



Laurea Triennale in Informatica - Università di Salerno
Corso di Ingegneria del Software - Professore A. De Lucia

RAD

Requirements Analysis Document



Sistema in breve





Obiettivi



- Creare un sistema di semplice utilizzo per raccogliere e gestire gli ordini dei clienti
- Semplificare la gestione della sala
- Fornire al gestore del locale una rapida vista delle entrate economiche
- Dare la possibilità all'executive chef di gestire con comodità gli aggiornamenti del menù



Requisiti funzionali



- **Gestione del locale**
- **Gestione dei dipendenti**
- **Gestione del menù**
- **Gestione delle comande**
- **Richiedere assistenza**





Requisiti non funzionali

- **Usabilità**
 - Semplice ed intuitivo
- **Affidabilità**
 - Dati sempre disponibili e protetti
- **Performance**
 - Sistema concorrente, leggero e performante
- **Supportabilità**
 - Aperto a futuri aggiornamenti
- **Implementazione**
 - Android, Java
- **Packaging**
 - Installazione e fornitura hardware a carico dei produttori





Attori



Proprietario



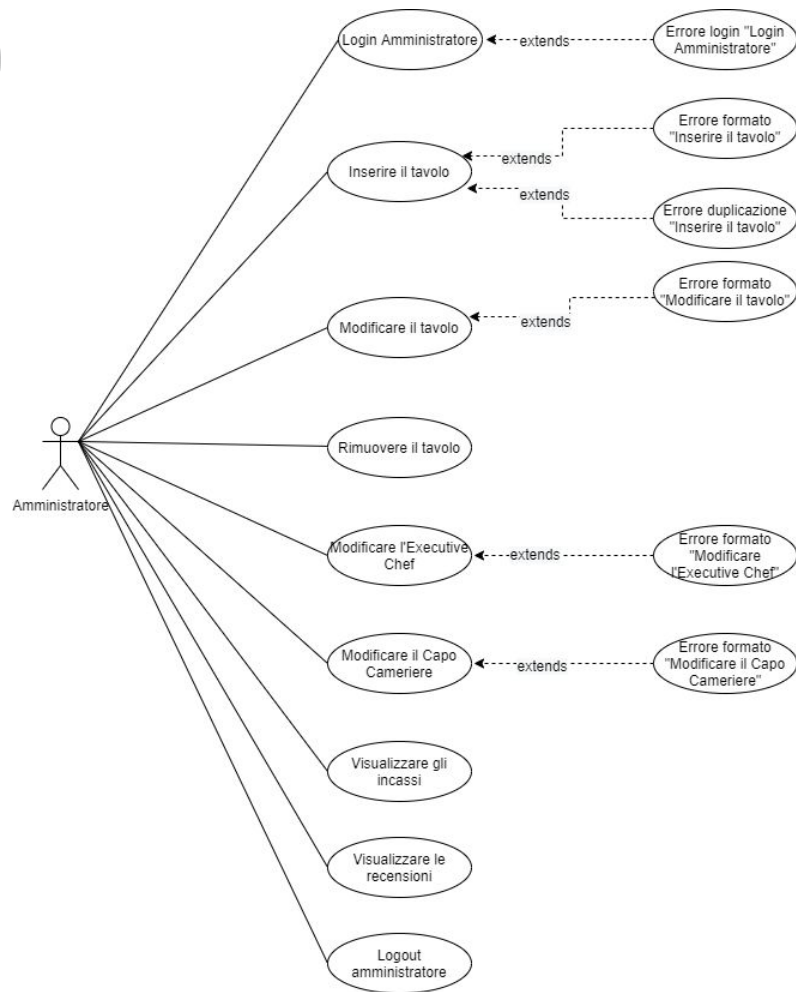
Executive Chef



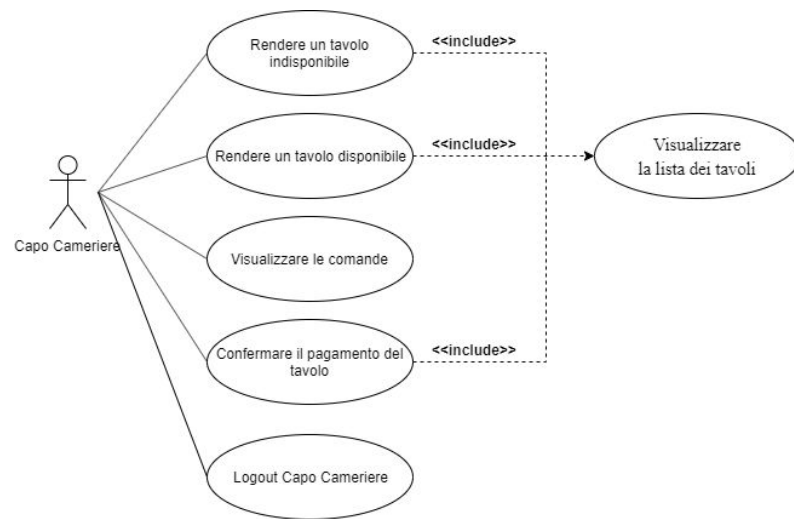
Capo Cameriere



Tavolo



Use case diagram





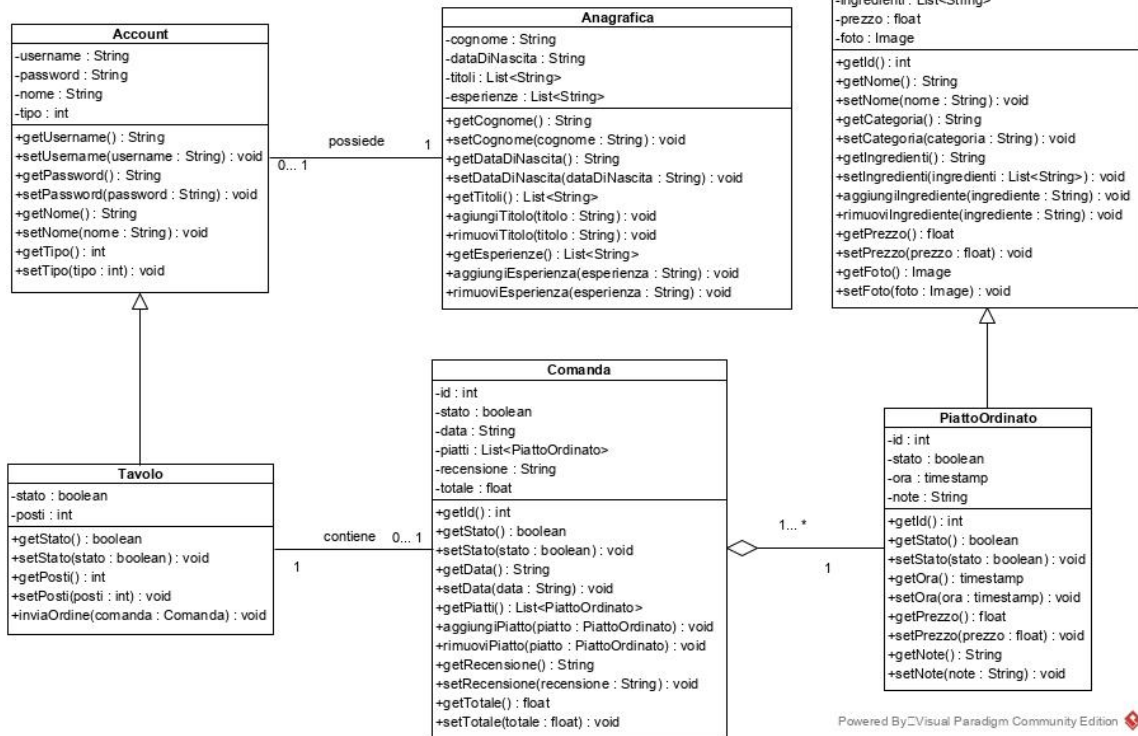
Use case diagram



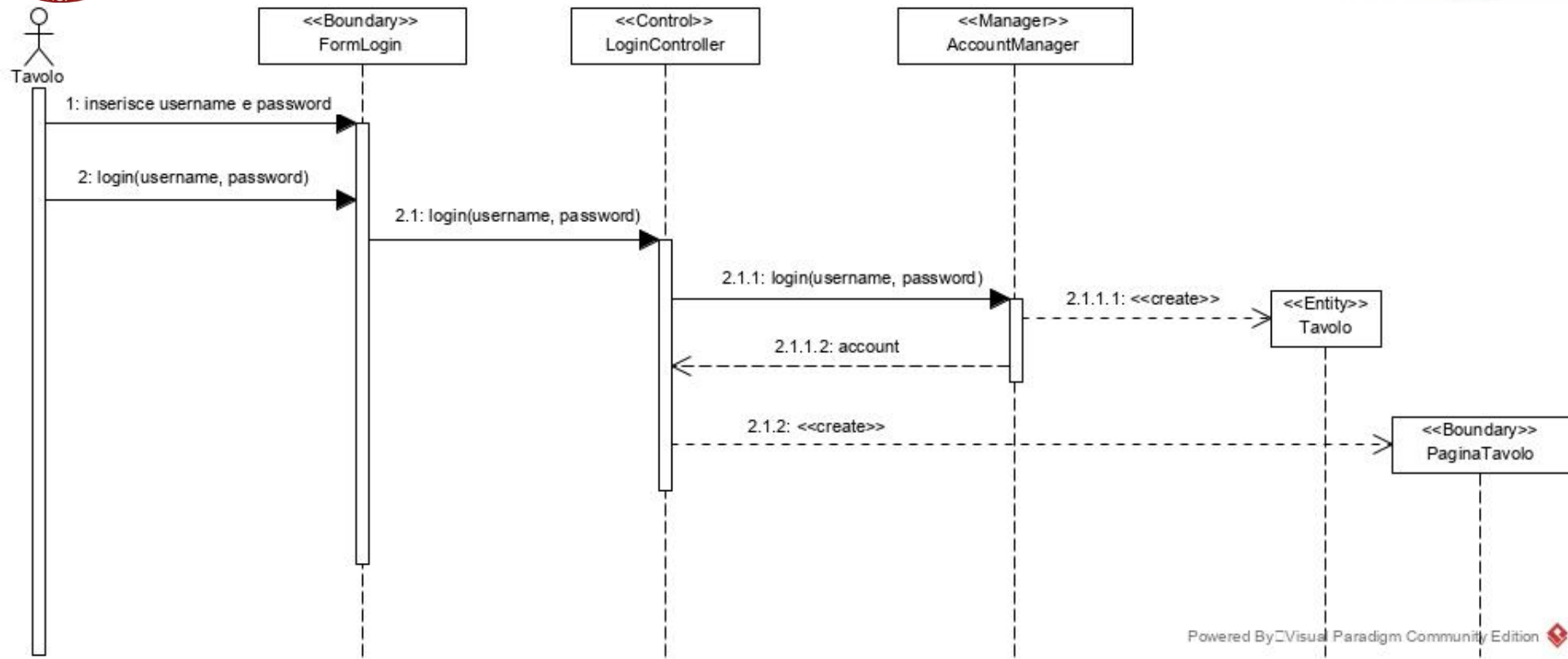
Use case

Id	EC05	
Nome	Spuntare i piatti pronti	
Attore iniziale	Executive Chef	
Precondizioni	<ul style="list-style-type: none"> - È stato effettuato il login come Executive Chef - Executive Chef si trova nella schermata delle <u>comande aperte</u> 	
Flusso degli eventi	Attore	Sistema
	1. Lo chef visualizza le comande aperte, trova il piatto che è stato appena preparato e lo spunta	
		2. Il sistema riceve il comando e richiede conferma all'executive chef
	3. Lo chef conferma l'operazione	
		4. Il sistema spunta il piatto come pronto e aggiorna la lista delle comande
Condizione di uscita	Il piatto è stato marcato come pronto	

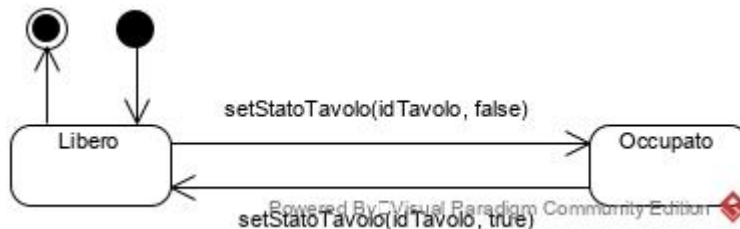
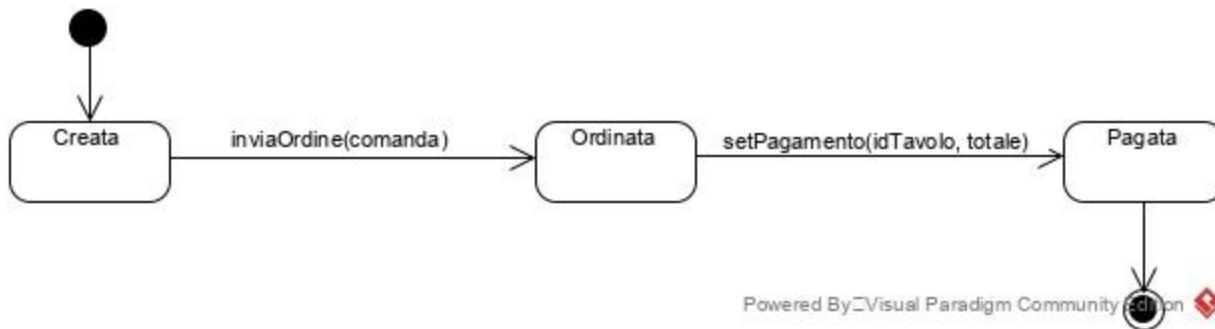
Object diagram

