

Simulazione 28 febbraio 2019

informatica sistemi e reti

Quest'anno la traccia ministeriale, disponibile al seguente link [simulazione](#), ci presenta la progettazione un sistema di noleggio biciclette organizzato da un ente pubblico, ovvero da un Comune. La novità è rappresentata da un primo punto in cui si chiede l'infrastruttura informatica prima della classica domanda sulla progettazione della base dati che invece troviamo subito dopo. All'interno della traccia sull'infrastruttura informatica vengono già indicate diverse soluzioni tecnologiche da scegliere (RFID) per il riconoscimento delle biciclette e degli utenti.

Il sistema descritto è molto chiaro e unisce più che le materie 'Informatica' e 'Sistemi e Reti', 'Informatica' e 'Tecnologie e Progettazione di Sistemi Informatici e di Telecomunicazione' ma ormai ci dobbiamo adeguare all'idea che per il Ministero le due materie coincidano, infatti anche l'anno scorso la prova di Sistemi era di fatto la progettazione di un sistema informatico studiata nella materia TPSIT.

PRIMA PARTE

1. Quesito 1 - Progettazione infrastruttura tecnologica ed informatica

1. il progetto, anche mediante rappresentazioni grafiche, dell'infrastruttura tecnologica ed informatica necessaria a gestire il servizio nel suo complesso, dettagliando:
 - a) l'infrastruttura di comunicazione, in termini di caratteristiche dei canali, degli apparati e dei protocolli, che permette di trasmettere le informazioni di ciascuna stazione al sistema centrale;
 - b) le caratteristiche generali dei componenti hardware e software del sistema sia a livello centrale che nelle stazioni;
 - c) le misure e gli apparati per assicurare la continuità del servizio.

Per realizzare il progetto dell'infrastruttura tecnologica, sarebbe opportuno chiarire il funzionamento del sistema informativo da realizzare cominciando da un disegno dei processi e delle entità che ne sono coinvolte.

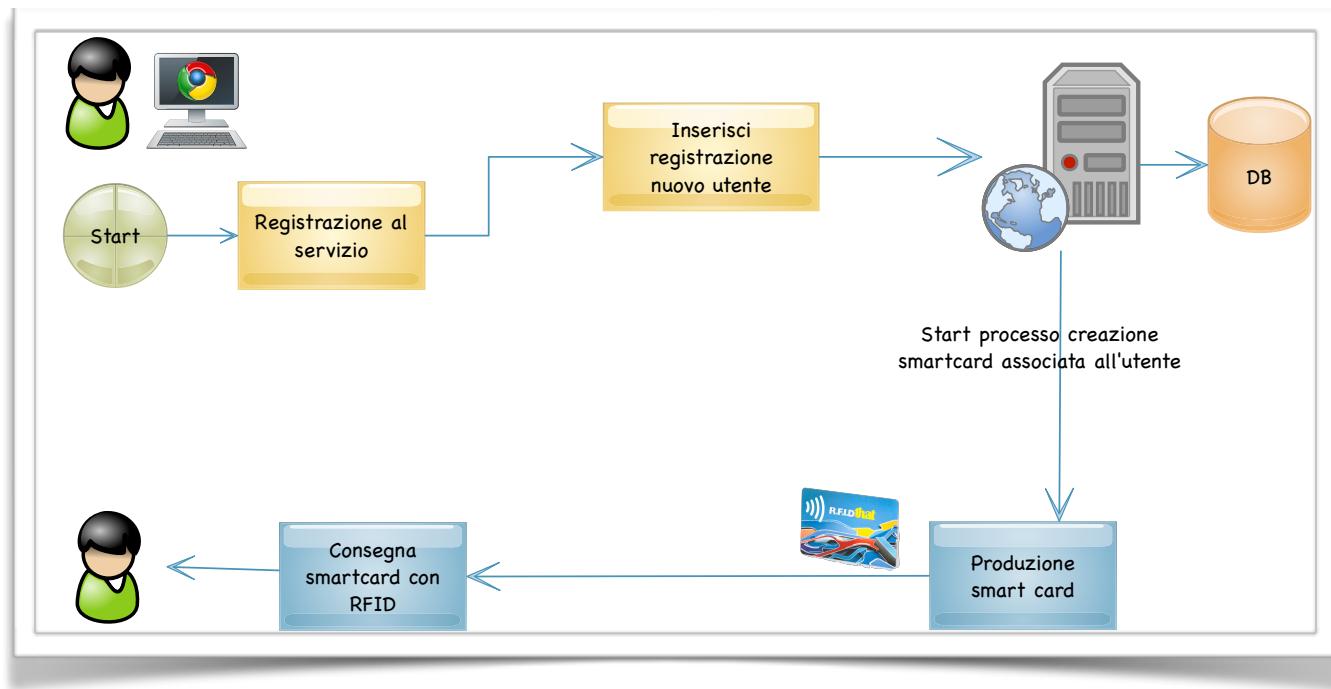
Gli attori coinvolti saranno l'utente, il sistema informativo posizionato all'interno del Comune, la stazione in cui un utente si identifica tramite smart card con un lettore RFID e tutti gli RFID reader posizionati sugli slot che bloccano le biciclette.

Come si illustra nella prima figura, la prima interazione con il nostro sistema è da parte di un "**utente**" che effettua la registrazione al servizio di noleggio biciclette.

In questa fase viene registrato un utente con una carta di credito valida su cui verranno accreditati i costi degli eventuali noleggi.

