di rilascio sono stringenti, possono essere rilasciate meno funzionalità di quelle richieste, ma nei tempi giusti.
    - **Tempo di rilascio vs. qualità**: Se i tempi di rilascio sono stretti, il project manager può decidere di rilasciare il software nei tempi prefissati ma con dei bug noti e, successivamente, di correggere gli errori.

1.3 *Definizioni, acronimi e abbreviazioni*

- **DBMS:** Database managment system;
- **Fault:** Comportamento run-time inaspettato ed errato osservato da un utente del sistema;
- **Web-Based :** Web-based ovvero "basato sul web".
  Quando si parla di un software Web-based si intende un programma in cui tutte le funzioni sono accessibili con un normale web-browser come Explorer ,quindi non necessita di alcun software di installazione sul computer degli utenti. Le applicazioni Web-based permettono ,quindi, all'utente di interagire con il sistema tramite un web-browser. Tale caratteristica permette all'utente di interagire da qualsiasi sede e di rendere quindi una postazione generica una vera e propria postazione di lavoro.

1.5 *Overview*

Il sistema che si vuole realizzare è un sistema distribuito nel quale esiste un nodo Server contente l'archivio dati a cui i vari attori, dalle loro postazioni, vogliono accedere. Pertanto esistono dei moduli client che richiedono dati ad un modulo server. Fra loro i vari moduli non comunicano, l'unica comunicazione è rivolta da e verso il server:

- Naturalmente tali moduli risiedono in macchine differenti;
- L'architettura scelta è pertanto quella client-server.

L'architettura su un singolo host è di tipo three-tier, molto adatta per sistemi dove c'è la necessità di un'organizzazione stratificata che delega ogni livello a un ristretto campo d'azione. In questo caso utilizzata per avere una netta distinzione tra interfaccia grafica e logica applicativa.
Molto lavoro è stato fatto per rendere l'applicazione flessibile tramite l'utilizzo delle interfacce.
Inoltre un'analisi dettagliata degli obiettivi di design è stata stilata per focalizzare il vero punto d'arrivo di questa attività fondamentale.

2. Sistema corrente

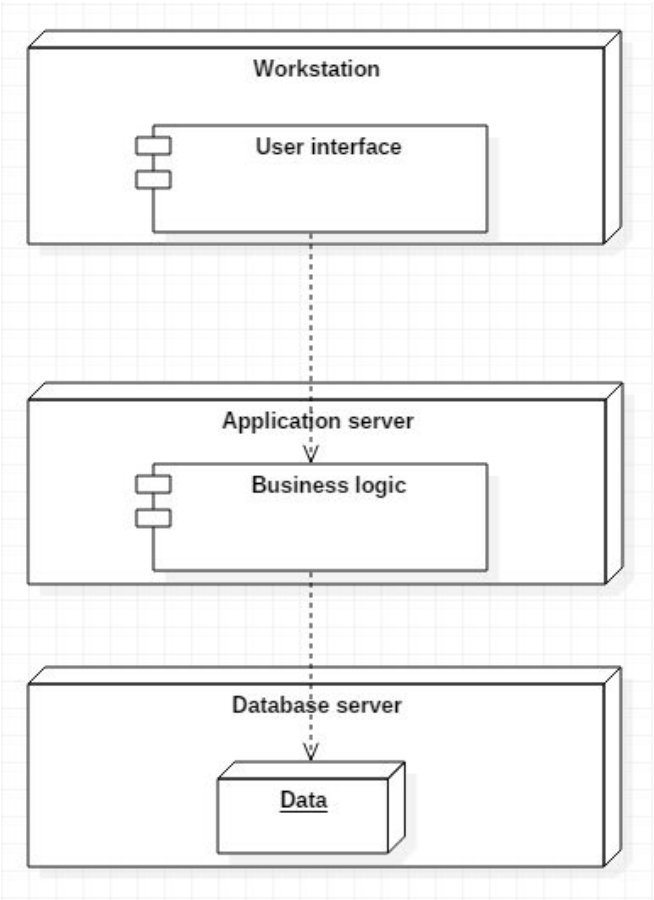
Non vi sono sistemi precedenti.

3. Architettura software proposta

3.1 *Overview*

Nella decomposizione del sistema si è scelto uno stile architetturale Three-tier.
Tale architettura organizza i sottosistemi in tre layers:

- **Il layer di interfaccia**: è lo strato che implementa la GUI del sistema, dove vi sono tutti quegli elementi che permettono all'utente di interagire con il sistema (form , button, text box, etc.)
- **Il layer di buisness logic**: racchiude tutta la logica applicativa.
- **Il layer di storage**: i dati acceduti dalla buisness logic sono presenti in maniera persistente in un DBMS.



3.2 *Identificazione dei sottosistemi*

Per prima cosa dobbiamo evidenziare l'interfaccia del nostro sistema che è caratterizzata da tre componenti. In primis troviamo l'interfaccia del "*login*" ,comune sia per l'autista che per l'amministratore, che a sua volta indirizza i rispettivi nelle proprie pagine personali ovvero "*Personal page driver*" e "*Personal page admin*".
In primo luogo abbiamo distinto tre parti principali del sistema TIME-PLATFORM: la gestione dell'autenticazione, la gestione degli autisti e la parte di amministrazione.
Un primo sottosistema è la gestione dell'autenticazione degli utenti.