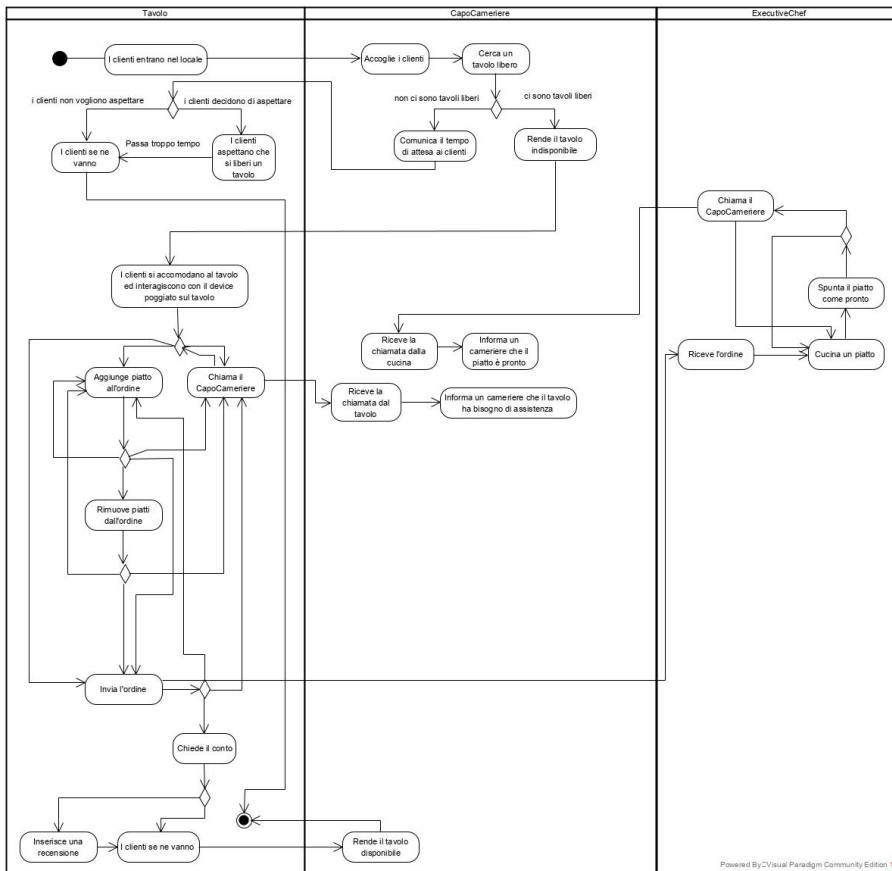


Sequence diagram



State chart diagram







Sottosistema implementato

- gestione della sala da parte del proprietario;
- gestione dei tavoli della sala da parte del capo cameriere (libera tavolo, occupa tavolo, conferma pagamento del tavolo);
- invio e ricezione delle notifiche;
- visualizzazione del menu;
- creazione, modifica, invio e rimozione di un ordine;
- visualizzazione e gestione delle comande.

— — —

SDD

System Design Document





Design goals



➤ Criteri di prestazioni

- Throughput
- Memoria

71%10:02

Specialità dello chef

Nel carrello

Antipasti	Kofta	25.5€	-	1	+	
Primi	Crema di castagne con finferli e speck	28.5€	-	1	+	
Secondi	Mozzarella in carrozza	3.5€	-	1	+	
Contorni	Vellutata di carote e zenzero	2.5€	-	2	+	
Bevande	Roselline di pasta fresca con salmone e pomodorini gialli	25.5€	-	1	+	
	Vino Spumante Shāh Mat Bianco	39.5€	-	5	+	

CHIAMA CAMERIERE

Totale: 285.5€

CONFERMA

Tavolo 1

CARRELLO

CHIUDI PASTO


Design goals

➤ Criteri di affidabilità

- Robustezza del sistema
- Attendibilità
- Disponibilità
- Sicurezza

➤ Criteri di costi

- Costi di sviluppo
- Costo di installazione
- Costo manutenzione
- Costo amministrazione



⚠ Credenziali errate

Username

tavolo

Password

...