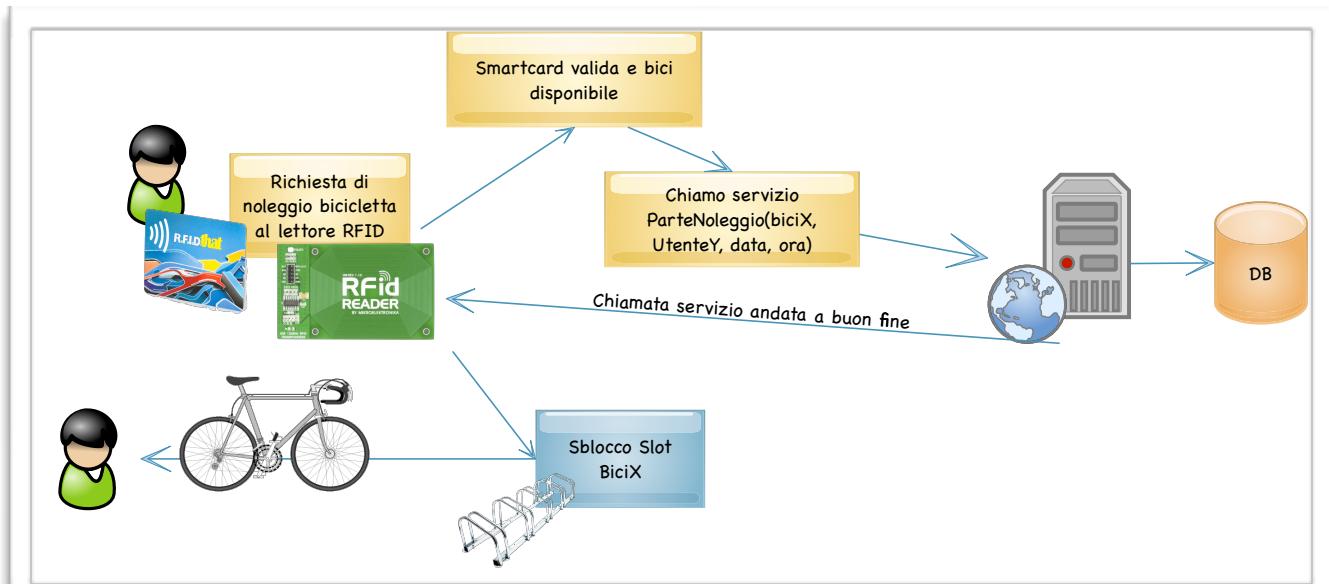
L'utente riceve la tessera a casa, oppure si reca presso un apposito ufficio del comune e solo in seguito potrà noleggiare una bicicletta. La traccia indica chiaramente che la sottoscrizione al servizio deve essere sottomessa in modalità **online**.

Il Comune mette a disposizione questo servizio noleggio e questa prima fase si conclude con la ricezione della carta RFID. Potremo anche tralasciare il duplice canale di ricezione, consegna a domicilio o presso un ufficio, perché appare solo una complicazione che non porta implicazioni successive. Un cliente provvisto di smartcard è un utente abilitato all'accesso al servizio.

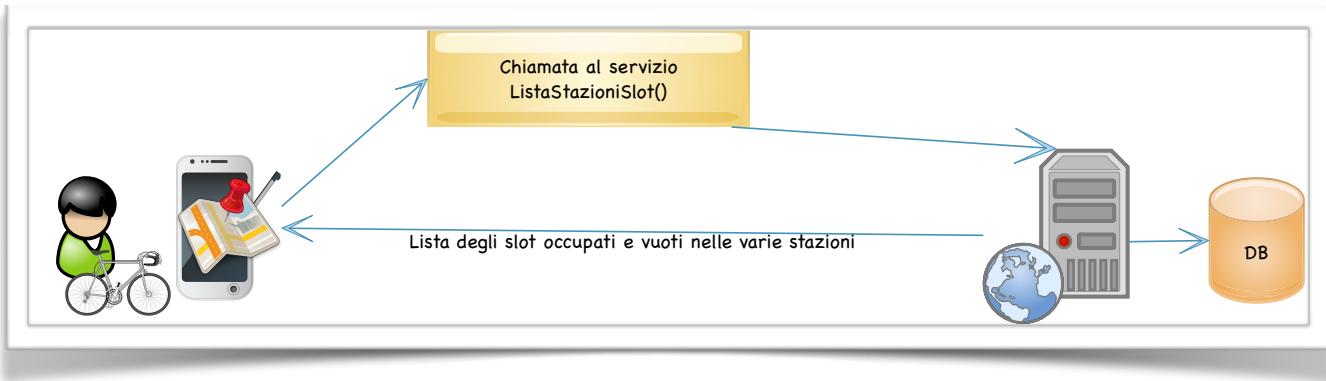


La fase successiva prevede che un utente si rechi presso una stazione che ha ancora slot con biciclette disponibili per il noleggio.

L'utente dovrà avvicinare la propria smartcard ad un apposito lettore, unico per la stazione ci specifica la traccia, e si aprirà uno slot rendendo una bici disponibile al noleggio. Nel momento in cui viene assegnata la bicicletta parte la tariffazione a tempo con una chiamata al servizio *StarNoleggio* presso i server del Comune.