La gestione degli autisti(sottosistema) si suddivide in sette funzionalità:

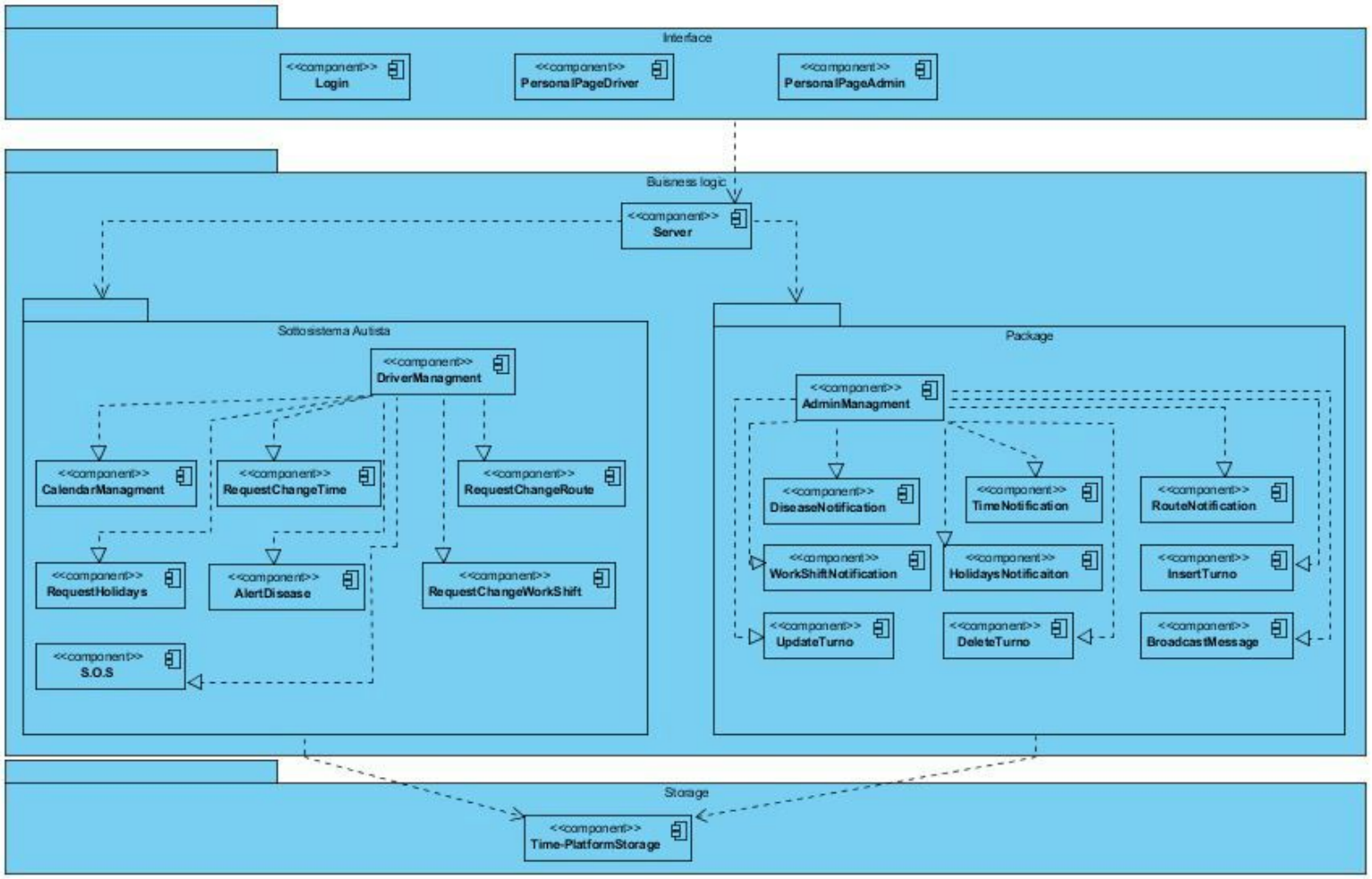
- **Calendar Managment** : Si occupa della creazione del calendario lavorativo.
- **Request change time**: Si occupa di inviare la richiesta di cambio orario.
- **Request change route**: Si occupa di inviare la richiesta di cambio linea.
- **Request change workshift** : Si occupa di inviare la richiesta di cambio turno.
- **Request holidays** : Si occupa di inviare la richiesta di ferie.
- **Alert disease** : Si occupa di inviare un avviso di messa in malattia.
- **SOS**: Si occupa di inviare una richiesta di soccorso.