LOGIN






Design goals

➤ Criteri di manutenzione

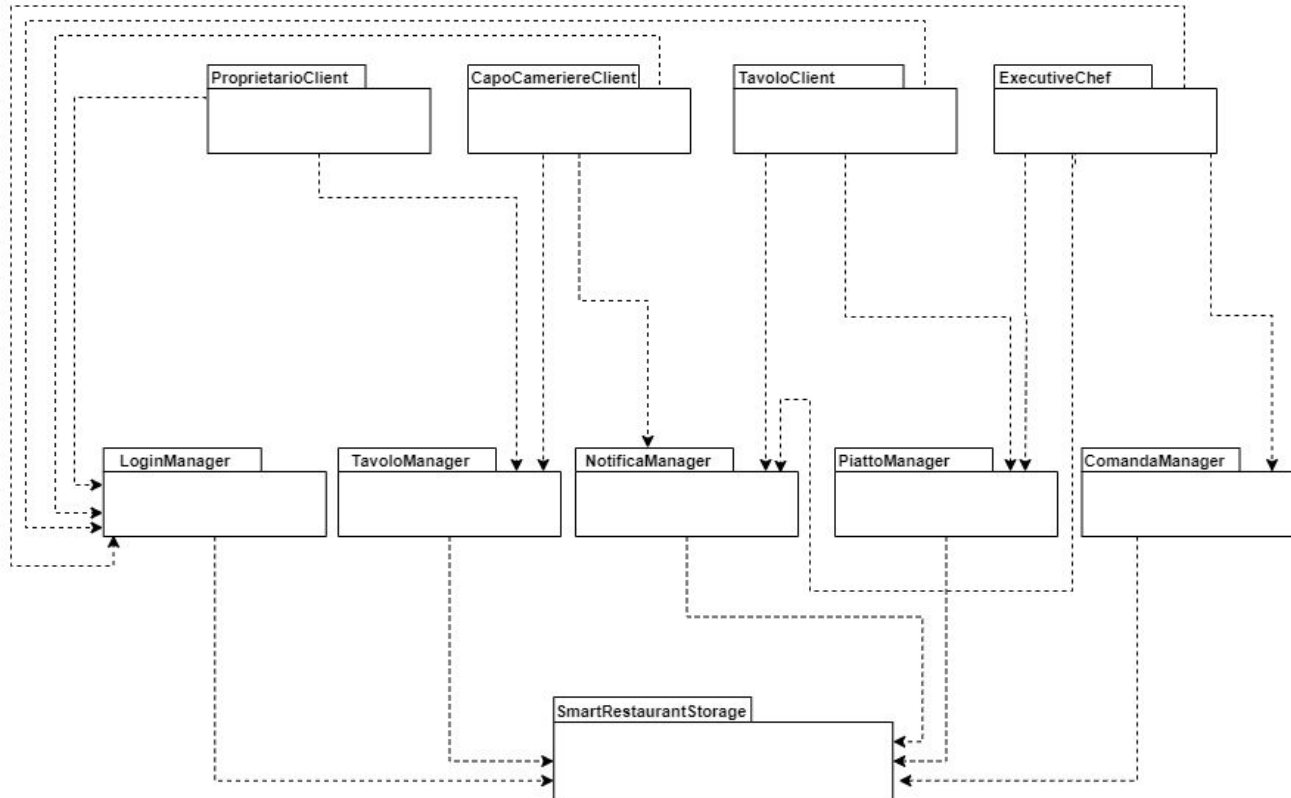
- Estensibilità
- Modificabilità
- Portabilità
- Leggibilità
- Tracciabilità dei requisiti

➤ Criteri end-user

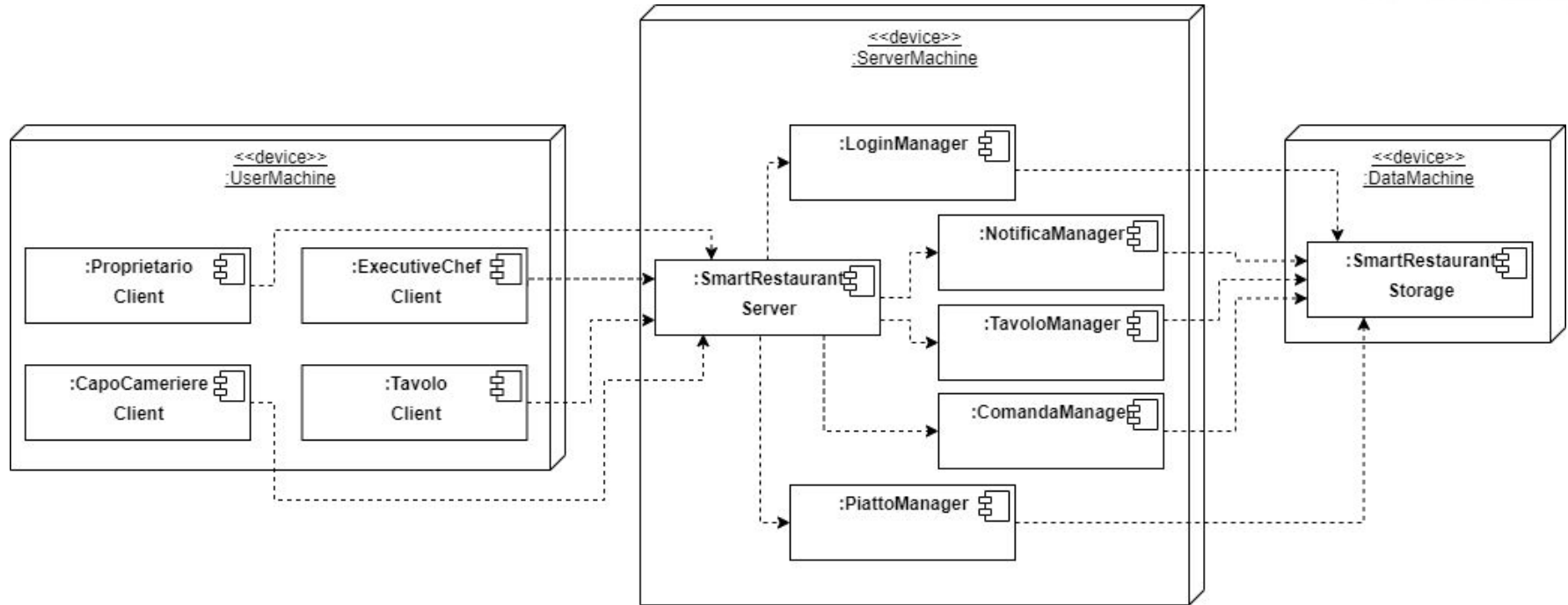
- Usabilità

Smart Restaurant - 10.09		
Notifiche	Tavolo 1	
Tavoli liberi	Tavolo 16	
Tavoli occupati	Tavolo 17	
Pagamenti	Tavolo 18	
	Tavolo 19	
	Tavolo 20	
	Tavolo 3	
	Tavolo 5	
	Tavolo 6	
	Tavolo 7	
		<div>  <div> Chef Kofta 25.5€ </div> </div>
		<div>  <div> Chef Crema di castagne con finferli e speck 28.5€ </div> </div>
		<div>  <div> Antipasti Mozzarella in carrozza 3.5€ </div> </div>
		<div>  <div> Primi Vellutata di carote e zenzero 2.5€ </div> </div>
		<div>  <div> Primi Roselline di pasta fresca con salmone e pomodorini gialli 25.5€ </div> </div>
Totale: 285.5€		<div>SEGNA COME PAGATO</div>