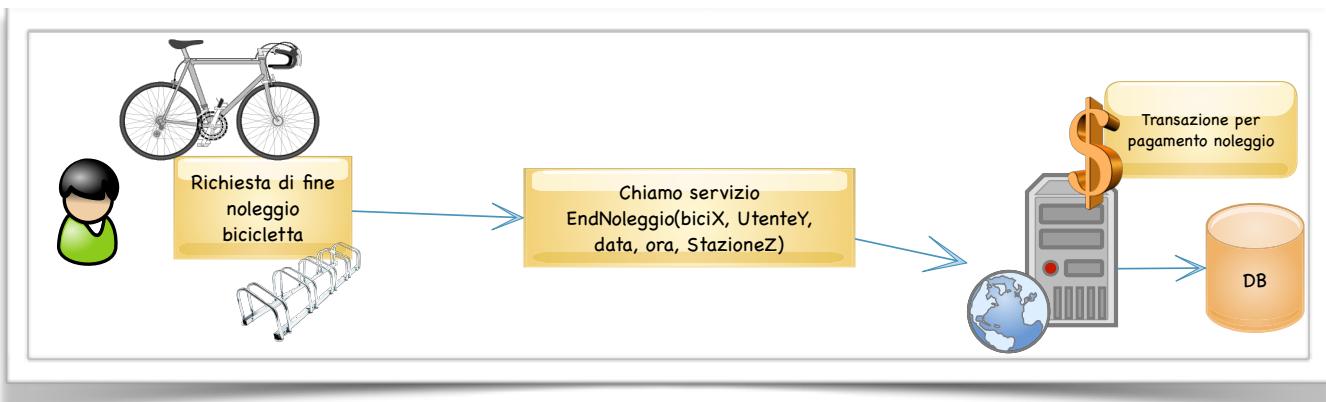


Alla fine un utente controlla, con un'app sul telefonino, dove si trova una stazione con uno slot libero per riconsegnare la bicicletta.



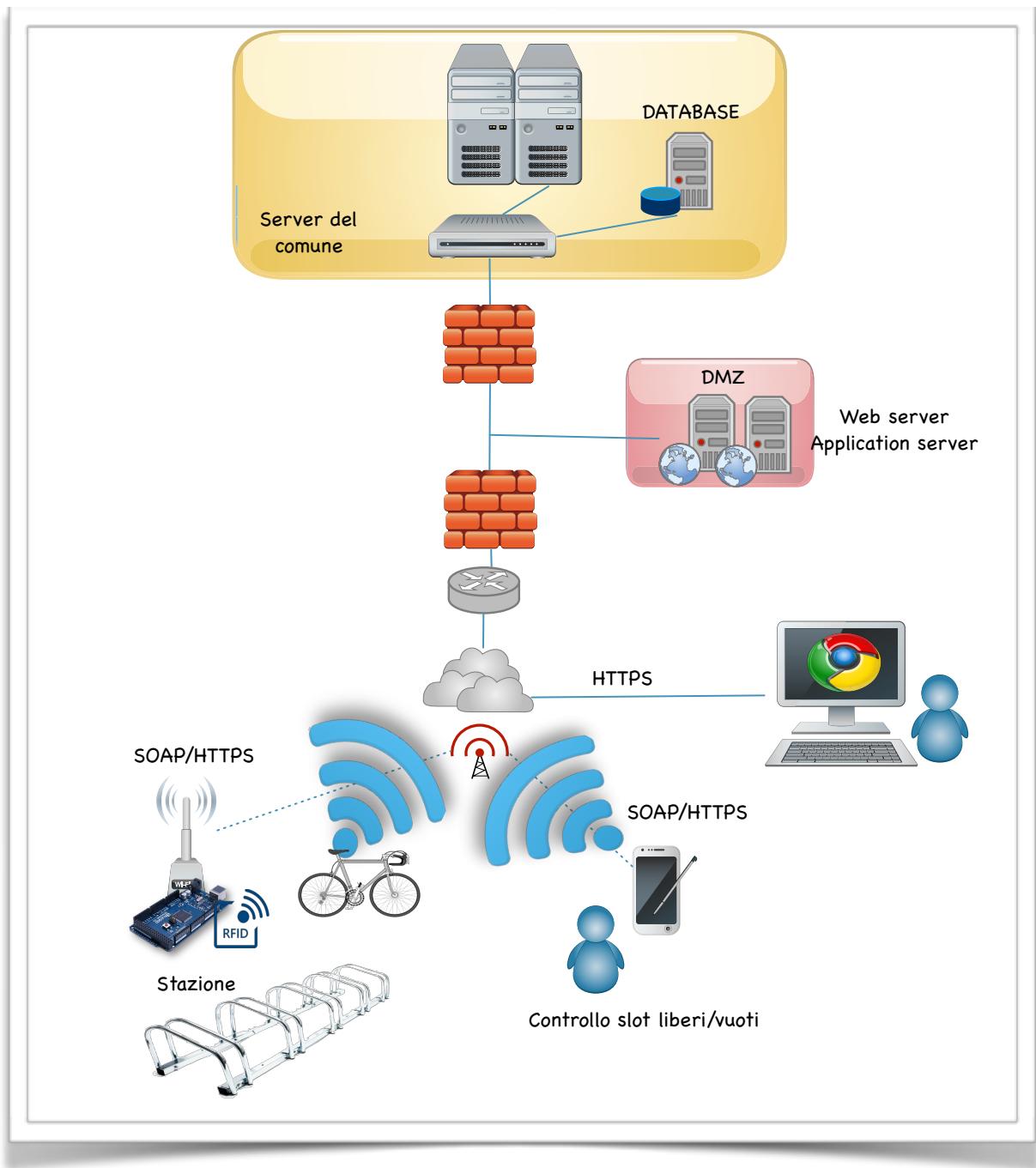
Lo stesso meccanismo di controllo slot liberi/occupati può servire anche prima del noleggio per sapere se esiste una bici disponibile in una certa stazione.

Nel momento in cui si deve riconsegnare la bicicletta, semplicemente si aggancia la bici ad uno slot libero che riconosce il codice RFID della bici ed invia un segnale al sistema della stazione che farà partire la richiesta di servizio *EndNoleggio* con tutti gli opportuni parametri per chiudere il noleggio e riscuotere il pagamento facendo partire una transazione verso la carta di credito dell'utente.



a) L'infrastruttura di comunicazione, in termini di caratteristiche dei canali, degli apparati e dei protocolli, che permette di trasmettere le informazioni di ciascuna stazione al sistema centrale;

Tutto il processo descritto si basa su una infrastruttura di comunicazione che andiamo a descrivere con un diagramma.