La gestione amministrativa (sottosistema) si suddivide in cinque funzionalità:

- **Disease notifications**: Si occupa di elencare tutte le notifiche di malattia .
- **Time notifications**: Si occupa di elencare tutte richieste di cambio orario.
- **Route notifications**: Si occupa di elencare tutte le richieste di cambio linea.
- **Workshift notifications**: Si occupa di elencare tutte le richieste di cambio turno.
- **Holidays notifications**: Si occupa di elencare tutte le richieste di ferie.
- **Inserimento turno di lavoro** : Permette di inserire un nuovo turno di lavoro.
- **Modifica turno di lavoro** : Permette di modificare un turno di lavoro.
- **Eliminazione turno di lavoro** : Permette di eliminare un turno di lavoro.

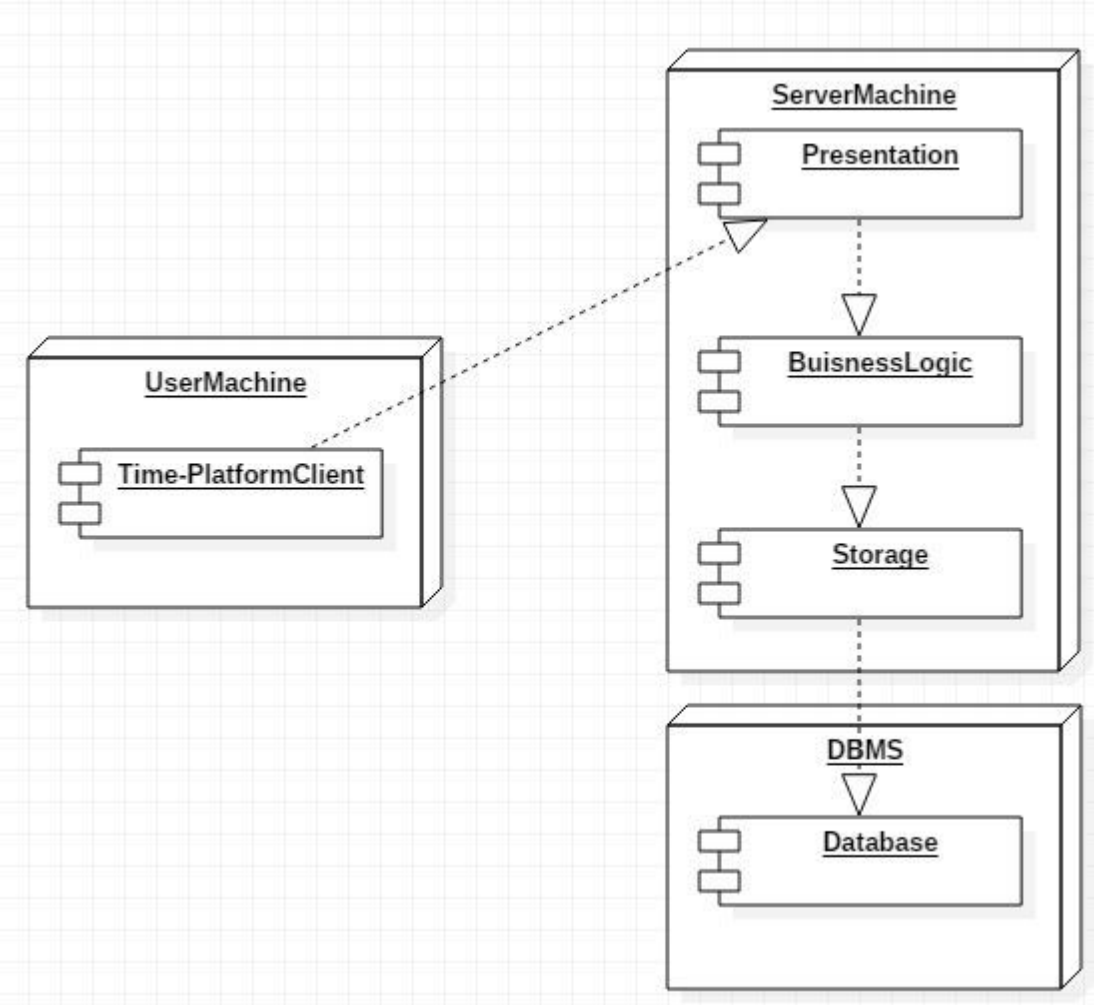
Infine come ultimo sottosistema abbiamo Time-PlatformStorage che si occupa della memorizzazione di dati in maniera persistente, ad esempio :

- Turni di lavoro degli autisti (data ,ora, linea).
- Informazioni personali degli autisti.
- Richieste di ferie .
- Richieste di eventuali cambi.