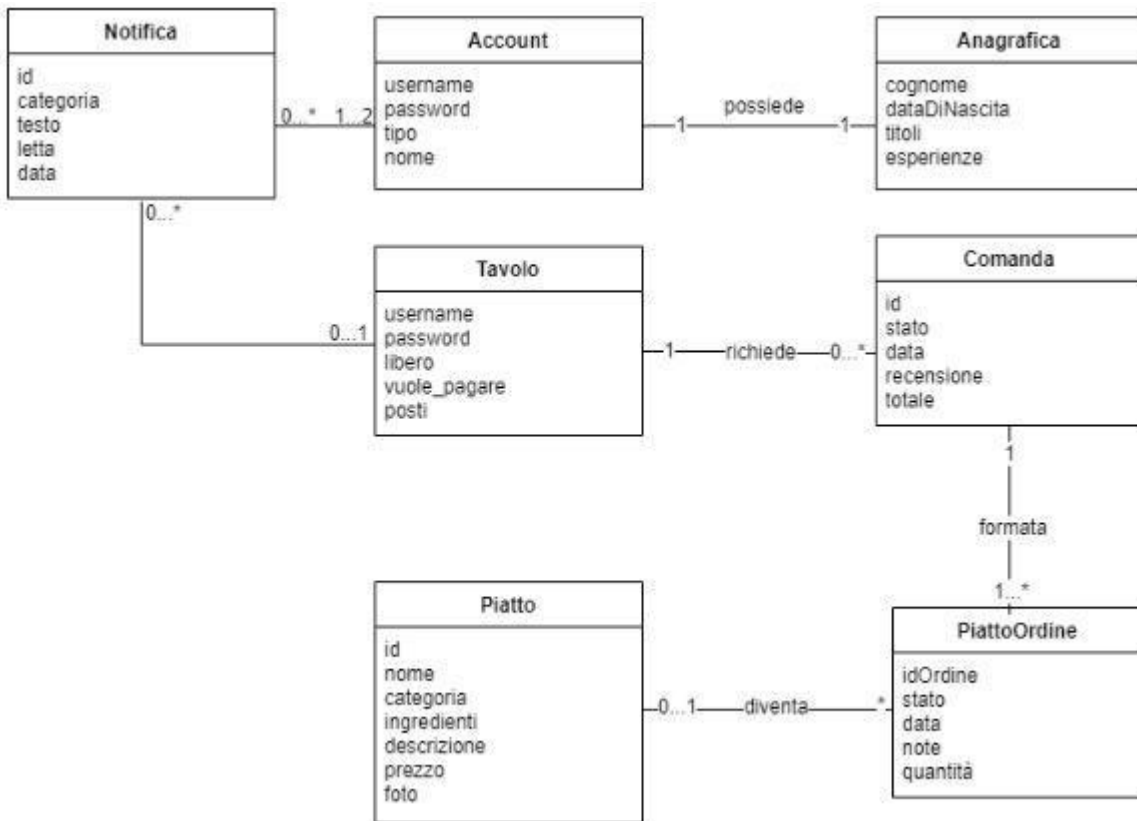
Decomposizione in sottosistemi



Mapping hardware-software



Gestione delle persistenze



Controllo degli accessi

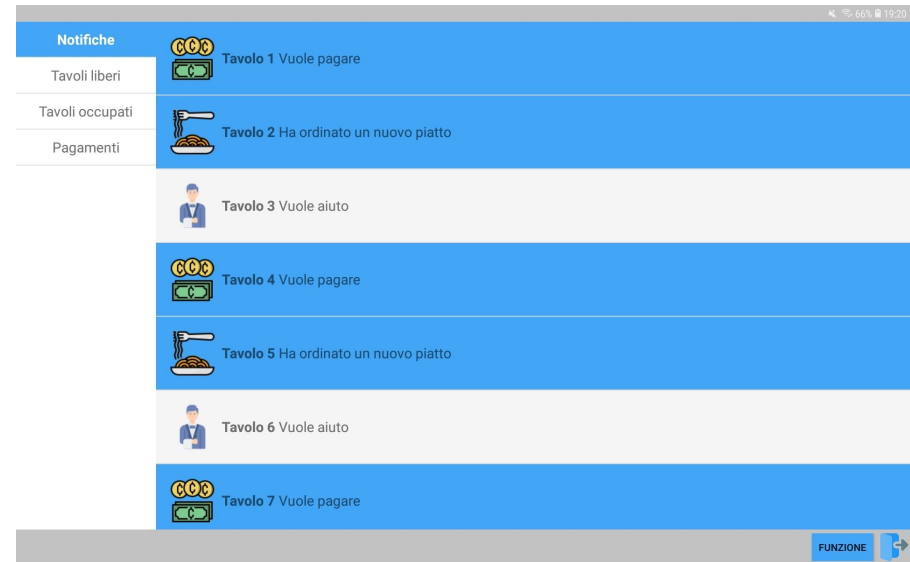
Utilizziamo una matrice di tipo capability

Oggetti Attori	Tavolo	Menu	PiattoOrdinato	Comanda	Notifica
Proprietario	crea visualizza modifica rimuovi				
Executive Chef		visualizza	visualizza check rimuovi	visualizza check	crea
Capo Cameriere	visualizza modifica		visualizza	visualizza modifica	visualizza
Tavolo	visualizza	visualizza	crea visualizza modifica rimuovi	crea visualizza modifica	crea



Controllo globale del sistema

Il sistema è di tipo
event-based



ODD

Object Design Document

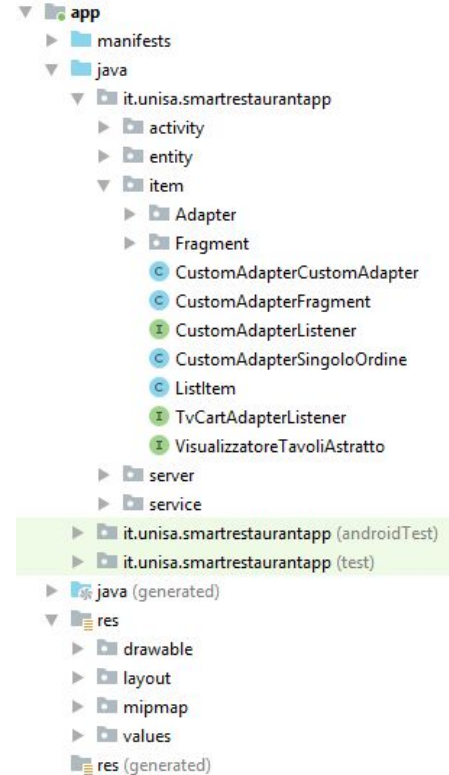
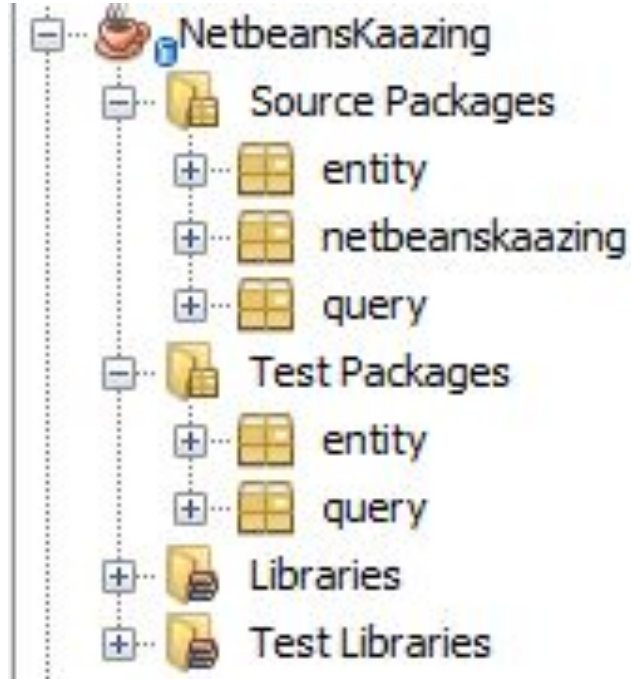