La fase di registrazione al Sistema si realizzerà con un web server nel Comune che riceverà delle richieste HTTPS da un client browser.

L'app su uno smartphone si collegherà al sistema tramite SOAP/HTTPS perché richiamerà un web service esposto per la richiesta di una lista delle stazioni e degli slot disponibili.

Rispetto ai Web Service di tipo REST, sembra opportuno utilizzare dei Web Service SOAP anche se più antiquati ma in questo caso garantiscono una connessione più forte tra client e server perché unicamente indirizzate al servizio e non alla condivisione di risorse. I Web Service basati su SOAP prevedono lo standard WSDL, Web Service Description Language, per definire l'interfaccia di un servizio che poi sarà invocato in

remoto. WSDL è una IDL (Interface Description Language) per un componente software.

Il sistema presente alla stazione di noleggio avrà un arduino con una GSM shield che

con gli opportuni moduli, lettore RFID e librerie per funzionare come un client SOAP, comunicheranno con i server del comune.

Il lettore RFID, all'interno della scatoletta/tower in cui sarà posizionato il sistema alla stazione dovrà prevedere uno schermo a cristalli liquidi LCD per le interazioni con l'utente, leggerà l'ID della smartcard e lo invierà al sistema del Comune per richiederne la validazione e l'associazione all'utente, della bici e dell'inizio del noleggio. Possiamo immaginare che sul display LCD possa

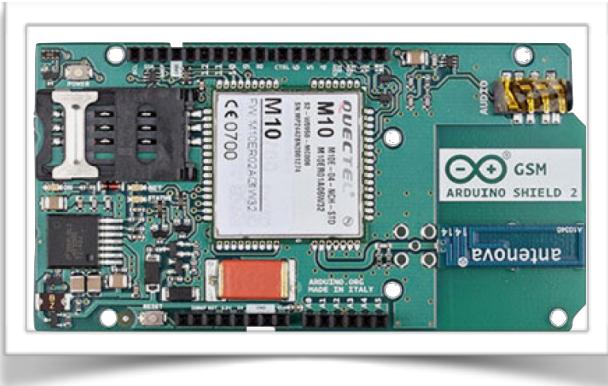
comparire data o ora, messaggi di errori eventuali, come tipo carta di credito scaduta e soprattutto il numero di slot aperto per prelevare la bicicletta. Sarà arduino stesso a comandare l'apertura del bloccaggio bicicletta, basterà una delle tante serrature elettriche in commercio.

Si avrà un RFID reader accanto ad ogni sistema di bloccaggio e un RFID montato sulla bicicletta per leggerne il codice univoco. In realtà, si dovrebbe anche prevedere un meccanismo montato sulla bicicletta per evitarne il furto, il codice RFID ha un raggio di pochi centimetri. Si potrebbe pensare ad una tecnologia beacon che ha un raggio fino a 70 metri oppure un localizzatore GPS per il recupero delle biciclette in caso di mancata restituzione oltre un certo periodo di tempo. Su questo la traccia non ci chiede però di progettare una soluzione in tal senso, per cui partiamo dall'assunzione che il sistema conosce l'accoppiata bicicletta/slot con tecnologia RFID da leggere sia quando viene noleggiata che quando viene riconsegnata.

b) le caratteristiche generali dei componenti hardware e software del sistema sia a livello centrale che nelle stazioni;

Per l'hardware avremo bisogno di macchine server di fascia medio/alta per contenere un web server anche open source come Apache. Per l'application server ed il database sarebbe preferibile affidarci ad un prodotto commerciale che garantisce una assistenza 24/7 giorni su 7. Per il database infatti sarebbe preferibile affidarci ad una scelta commerciale, ad esempio Oracle per avere più affidabilità e che offre anche una piattaforma per esporre servizi come web service(Oracle Application Server 10g). La scelta commerciale si basa sulla considerazione che abbiamo una sistema informatico che deve tutelare le carte di credito dei cittadini e deve funzionare garantendo una continuità di servizio costante (ne parleremo nel punto c).

Per le pagine web utilizzeremo HTML e PHP, mentre per l'app su smartphone android e per i web service programmaremo in JAVA. Per la parte di arduino dovreмо programmare in C tramite l'IDE arduino utilizzando le opportune librerie a seconda dei componenti utilizzati.



c) le misure e gli apparati per assicurare la continuità del servizio

(Ripreso dalla mia soluzione dalla traccia del 2018 - c'era, anche in quel caso, quasi la stessa domanda)

La *business continuity* e la *fault tolerance* sono dei punti nevralgici all'interno di una sistema informatico. Una qualsiasi società si volesse imporre sul mercato, avrebbe bisogno di un settore dedicato alla gestione della business continuity, fault tolerance e del disaster recovery. Già prima avevamo accennato come la scelta dei prodotti software sia di centrale importanza per un sistema che voglia mantenere un livello di servizi sempre molto alto. I servizi offerti devono essere sempre disponibili, resistenti ai guasti e anche ad eventuali calamità naturali. La soluzione di affidarsi a enti terzi, che possano garantire una assistenza h24, è un primo livello di sicurezza indispensabile per quanto riguarda la funzionalità dei prodotti software. Ci si deve affidare ad apparati di rete collaudati e sicuri ed anche a gestori telefonici, con copertura quasi totale sul territorio nazionale. Sia i sistemi di elaborazione che i dati in essi contenuti sono di fondamentale importanza, per questo motivo vanno custoditi e cautelati da rischi di natura "accidentale", come ad esempio incendi ma anche terremoti o inondazioni.