3.3 *Hardware/Software mapping*

Il sistema ha principalmente come prima vista quella dell'autenticazione. Ogni utente avrà la possibilità, mediante un dispositivo personale, di connettersi al sistema tramite una form di login. Come abbiamo precedentemente accennato il nostro sistema si basa su una architettura a tre livelli; il primo di questi è il livello di presentazione che è stato creato tramite HTML e jQuery, quest'ultimo è una libreria JavaScript che ci ha permesso di manipolare e gestire eventi ed animazioni degli elementi del DOM in maniera molto semplice.
Time-Platform è un sistema multipiattaforma ovvero accessibile da qualsiasi web-browser, poiché tutto il codice è presente su un web-server che lo esegue. Di conseguenza Time-PlatformClient è un browser web e Time-PlatformServer crea e gestisce le pagine web. I linguaggi di programmazione utilizzati sono : PHP per la gestione del login, la gestione delle sessioni dei vari utenti e per effettuare query al database ; jQuery per la gestione del calendario (plugin scaricato dal sito fullcalendar.io) e per la grafica; JavaScript per prelevare i dati dalle form compilate dagli utenti.
Il webserver utilizzato è Apache 2.4.17 .
Rivolgiamo ora la nostra attenzione alla gestione della persistenza.



3.4 *Gestione dei dati persistenti*

Dopo un'analisi accurata siamo giunti alla conclusione che è necessario utilizzare un database relazionale ai fini di tener traccia della grande quantità di dati da gestire.
E' stato scelto un DBMS MySQL 5.6.19 , che ci permette di abbattere i costi e i tempi, poichè non si deve progettare ed implementare tutta la logica per rendere persistenti i dati.
La connessione al database viene effettuata tramite il linguaggio php con il driver nativo MySQL.
Per effettuare delle query sul database si devono eseguire le seguenti istruzioni:

- mysql_connect()
- mysql_select_db()
- mysql_query()
- mysql_fetch_assoc()

L'istruzione mysql_connect() necessità dell'indirizzo(IP) del server e del nome utente e password; come risultato restituisce un numero che costituisce l'identificativo di connessione MySQL ("MySQL link identifier") in caso di successo oppure FALSE in caso di fallimento.
Dopo aver effettuato la connessione a MySQL si deve selezionare il database sulla quale si deve operare, tramite l'istruzione mysql_select_db() , che restituisce TRUE o FALSE come esito della selezione.

Per effettuare una query al database si utilizza la funzione mysql_query() che prende come parametro una query sql e la connessione effettuata con mysql_connect() , dopodichè si recuperano i risultati con mysql_fetch_assoc() se tale funziona ha restituito TRUE .
Infine si deve terminare la connessione con il database con l'istruzione mysql_close(), passando in input il *MySQL link identifier* ottenuto da mysql_connect().

3.4.1 *Identificazione degli oggetti persistenti*

I dati persistenti sono quei dati che devono sopravvivere alla singola esecuzione del sistema. In particolare i dati identificati come persistenti sono:

- **Autista**: Questo oggetto memorizza i dati di tutti gli autisti presenti in azienda.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	Matricola	int(4)			No	*Nessuno*
2	Password	varchar(30)	latin1_swedish_ci		No	*Nessuno*
3	Nome	varchar(30)	latin1_swedish_ci		No	*Nessuno*
4	Cognome	varchar(35)	latin1_swedish_ci		No	*Nessuno*
5	Cf	varchar(16)	latin1_swedish_ci		No	*Nessuno*
6	Cellulare	int(10)			No	*Nessuno*

- **Amministratore**: Questo oggetto memorizza i dati di tutti gli amministratori presenti in azienda.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	Matricola	int(4)			No	*Nessuno*
2	Password	varchar(30)	latin1_swedish_ci		No	*Nessuno*
3	Nome	varchar(30)	latin1_swedish_ci		No	*Nessuno*
4	Cognome	varchar(35)	latin1_swedish_ci		No	*Nessuno*
5	Cf	varchar(16)	latin1_swedish_ci		No	*Nessuno*
6	Tel	int(10)			No	*Nessuno*

- **Capo fabbrica**: Questo oggetto memorizza i dati di tutti i capo fabbrica presenti in azienda.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	Matricola	int(4)			No	*Nessuno*
2	Password	varchar(30)	latin1_swedish_ci		No	*Nessuno*
3	Nome	varchar(30)	latin1_swedish_ci		No	*Nessuno*
4	Cognome	varchar(35)	latin1_swedish_ci		No	*Nessuno*
5	Cf	varchar(16)	latin1_swedish_ci		No	*Nessuno*
6	Tel	int(10)			No	*Nessuno*

- **Richiesta cambio orario**: Questo oggetto memorizza tutte le richieste di cambio orario per uno specifico turno, inviate dagli autisti.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	id	int(5)			No	*Nessuno*
2	descrizione	varchar(100)	latin1_swedish_ci		Sì	*NULL*
3	oraInizio	time(6)			Sì	*NULL*
4	oraFine	time(6)			Sì	*NULL*
5	matricolaAut	int(4)			No	*Nessuno*
6	idturno	int(4)			No	*Nessuno*

- **Richiesta cambio linea**: Questo oggetto memorizza tutte le richieste di cambio linea per uno specifico turno, inviate dagli autisti.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	id	int(5)			No	*Nessuno*
2	descrizione	varchar(100)	latin1_swedish_ci		Sì	*NULL*
3	Linea	varchar(5)	latin1_swedish_ci		Sì	*NULL*
4	matricolaAut	int(4)			No	*Nessuno*
5	idturno	int(4)			No	*Nessuno*