Object Design Trade-offs

- **Sicurezza vs efficienza**
- **Response time vs hardware**

— — —

Struttura dei progetti



Implementazione



Linguaggi utilizzati





Tecnologie utilizzate





Richiesta client



```
private static SmartRestaurantDispatcher sender;
```

```
/**
```

```
 * Crea un ComandaService. Inizializza la coda
```

```
 * sulla quale verranno inviati i messaggi
```

```
 */
```

```
public ComandaService() {
```

```
    sender = new SmartRestaurantDispatcher( wsurl: "ws://" + DbManager.getIp() + ":" + DbManager.getPorta() + "/jms", destination: "/queue/cucinaTabletToServer");
```

```
}
```

```
/**
```

```
 * Richiede tutte le comande presenti nel database
```

```
 * @param UUID ID univoco della richiesta
```

```
 */
```

```
public void requestAllComande(String UUID) {
```

```
    HashMap<String, String> properties = new HashMap<>();
```

```
    properties.put("action", DbManager.COMANDE_GET_ALL);
```

```
    properties.put("UUID", UUID);
```

```
    sender.sendMessage( msg: "", properties);
```

```
}
```

Risposta server

```
/**
 * Si occupa di fornire tutte le comande alla cucina
 */
cucinaTabletToServer.subscribe(new MessageListener() {
    @Override
    public void onMessage(Message msg) {
        try {
            String azione = msg.getStringProperty("action");
            String uniqueId = msg.getStringProperty("UUID");
            System.out.println("Ho ricevuto: " + msg.getStringProperty("action") + ", UUID: " + uniqueId);
            TavoloQuery tavoloQuery = new TavoloQuery();
            int id_piatto_ordinato;
            PiattoOrdinatoQuery poq = null;
            PiattoOrdinato piatto = null;
            ComandaQuery cmq = new ComandaQuery();

            switch (azione) {
                case DbManager.COMANDE_GET_ALL:
                    List<Tavolo> tavoli = tavoloQuery.findAllComanda();
                    Gson gson = new Gson();
                    String json = gson.toJson(tavoli);

                    HashMap<String, String> properties = new HashMap<>();
                    properties.put("UUID", uniqueId);
                    properties.put("tavoli", json);

                    cucinaServerToTablet.sendMessage("Comande", properties);
                    break;
            }
        }
    }
});
```


Query al Database

```
/**
 * Restituisce tutti i tavoli occupati aventi una comanda
 * @return tutti i tavoli occupati aventi una comanda
 */
@Override
public List<Tavolo> findAllComanda() {
    ArrayList<Tavolo> tavoli = new ArrayList();

    try {
        connection = DriverManagerConnectionPool.getConnection();

        String query = "SELECT t.username, aes_decrypt(t.password, \"\" + DbManager.getKey() + \"\") as password,"
            + "t.nome, t.posti, t.libero, t.vuole_pagare, R1.data FROM Tavolo AS t\n"
            + "JOIN (SELECT c.id, username, stato, data, recensione, totale FROM Comanda AS c"
            + "JOIN (SELECT MAX(c.id) AS id FROM Comanda AS c GROUP BY c.username) AS R1 WHERE R1.id = c.id) AS R1\n"
            + "WHERE t.username = R1.username AND t.libero = false ORDER BY R1.data DESC;";

        PreparedStatement statement = null;
        statement = connection.prepareStatement(query);
        ResultSet rs = statement.executeQuery();

        while(rs.next()) {
            Tavolo newTavolo = new Tavolo(rs.getString("username"), rs.getString("password"), rs.getString("nome"),
                rs.getInt("posti"), rs.getBoolean("libero"), rs.getBoolean("vuole_pagare"));

            ComandaQuery comandaQuery = new ComandaQuery();
            Comanda comanda = comandaQuery.findByUsername(newTavolo.getUsername());
            if (comanda != null) {
                newTavolo.setComanda(comanda);
                tavoli.add(newTavolo);
            }
        }
    } catch (SQLException ex) {
        ex.printStackTrace();
    } finally {
        close();
    }

    return tavoli;
}
```



Risultato client



Ordini

Specialità dello chef

Antipasti

Primi

Secondi

Contorni

Dolci

Bevande

Tavolo 1

Antipasti

Mozzarella in carrozza ☒ ☐

Primi

Vellutata di carote e zenzero ☒ ☐

Roselline di pasta fresca con salmone e pomodorini gialli ☒ ☐

Tavolo 2

Secondi

Tonno in crosta di pistacchio ☒ ☐

Spiedini di polpo e calamari ☒ ☐

Dolci

Naked cake ☒ ☐

CHIAMA CAMERIERE

INDIETRO

AVANTI





Demo live

Testing



Tecnologie utilizzate





Testing



è stata utilizzata la tecnica di category partition per testare le funzionalità sulle possibili combinazioni di valori che può inserire l'utente



Testing



9.2 Gestione sala

9.2.1 Aggiungere un tavolo

Parametro	Username	
	Formato	<code>^[a-zA-Z0-9]{1,50}+\$</code>
Formato fu	<ol style="list-style-type: none">1. Non rispetta il formato [errore]2. Rispetta il formato [property formatoUok]	
Lunghezza lu	<ol style="list-style-type: none">1. $1 \leq \text{lunghezza} \leq 50$ [if formatoUok] [property lunghezzaUok]2. $\text{lunghezza} < 1$ [errore]3. $\text{lunghezza} > 50$ [errore]	

Parametro	Password	
	Formato	<code>^[a-zA-Z0-9]{3,50}+\$</code>
Formato fp	<ol style="list-style-type: none">1. Non rispetta il formato [errore]2. Rispetta il formato [property formatoPok]	
Lunghezza lp	<ol style="list-style-type: none">1. $3 \leq \text{lunghezza} \leq 50$ [if formatoPok] [property lunghezzaPok]2. $\text{lunghezza} < 3$ [errore]3. $\text{lunghezza} > 50$ [errore]	