Per gli incendi, le sale server devono essere a norma e predisposte con opportuni allarmi e sistemi anti-incendio. Per i terremoti o le inondazioni, di sicuro ci saranno dei rilevamenti sulla sismicità della zona e anche una collocazione lontana dalla costa o da fiumi. Per questo tipo di rischi di natura accidentale, di fondamentale importanza è un sistema di *backup* ed eventuale *recovery*. In genere in una azienda ci sono delle persone dedicate alla gestione e al controllo dei backup di tutti i dati e delle infrastrutture software. Se il budget lo consente, per una maggiore sicurezza e per garantire in qualsiasi caso la *business continuity*, in genere si crea un sistema di "*replica*" ovvero un sistema informativo totalmente duplicato, con macchine meno potenti che entreranno in funzione nel caso di guasti o di indisponibilità del sistema principale. A volte sono anche macchine che partecipano in load balancing con i server principali, anche se hanno capacità ridotte di smistamento traffico.

Un ulteriore problema, da tenere sempre sotto controllo, riguarda l'alimentazione dei sistemi informativi. Si devono prevenire rischi elettrici legati alle cadute di tensione, ad un fulmine o ad un guasto di qualche centralina elettrica. In questo caso le sale con i server devono essere predisposte con degli opportuni gruppi di continuità che dovranno garantire l'erogazione della corrente elettrica anche in caso di blackout.

Per quanto riguarda i rischi legati agli accessi illegali e fraudolenti al sistema, si metteranno in atto tutte le opportune misure di sicurezza per autenticare ed autorizzare, tramite anche impronta digitale o lettori di badge, solo gli utenti che effettivamente hanno le credenziali per accedere al sistema.

Le comunicazioni saranno sempre su un canale criptato e i programmi saranno sempre subordinati ad un accesso tramite login e password.

2. Quesito numero 2- il database

2. il progetto della base di dati per la gestione delle informazioni relative agli utenti, alle operazioni di noleggio e riconsegna delle biciclette ed alla situazione di occupazione delle stazioni: in particolare si richiede il modello concettuale e il corrispondente modello logico.

Rileggiamo la traccia evidenziando in giallo quelle che diventeranno le nostre entità e gli eventuali attributi in verde.

Il Comune di una città europea di medie dimensioni vuole implementare, per sostenere politiche di mobilità sostenibile, un servizio di noleggio di biciclette attraverso stazioni di “noleggio e riconsegna” dislocate in diversi punti della città. Al fine di addebitare il costo del servizio di noleggio, si vuole conoscere in ogni momento chi ha preso in uso una determinata bicicletta.

Il servizio è fruibile previa registrazione online dei dati dell’utente, incluso un numero di carta di credito valida. A seguito della registrazione, il Comune provvederà alla consegna di una tessera elettronica (smart card) al domicilio dell’utente o presso appositi uffici, che conterrà il codice identificativo dell’utente leggibile in modalità senza contatto (*contactless*).

Ogni stazione di noleggio e riconsegna è dotata di cinquanta slot, ciascuno dei quali può ospitare una bicicletta ed è dotato di un sistema di blocco meccanico della bicicletta stessa, mediante un lucchetto controllato elettronicamente. Per noleggiare una bicicletta, l’utente dovrà avvicinare la propria tessera elettronica ad un apposito lettore, unico per la stazione: di conseguenza verrà sbloccata una delle biciclette inserite negli slot. Ogni bicicletta è dotata di un proprio tag a radiofrequenza (RFID) che ne riporta il codice univoco: questo tag viene letto da un apposito dispositivo su ogni slot (RFID reader) sia in ingresso che in uscita della bicicletta. L’utente potrà successivamente riconsegnare la bicicletta presso una qualsiasi stazione cittadina (quella di noleggio o un’altra) che abbia slot liberi. In questo modo, per ogni stazione è sempre possibile sapere quali biciclette sono bloccate negli slot e disponibili per il noleggio, quali sono state noleggiate e quali vengono riconsegnate.

L’operazione di noleggio o di riconsegna di una bicicletta comporta la registrazione dei seguenti dati:

- identificativo della bicicletta noleggiata o riconsegnata
- identificativo dell’utente
- data e ora dell’operazione
- identificativo della stazione di noleggio o di riconsegna

La registrazione dei dati delle due operazioni è finalizzata anche alla loro trasmissione in tempo reale ad un sistema centrale per il monitoraggio, controllo e tariffazione del servizio.

Per mezzo di una mappa, visualizzabile su web o su app per telefono cellulare, si può conoscere per ogni stazione cittadina quante biciclette sono disponibili per il noleggio e quanti slot sono liberi per la riconsegna di una bicicletta noleggiata.

Dopo una opportuna analisi e diversi ripensamenti, pare evidente che la soluzione migliore sia optare per le seguenti entità:

UTENTE: L’attore che interagisce con il nostro sistema, un codice univoco come PK assegnato dal sistema che poi verrà riportato nella smart card.