- **Richiesta cambio turno**: Questo oggetto memorizza tutte le richieste di cambio turno, inviate dagli autisti.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	id	int(5)			No	*Nessuno*
2	descrizione	varchar(100)	latin1_swedish_ci		Sì	*NULL*
3	oraInizio	time(6)			Sì	*NULL*
4	oraFine	time(6)			Sì	*NULL*
5	matricolaAut	int(4)			No	*Nessuno*
6	idturno	int(4)			No	*Nessuno*

- **Richiesta ferie**

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	id	int(5)			No	*Nessuno*
2	dataInizio	date			Sì	*NULL*
3	dataFine	date			Sì	*NULL*
4	matricolaAut	int(4)			No	*Nessuno*

- **Richiesta straordinario**

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	id	int(5)			No	*Nessuno*
2	oraInizio	time(6)			Sì	*NULL*
3	oraFine	time(6)			Sì	*NULL*
4	matricolaAut	int(4)			No	*Nessuno*
5	Data	date			No	*Nessuno*

- **Avviso malattia**

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	id	int(11)			No	*Nessuno*
2	Data	date			No	*Nessuno*
3	Descrizione	varchar(100)	latin1_swedish_ci		No	*Nessuno*
4	MatricolaAut	int(11)			No	*Nessuno*

- **SOS**: Questo oggetto memorizza le richieste di soccorso inviate dagli autisti.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	id	int(5)			No	*Nessuno*
2	descrizione	varchar(100)	latin1_swedish_ci		Sì	*NULL*
3	datiGPS	varchar(50)	latin1_swedish_ci		Sì	*NULL*
4	matricolaAut	int(4)			No	*Nessuno*
5	CapoFabbricaMatricola	int(4)			No	*Nessuno*

- **Turno**: Questo oggetto memorizza tutti i turni di lavoro degli autisti.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	id	int(5)			No	*Nessuno*
2	start	varchar(30)	latin1_swedish_ci		No	*Nessuno*
3	end	varchar(30)	latin1_swedish_ci		No	*Nessuno*
4	idLinea	varchar(5)	latin1_swedish_ci		No	*Nessuno*
5	MatricolaAut	int(11)			No	*Nessuno*
6	AdminMatricola	int(11)			No	*Nessuno*
7	title	varchar(30)	latin1_swedish_ci		No	*Nessuno*

- **Linea**: Questo oggetto memorizza tutte le corse che effettua l'azienda.

#	Nome	Tipo	Codifica caratteri	Attributi	Null	Predefinito
1	Linea	varchar(5)	latin1_swedish_ci		No	*Nessuno*

3.4.1.1 *Definizione Sql del Database*

Di seguito è riportato la definizione del database in termini di query SQL e un diagramma che riassume quali sono gli oggetti persistenti e che relazione c'è tra gli stessi:

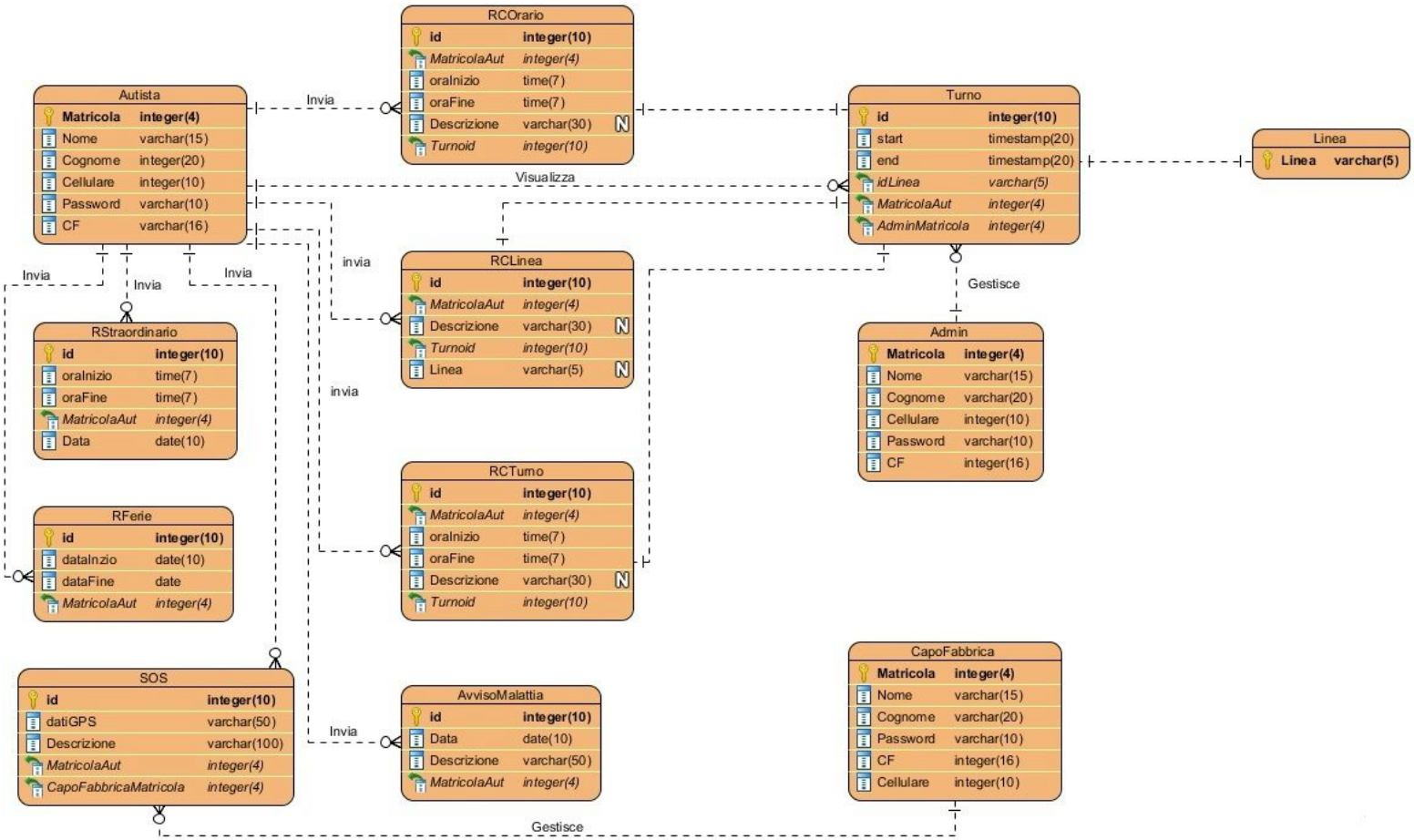
```
CREATE TABLE IF NOT EXISTS `admin` (
  `Matricola` int(4) NOT NULL,
  `Password` varchar(30) NOT NULL,
  `Nome` varchar(30) NOT NULL,
  `Cognome` varchar(35) NOT NULL,
  `Cf` varchar(16) NOT NULL,
  `Tel` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE IF NOT EXISTS `autisti` (
  `Matricola` int(4) NOT NULL,
  `Password` varchar(30) NOT NULL,
  `Nome` varchar(30) NOT NULL,
  `Cognome` varchar(35) NOT NULL,
  `Cf` varchar(16) NOT NULL,
  `Cellulare` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `avvisomalattia` (
`id` int(11) NOT NULL,
  `Data` date NOT NULL,
  `Descrizione` varchar(100) NOT NULL,
  `MatricolaAut` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;

CREATE TABLE IF NOT EXISTS `capofabbrica` (
  `Matricola` int(4) NOT NULL,
  `Password` varchar(30) NOT NULL,
  `Nome` varchar(30) NOT NULL,
  `Cognome` varchar(35) NOT NULL,
  `Cf` varchar(16) NOT NULL,
  `Tel` int(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `linee` (
  `Linea` varchar(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `rclinea` (
  `id` int(5) NOT NULL,
  `descrizione` varchar(100) DEFAULT NULL,
  `Linea` varchar(5) DEFAULT NULL,
  `matricolaAut` int(4) NOT NULL,
  `idturno` int(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `rcorario` (
  `id` int(5) NOT NULL,
  `descrizione` varchar(100) DEFAULT NULL,
  `oraInizio` time(6) DEFAULT NULL,
  `oraFine` time(6) DEFAULT NULL,
  `matricolaAut` int(4) NOT NULL,
  `idturno` int(4) NOT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `rcturno` (
  `id` int(5) NOT NULL,
  `descrizione` varchar(100) DEFAULT NULL,
  `oraInizio` time(6) DEFAULT NULL,
  `oraFine` time(6) DEFAULT NULL,
  `matricolaAut` int(4) NOT NULL,
  `idturno` int(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `rferie` (
  `id` int(5) NOT NULL,
  `dataInizio` date DEFAULT NULL,
  `dataFine` date DEFAULT NULL,
  `matricolaAut` int(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `rsos` (
  `id` int(5) NOT NULL,
  `descrizione` varchar(100) DEFAULT NULL,
  `datiGPS` varchar(50) DEFAULT NULL,
  `matricolaAut` int(4) NOT NULL,
  `CapoFabbricaMatricola` int(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `rstraordinario` (
  `id` int(5) NOT NULL,
  `oraInizio` time(6) DEFAULT NULL,
  `oraFine` time(6) DEFAULT NULL,
  `matricolaAut` int(4) NOT NULL,
  `Data` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `turno` (
`id` int(5) NOT NULL,
  `start` varchar(30) NOT NULL,
  `end` varchar(30) NOT NULL,
  `idLinea` varchar(5) NOT NULL,
  `MatricolaAut` int(11) NOT NULL,
  `AdminMatricola` int(11) NOT NULL,
  `title` varchar(30) NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;
```

```
ALTER TABLE `admin`
 ADD PRIMARY KEY (`Matricola`);

ALTER TABLE `autisti`