Parametro	Nome	
	Formato	<code>^[a-zA-Z0-9]{1,70}+\$</code>
Formato fn	<ol style="list-style-type: none">1. Non rispetta il formato [errore]2. Rispetta il formato [property formatoNok]	
Lunghezza ln	<ol style="list-style-type: none">1. $1 \leq \text{lunghezza} \leq 70$ [if formatoNok] [property lunghezzaNok]2. $\text{lunghezza} < 1$ [errore]3. $\text{lunghezza} > 70$ [errore]	

Parametro	Posti	
	Formato	<code>^[1-9]{1,3}+\$</code>
Range rpo	<ol style="list-style-type: none">1. numero < 1 [errore]2. numero > 999 [errore]3. $1 \leq \text{numero} \leq 999$	

Parametro	Stato (libero)	
	Formato	Boolean
Formato fs	<ol style="list-style-type: none">1. true2. false	

Testing

Codice	Combinazione	Esito
TC_2.1_01	fu1	[errore]
TC_2.1_02	fu2, lu2	[errore]
TC_2.1_03	fu2, lu3	[errore]
TC_2.1_04	fu2, lu1, fp1	[errore]
TC_2.1_05	fu2, lu1, fp2, lp2	[errore]
TC_2.1_06	fu2, lu1, fp2, lp3	[errore]
TC_2.1_07	fu2, lu1, fp2, lp1, fn1	[errore]
TC_2.1_08	fu2, lu1, fp2, lp1, fn2, ln2	[errore]
TC_2.1_09	fu2, lu1, fp2, lp1, fn2, ln3	[errore]
TC_2.1_10	fu2, lu1, fp2, lp1, fn2, ln1, rpo1	[errore]
TC_2.1_11	fu2, lu1, fp2, lp1, fn2, ln1, rpo2	[errore]
TC_2.1_12	fu2, lu1, fp2, lp1, fn2, ln1, rpo3, fs1	[inserimento]
TC_2.1_13	fu2, lu1, fp2, lp1, fn2, ln1, rpo3, fs2	[inserimento]

Testing

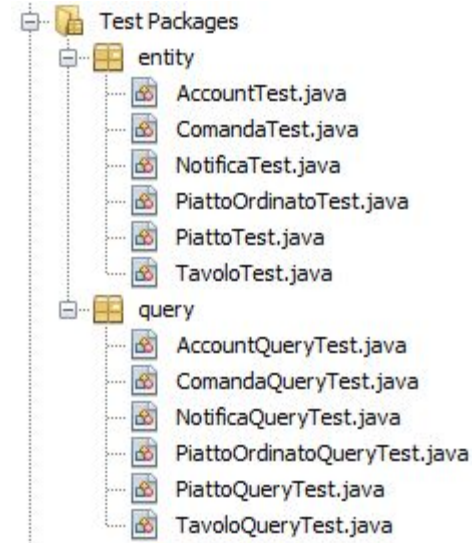
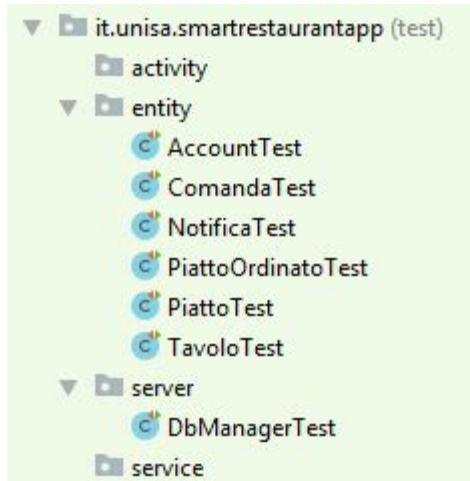
TC_2.1 Aggiungere un tavolo

Test Case ID:	TC_2.1_01												
Precondizioni													
Si è loggati con un account di tipo "Proprietario" e viene inserito un nuovo tavolo.													
Flusso degli eventi													
<p>1. Viene compilato il form di inserimento con i seguenti dati</p> <table border="1"> <thead> <tr> <th>Input</th><th>Valore</th></tr> </thead> <tbody> <tr> <td>Username</td><td>T@volo150</td></tr> <tr> <td>Password</td><td>pass</td></tr> <tr> <td>Nome</td><td>Crystal</td></tr> <tr> <td>Posti</td><td>6</td></tr> <tr> <td>Libero</td><td>True</td></tr> </tbody> </table>		Input	Valore	Username	T@volo150	Password	pass	Nome	Crystal	Posti	6	Libero	True
Input	Valore												
Username	T@volo150												
Password	pass												
Nome	Crystal												
Posti	6												
Libero	True												
2. L'utente clicca sul pulsante "salva"													
Oracolo													
L'operazione non va a buon fine perché il campo "Username" non rispetta il formato.													

Testing

Test Case ID:	TC_2.1_12												
Precondizioni													
Si è loggati con un account di tipo "Proprietario" e viene premuto il tasto relativo all'inserimento di un tavolo.													
Flusso degli eventi													
<p>1. Viene compilato il form di inserimento con i seguenti dati</p> <table> <tr> <th>Input</th><th>Valore</th></tr> <tr> <td>Username</td><td>Tavolo1</td></tr> <tr> <td>Password</td><td>pass</td></tr> <tr> <td>Nome</td><td>Crystal</td></tr> <tr> <td>Posti</td><td>6</td></tr> <tr> <td>Libero</td><td>True</td></tr> </table> <p>2. L'utente clicca sul pulsante "salva"</p>		Input	Valore	Username	Tavolo1	Password	pass	Nome	Crystal	Posti	6	Libero	True
Input	Valore												
Username	Tavolo1												
Password	pass												
Nome	Crystal												
Posti	6												
Libero	True												
Oracolo													
L'inserimento viene eseguito con successo.													