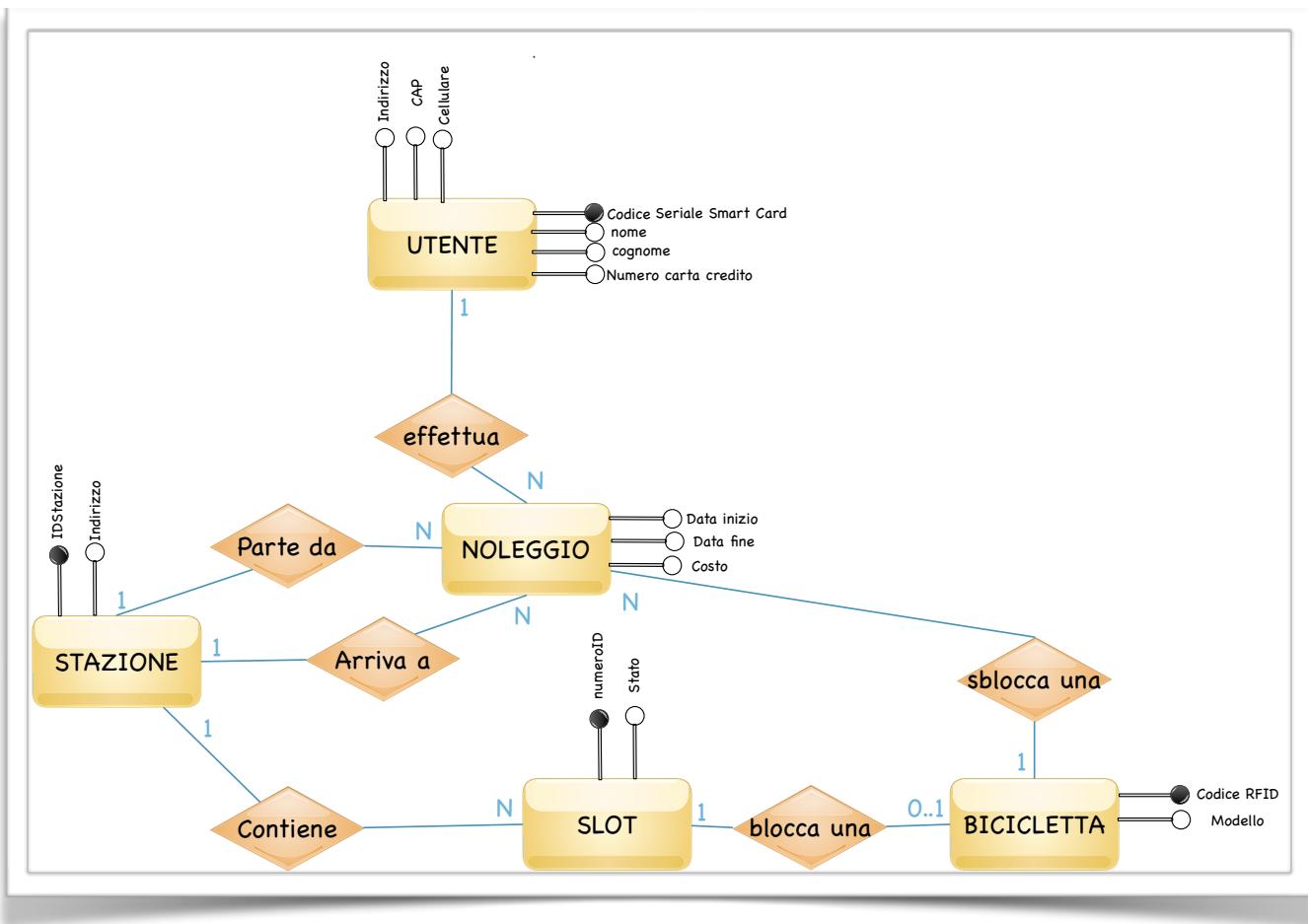
BICICLETTA: Ha un codice univoco associato per cui è evidente che si tratti di una entità.

STAZIONE: La stazione è il contenitore degli slot e delle biciclette.

SLOT: Appare evidente dalle query richieste che lo slot debba diventare una entità perché contiene lo stato occupato/libero e sembra opportuno di assegnargli un numero progressivo da 1 a 50 come chiave primaria dovendo indicare, nel momento in cui l'utente vuole una bicicletta, quale numero è sbloccato per rilasciargli la bicicletta.

NOLEGGIO: appare utile inserire questa entità debole che è in relazione con quasi tutte le altre entità. Conviene inserirla come entità per comodità ma anche perché dalla traccia l'operazione NOLEGGIO viene invocata più volte con un suo costo e una data di inizio e di riconsegna.

Il diagramma ER risultante:



Relazione **effettua** tra UTENTE e NOLEGGIO uno-a-molti: un utente effettua nell'arco del tempo più noleggi ma un singolo noleggio riguarda un solo utente.

Relazione **sblocca una** tra BICICLETTA e NOLEGGIO uno-a-molti: una bicicletta viene noleggiata nell'arco del tempo più volte ma un singolo noleggio riguarda una sola bicicletta.

Relazione **parte da** tra STAZIONE e NOLEGGIO uno-a-molti: un noleggio parte da una singola stazione che ovviamente può avere molti noleggi.