 ADD PRIMARY KEY (`Matricola`);

ALTER TABLE `avvisomalattia`
 ADD PRIMARY KEY (`id`), ADD KEY `MatricolaAut` (`MatricolaAut`);

ALTER TABLE `capofabbrica`
 ADD PRIMARY KEY (`Matricola`);

ALTER TABLE `linee`
 ADD PRIMARY KEY (`Linea`);

ALTER TABLE `rclinea`
 ADD PRIMARY KEY (`id`), ADD KEY `matricolaAut` (`matricolaAut`), ADD KEY `idturno`
(`idturno`);

ALTER TABLE `rcorario`
 ADD PRIMARY KEY (`id`), ADD KEY `matricolaAut` (`matricolaAut`), ADD KEY `idturno`
(`idturno`);

ALTER TABLE `rcturno`
 ADD PRIMARY KEY (`id`), ADD KEY `matricolaAut` (`matricolaAut`), ADD KEY `idturno`
(`idturno`);

ALTER TABLE `rferie`
 ADD PRIMARY KEY (`id`), ADD KEY `matricolaAut` (`matricolaAut`);

ALTER TABLE `rsos`
 ADD PRIMARY KEY (`id`), ADD KEY `matricolaAut` (`matricolaAut`), ADD KEY
`CapoFabbricaMatricola` (`CapoFabbricaMatricola`);

ALTER TABLE `rstraordinario`
 ADD PRIMARY KEY (`id`), ADD KEY `matricolaAut` (`matricolaAut`);

ALTER TABLE `turno`
 ADD PRIMARY KEY (`id`), ADD KEY `idLinea` (`idLinea`), ADD KEY `MatricolaAut`
(`MatricolaAut`), ADD KEY `AdminMatricola` (`AdminMatricola`);


ALTER TABLE `avvisomalattia`
MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
ALTER TABLE `turno`
MODIFY `id` int(5) NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=3;
```

```
ALTER TABLE `avvisomalattia`
ADD CONSTRAINT `avvisomalattia_ibfk_1` FOREIGN KEY (`MatricolaAut`) REFERENCES
`autisti` (`Matricola`);

ALTER TABLE `rclinea`
ADD CONSTRAINT `rclinea_ibfk_1` FOREIGN KEY (`matricolaAut`) REFERENCES `autisti`
(`Matricola`),
ADD CONSTRAINT `rclinea_ibfk_2` FOREIGN KEY (`idturno`) REFERENCES `turni` (`id`);

ALTER TABLE `rcorario`
ADD CONSTRAINT `rcorario_ibfk_1` FOREIGN KEY (`matricolaAut`) REFERENCES
`autisti` (`Matricola`),
ADD CONSTRAINT `rcorario_ibfk_2` FOREIGN KEY (`idturno`) REFERENCES `turni` (`id`);

ALTER TABLE `rcturno`
ADD CONSTRAINT `rcturno_ibfk_1` FOREIGN KEY (`matricolaAut`) REFERENCES
`autisti` (`Matricola`),
ADD CONSTRAINT `rcturno_ibfk_2` FOREIGN KEY (`idturno`) REFERENCES `turni` (`id`);

ALTER TABLE `rferie`
ADD CONSTRAINT `rferie_ibfk_1` FOREIGN KEY (`matricolaAut`) REFERENCES `autisti`
(`Matricola`);

ALTER TABLE `rsos`
ADD CONSTRAINT `rsos_ibfk_1` FOREIGN KEY (`matricolaAut`) REFERENCES `autisti`
(`Matricola`),
ADD CONSTRAINT `rsos_ibfk_2` FOREIGN KEY (`CapoFabbricaMatricola`)
REFERENCES `capofabbrica` (`Matricola`);

ALTER TABLE `rstraordinario`
ADD CONSTRAINT `rstraordinario_ibfk_1` FOREIGN KEY (`matricolaAut`) REFERENCES
`autisti` (`Matricola`);

ALTER TABLE `turno`
ADD CONSTRAINT `turno_ibfk_1` FOREIGN KEY (`idLinea`) REFERENCES `linee`
(`Linea`),
ADD CONSTRAINT `turno_ibfk_2` FOREIGN KEY (`MatricolaAut`) REFERENCES `autisti`
(`Matricola`),
ADD CONSTRAINT `turno_ibfk_3` FOREIGN KEY (`AdminMatricola`) REFERENCES
`admin` (`Matricola`);
```