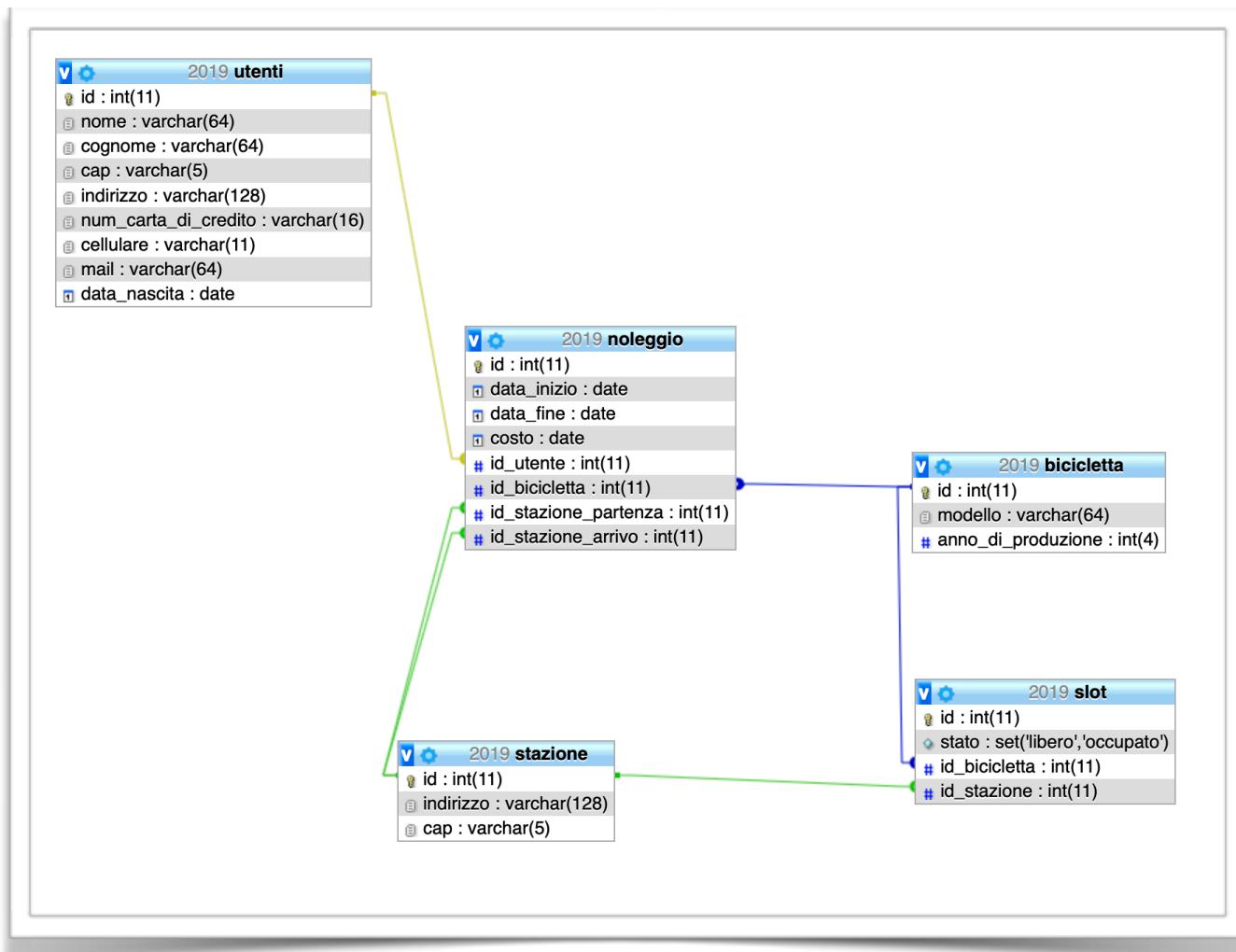
Relazione **arriva a** tra STAZIONE e NOLEGGIO uno-a-molti: un noleggio arriva e termina in una certa stazione che può avere molti noleggi.

Relazione **blocca una** tra BICICLETTA e SLOT uno-a-uno: una bicicletta viene associata ad uno slot ma uno slot può avere o non avere una bicicletta trovandosi nello stato libero o occupato, se è occupato contiene una sola bicicletta con un determinato RFID letto dal suo lettore, altrimenti il legame è nullo e lo slot risulta vuoto.

Relazione **contiene** tra STAZIONE e SLOT uno-a-molti: classica relazione ‘contenitore’ una stazione contiene più slot, ma un determinato slot, numerato da 1 a 50 appartiene ad una e una sola stazione.

Modello logico

Applicando le regole di derivazione abbiamo il seguente modello logico:



Le entità diventano tabelle, prendono un nome al plurale e aggiungiamo le foreign key per risolvere le relazioni uno-a-molti.

NOLEGGI si porta dietro le foreign key verso le STAZIONI sia di arrivo che di partenza, UTENTI, BICICLETTE ed è stato aggiunto un **id** per praticità altrimenti la PK sarebbe stata composta da 3 record: data_inizio, utente e bicicletta. La data_fine e stazione_di_arrivo possono anche essere NULL nel momento in cui il noleggio è ancora in corso e non concluso.

La tabella SLOT si porta le foreign key verso la stazione che lo contiene e la bicicletta contenuta, che si sposta da uno slot all'altro ma non necessita di uno storico ma solo della informazione estemporanea nel momento in cui si deve effettuare un noleggio. La foreign key verso la bicicletta può essere NULL nel caso di assenza di bicicletta. Lo stato dello slot è stato definito come un SET dai soli due valori 'libero', 'occupato'.

Riporto il modello fisico, non richiesto dalla traccia, nel caso si voglia riprodurre il database:

```

CREATE TABLE biciclette (
    id int(11) NOT NULL,
    modello varchar(64) NOT NULL,
    anno_di_produzione int(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE noleggi (
    id int(11) NOT NULL,
    data_inizio date NOT NULL,
    data_fine date DEFAULT NULL,
    costo date DEFAULT NULL,
    id_utente int(11) NOT NULL,
    id_bicicletta int(11) NOT NULL,
    id_stazione_partenza int(11) NOT NULL,
    id_stazione_arrivo int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE slot (
    id int(11) NOT NULL,
    stato set('libero','occupato') NOT NULL,
    id_bicicletta int(11) DEFAULT NULL,
    id_stazione int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE stazioni (
    id int(11) NOT NULL,
    indirizzo varchar(128) NOT NULL,
    cap varchar(5) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE utenti (
    id int(11) NOT NULL,
    nome varchar(64) NOT NULL,
    cognome varchar(64) NOT NULL,
    cap varchar(5) NOT NULL,
    indirizzo varchar(128) NOT NULL,
    num_carta_di_credito varchar(16) NOT NULL,
    cellulare varchar(11) NOT NULL,