3.5 *Access control and security*

Time-Platform è un sistema multi-user , in cui differenti attori possono accedere a differenti funzionalità e a differenti dati.
Gli attori del sistema sono :

- L'**autista**, accede alla piattaforma principalmente per visualizzare i propri turni di lavoro, inoltre può effettuare richieste di cambio orario/linea/turno , inviare richieste di soccorso nel caso in cui il pullman si guasti, richieste di eventuali ferie, ed inviare l'avviso che non può presentarsi sul posto di lavoro in quanto malato.
- L'**amministratore**, gestisce tutte le richieste inviate dall'autista, tranne la richiesta di SOS. Queste richieste vengono soddisfatte dopo aver preso nota delle notifiche presenti nella propria pagina personale. Inoltre inserisce/modifica/elimina i turni di lavoro degli autisti.
- Il **Capo fabbrica**, gestisce tutte le richieste di soccorso inviate dagli autisti.

Da questa breve descrizione si evince la necessità di capire in ogni momento "chi, può accedere a che cosa".
Per rispondere alla prima parte della domanda, è necessario un modo per identificare l'utente che sta accedendo al sistema, quindi, nel momento in cui un generico utente tenta di accedere al sistema, si deve autenticare. L'autenticazione richiede una matricola ed una

corrispondente password che è conosciuta solo da chi ha i permessi per l'accesso all'applicazione.
Le informazioni utilizzate durante il processo di autenticazione , sono sensibili, quindi viene utilizzato il protocollo HTTPS (ovvero il protocollo HTTP con l'aggiunta del protocollo crittografico SSL), quest'ultimo permette al browser web di accertarsi che il server a cui si è connessi sia autentico , ossia che corrisponda effettivamente a quello che dichiara di essere. Per verificare l'autenticità del server, il browser web utilizza la chiave pubblica indicata nel documento digitale per scambiare i dati in modo sicuro, così da garantire che non vengano intercettati da utenti esterni.

3.6 *Global software control*

Come abbiamo già detto l' architettura del nostro sistema è di tipo three-tier(tre livelli),in tal caso il controllo del flusso nei vari strati è indipendente dagli altri tranne che per la comunicazione col database. Il funzionamento del server è piuttosto semplice : esso è sempre in funzione ,attende la richiesta di un servizio da parte di un sottosistema così da poterla subito soddisfare. Nel caso in cui più sottosistemi fanno una richiesta allora queste ultime vengono soddisfatte dal server una alla volta e vengono gestite in modo sequenziale (chi prima arriva prima viene servito) offrendo ordine ed efficienza.

3.7 *Boundary conditions*

Il sistema garantisce l'esecuzione indipendente delle diverse funzionalità, essendo però un'applicazione dipendente da un server, il corretto funzionamento del sistema è strettamente legato alla sua attivazione; ognuno dei sottosistemi, infatti, usufruisce dei servizi offerti da questo ultimo, tra i quali l'utilizzo del database. Sono stati individuati due casi d'uso addizionali:
StartTimePlatformServer e StopTimePlatformServer.

StartTimePlatformServer : L'amministratore del sistema avvia la macchina server. Non appena il sistema si è avviato , gli autisti , amministratori e capi fabbrica possono accedervi ed utilizzare tutte le funzionalità.

StopTimePlatformServer : L'amministratore del sistema arresta la macchina server, di conseguenza nessun utente può accedere alla piattaforma. Non è prevista nessuna attività di notifica ai sottosistemi attivi qualora uno di questi venga disattivato. Nella fase di terminazione inoltre, non è prevista un'operazione di aggiornamento del database poiché ogni singola postazione non gestisce operazioni sui dati in locale, ma effettua le modifiche in tempo reale.

Fallimenti

La prevenzione di eventuali danni causati dall'usura dell'hardware dedicato alla memorizzazione di dati persistenti è gestita da più unità di backup. Un eventuale crash del server comporta il riavvio di ognuno dei sottosistemi ed il ripristino del database allo stato precedente al crash stesso.

4 *Servizi dei sottosistemi*

Sottosistema	Login
Descrizione	Questo sottosistema permette l'accesso alla propria pagina personale.
Servizi offerti	
Servizio	Login
Descrizione	Permette di accedere alla propria pagina personale in base alle credenziali inserite, quindi a seconda della propria mansione tale sottosistema indirizzerà l'utente verso la pagina corretta.

Sottosistema	DriverManagment
Descrizione	Questo sottosistema comprende tutte le funzioni presenti nella pagina personale dell'autista.
Servizi offerti	
Servizio	**Descrizione**
Calendar managment	Crea il calendario dove vengono inseriti i turni di lavoro dell'autista che accede al sistema.
Request change time	Invia le richieste di cambio orario.
Request change route	Invia le richieste di cambio linea.
Request change workshift	Invia le richieste di cambio turno.
Request holidays	Invia le richieste di ferie.
Alert disease	Invia l'avviso di malattia.
SOS	Invia una richiesta di soccorso al capofabbrica.

Sottosistema	AdminManagment
Descrizione	Questo sottosistema comprende tutte le funzioni presenti nella pagina personale dell'amministratore.
Servizi offerti	
Servizio	**Descrizione**
Disease notifications	Crea una lista di tutte le notifiche di malattia, inviate dagli autisti.
Time notifications	Crea una lista di tutte le notifiche di richiesta cambio orario, inviate dagli autisti.
Route notifications	Crea una lista di tutte le notifiche di richiesta cambio linea, inviate dagli autisti.
Workshift notifications	Crea una lista di tutte le notifiche di richiesta cambio turno, inviate dagli autisti.
Holidays notifications	Crea una lista di tutte le notifiche di richiesta ferie, inviate dagli autisti.