```

```
mail varchar(64) NOT NULL,
data_nascita date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

ALTER TABLE biciclette
ADD PRIMARY KEY (id);

ALTER TABLE noleggi
ADD PRIMARY KEY (id),
ADD KEY bici (id_bicicletta),
ADD KEY ut (id_utente),
ADD KEY par (id_stazione_partenza),
ADD KEY arr (id_stazione_arrivo);

ALTER TABLE slot
ADD PRIMARY KEY (id),
ADD KEY st (id_stazione),
ADD KEY bic (id_bicicletta);

ALTER TABLE stazioni
ADD PRIMARY KEY (id);

ALTER TABLE utenti
ADD PRIMARY KEY (id);

ALTER TABLE noleggi
ADD CONSTRAINT arrivo FOREIGN KEY (id_stazione_arrivo) REFERENCES stazioni (id),
ADD CONSTRAINT bici FOREIGN KEY (id_bicicletta) REFERENCES biciclette (id),
ADD CONSTRAINT partenza FOREIGN KEY (id_stazione_partenza) REFERENCES stazioni (id),
ADD CONSTRAINT ut FOREIGN KEY (id_utente) REFERENCES utenti (id);

ALTER TABLE slot
ADD CONSTRAINT bic FOREIGN KEY (id_bicicletta) REFERENCES biciclette (id),
ADD CONSTRAINT st FOREIGN KEY (id_stazione) REFERENCES stazioni (id);
COMMIT;
```

3. Quesito 3 - pagine html e php

3. il progetto delle pagine web che permettono le seguenti funzioni, codificandone una con i linguaggi ritenuti più idonei:
 - a) a partire da una mappa delle stazioni, verificare se una certa stazione ha biciclette disponibili per il noleggio;
 - b) consentire al gestore del sistema di visualizzare le bici attualmente in uso, da quali utenti e presso quale stazione sono state prelevate.

In questo punto vengono chiesti sia i codice php e anche le relative query.

La form HTML per richiedere le biciclette disponibili:

```
<html>
<head>
    Ricerca Biciclette libere
</head>
<body>
    <form action="2019.php" method="POST" name="form">

        <label for="stazione">Id Stazione: </label>
        <input id="stazione" name="stazione" type="text" size="30"><br><br>

        <input type="submit" value="Ricerca"><br>
    </form>
</body>
</html>
```

La pagina php chiamata:

```
<html>
<head>
    <title>Esempio traccia 2019</title>
</head>
<body>
    <?
    $stazione=$_POST['stazione'];
    $conn = new mysqli("localhost","root","","2019");
    if ($conn->connect_error) {
        die("Connessione fallita con errore: " . $conn->connect_error);
    }
    $query = "select stazione.id as id, slot.id_bici as bici from slot,
    stazione where slot.id_stazione = stazione.id and stato = 'occupato' and stazione.id= '".$stazione."'";
    $result = $conn->query($query);
    $i = 0;
    if ($result->num_rows > 0) {
        echo "<table border='1'>";
        echo "<tr><td bgcolor=\"#FFFFFF\">Codice Stazione</td><td  bgcolor=\"#FFFFFF\">Bici</td></tr>";

        while($row = $result->fetch_assoc()) {
            if($i % 2 == 0){
                echo "<tr><td bgcolor=\"#BBBBBB\">&ampnbsp".$row['id'];
                echo "</td><td  bgcolor=\"#BBBBBB\">&ampnbsp".$row['bici']."'</tr>";
            }
            else{
                echo "<tr><td bgcolor=\"#DDDDDD\">&ampnbsp".$row['id'];
                echo "</td><td  bgcolor=\"#DDDDDD\">&ampnbsp".$row['bici']."'</tr>";
            }
            $i++;
        }
    } else {
        echo "0 results";
    }
    echo "</table>";
    $conn->close();
    ?>
</body>
</html>
```

Per visualizzare le biciclette disponibili sul nostro database si devono interrogare gli slot: ‘occupati’.

Per il punto b riporto la query che visualizza le bici attualmente in uso, da quali utenti e presso quale stazione sono state prelevate:

```
select biciclette.id, utenti.nome, utenti.cognome,
noleggi.id_stazione_partenza
from biciclette, utenti, noleggi
where
noleggi.id_utente = utenti.id
and noleggi.id_bicicletta = biciclette.id;
```

SECONDA PARTE

QUESITO II

Si dovevano scegliere due tra i quesiti proposti.

Il quesito II chiede delle ulteriori query sul database disegnato.

- II. In relazione al tema proposto nella prima parte, si sviluppino in linguaggio SQL le query che consentono di soddisfare le seguenti richieste:
- dato il codice di una bicicletta elencare gli utenti che l'hanno utilizzata nel mese corrente
 - mostrare la stazione presso la quale è stato effettuato il maggior numero di noleggi in un dato periodo.

Query a:

```
select utenti.nome, utenti.cognome
from utenti, noleggi
where
noleggi.id_bicicletta = 1
and utenti.id = noleggi.id_utente
and MONTH(noleggi.data_inizio) = MONTH(CURRENT_DATE());
```

Query b:

```
select noleggi.id_stazione_partenza, count(noleggi.id) as
countnol
from noleggi
where (noleggi.data_inizio BETWEEN '2019-01-01' AND
'2019-03-01')
group by noleggi.id_stazione_partenza
order by countnol DESC limit 1
```

Piuttosto che il max, per la seconda query è molto più semplice utilizzare un ordinamento discendente e prendere la prima linea, ovvero quella che ha maggiori noleggi in una stazione per un determinato periodo di tempo.

Il secondo quesito, a cui si poteva dare una facile risposta, poteva essere il **IV** che comunque è una domanda di teoria a cui poter rispondere riportando di sistemi LDAP, sistemi OTP (one time password) e molti altri riportati nel programma di sistemi e reti.