Testing con JUnit



Testing con JUnit (client)

```
@Before
public void setUp() throws Exception {
    account = new Account();
    account.setName("Gino");
    account.setUsername("Proprietario");
    account.setPassword("pass");
    account.setTipo(Account.TYPE_PROPRIETARIO);
    Assert.assertEquals(account.toString(), actual: "Account{" +
        "username='Proprietario' " +
        "password='pass' " +
        "nome='Gino' " +
        "tipo=" + Account.TYPE_PROPRIETARIO +
        "'}");

    account = new Account( username: "Proprietario", password: "pass", nome: "Gino", Account.TYPE_PROPRIETARIO);
}

@After
public void tearDown() throws Exception {
}

@Test
public void getUsername() { Assert.assertEquals(account.getName(), actual: "Gino"); }

@Test
public void setUsername() { Assert.assertEquals(account.getUsername(), actual: "Proprietario"); }

@Test
public void getPassword() { Assert.assertEquals(account.getPassword(), actual: "pass"); }
```



Testing con JUnit (server)

```
/**
 * Test of getTipo method, of class Account.
 */
@Test
public void testGetTipo() {
    System.out.println("getTipo");
    Account instance = new Account("username", "password", "nome", 0);
    Integer expResult = 0;
    Integer result = instance.getTipo();
    assertEquals(expResult, result);
}

/**
 * Test of setTipo method, of class Account.
 */
@Test
public void testSetTipo() {
    System.out.println("setTipo");
    Integer tipo = 0;
    Account instance = new Account("username", "password", "nome", 0);
    instance.setTipo(tipo);
    Integer result = instance.getTipo();
    assertEquals(tipo, result);
}
```



Testing con Espresso

- ▼ it.unisa.smartrestaurantapp (androidTest)
 - activity
 - ▼ gestioneaccount
 - LoginProprietario
 - LoginTC1101
 - LoginTC1102
 - LoginTC1103
 - LoginTC1104
 - LoginTC1105
 - LoginTC1106
 - LoginTC1107
 - LogoutTC1201
 - LogoutTC1202
 - LogoutTC1203
 - LogoutTC1204
 - ▼ gestinemenu
 - VisualizzareMenuTC401
 - ▼ gestioneordini
 - CancellarePreparazionePiattoTC703
 - ConfermareIlPagamentoTC504
 - ConfermarePreparazionePiattoTC702
 - InserireUnPiattoTC3101
 - InserireUnPiattoTC3102
 - InserireUnPiattoTC3103
 - InserireUnPiattoTC3104
 - InviareOrdineInCucinaTC602
 - RichiederelPagamentoTC3201
 - RichiederelPagamentoTC3202
 - RimuoverePiattoDaOrdineTC601

- VisualizzareLeComandeTC701
 - ▼ gestionesala
 - ▼ aggiunta
 - AggiungereUnTavoloTC2101
 - AggiungereUnTavoloTC2102
 - AggiungereUnTavoloTC2103
 - AggiungereUnTavoloTC2104
 - AggiungereUnTavoloTC2105
 - AggiungereUnTavoloTC2106
 - AggiungereUnTavoloTC2107
 - AggiungereUnTavoloTC2108
 - AggiungereUnTavoloTC2109
 - AggiungereUnTavoloTC2110
 - AggiungereUnTavoloTC2111
 - AggiungereUnTavoloTC2112
 - AggiungereUnTavoloTC2113
 - ▼ liberazione
 - LiberaTavoloTC503
 - ▼ modifica
 - ModificareUnTavoloTC2201
 - ModificareUnTavoloTC2202
 - ModificareUnTavoloTC2203
 - ModificareUnTavoloTC2204
 - ModificareUnTavoloTC2205
 - ModificareUnTavoloTC2206
 - ModificareUnTavoloTC2207
 - ModificareUnTavoloTC2208
 - ModificareUnTavoloTC2209
 - ModificareUnTavoloTC2210

- ▼ notifiche
 - VisualizzaNotificaTC505
- ▼ occupazione
 - OccupiTavoloTC502
 - AggiungereUnTavoloTC2101
 - AggiungereUnTavoloTC2102
 - AggiungereUnTavoloTC2103
 - AggiungereUnTavoloTC2104
 - AggiungereUnTavoloTC2105
 - AggiungereUnTavoloTC2106
 - AggiungereUnTavoloTC2107
 - AggiungereUnTavoloTC2108
 - AggiungereUnTavoloTC2109
 - AggiungereUnTavoloTC2110
 - AggiungereUnTavoloTC2111
 - AggiungereUnTavoloTC2112
 - AggiungereUnTavoloTC2113
 - AggiungiTavolo
 - LiberaTavoloTC503
 - ModificareUnTavoloTC2201
 - ModificareUnTavoloTC2202
 - ModificareUnTavoloTC2203
 - ModificareUnTavoloTC2204
 - ModificareUnTavoloTC2205
 - ModificareUnTavoloTC2206
 - ModificareUnTavoloTC2207
 - ModificareUnTavoloTC2208
 - ModificareUnTavoloTC2209
 - ModificareUnTavoloTC2210

- OccupiTavoloTC502
- RimuoviTavolo
- VisualizzaNotificaTC505
- FragmentManager
- SuiteTest
- ToastMatcher









Testing con Espresso







```
public class InserireUnPiattoTC3101 {  
    @Rule  
    public ActivityTestRule<LoginActivity> loginActivityActivityTestRule = new ActivityTestRule<>(LoginActivity.class);  
  
    @Before  
    public void setUp() { Intents.init(); }  
  
    @After  
    public void cleanUp() { Intents.release(); }  
  
    @Test  
    public void useAppContext() throws InterruptedException {  
        ##### Login  
        //username  
        Espresso.onView(withId(R.id.username)).perform(typeText( stringToBeTyped: "Tavolo1"));  
        //password  
        Espresso.onView(withId(R.id.password)).perform(typeText( stringToBeTyped: "pass"));  
        Espresso.onView(withId(R.id.bt_login)).perform(click());  
        Thread.sleep( millis: 3000);  
  
        //Controllo che venga caricata l'activity del tavolo  
        intended(hasComponent(TvActivity.class.getName()));  
  
        Espresso.onData(anything()).inAdapterView(allOf(withId(R.id.gv_piatti), isCompletelyDisplayed())).atPosition(0).perform(click());  
        Espresso.onView(withId(R.id.btn_aggiungi)).perform(click());  
        Espresso.onView(withId(R.id.btn_carrello)).perform(click());  
        Espresso.onData(anything()).inAdapterView(allOf(withId(R.id.list_piatti), isCompletelyDisplayed())).atPosition(0).perform(click());  
        Thread.sleep( millis: 1500);  
    }  
}
```

Coverage


JaCoCoverage analysis of project "NetbeansKaazing" (powered by JaCoCo from EcJemma)

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
 query		72%		57%	26	84	109	569	2	47	0	8
 entity		98%		90%	10	153	10	317	1	106	1	7
Total	2 313 of 2 613	86%	133 of 260	72%	36	236	119	886	3	153	1	15

85% classes, 98% lines covered in package 'it.unisa.smartrestaurantapp.entity'

Element	Class, %	Method, %	Line, %
 Account	100% (1/1)	100% (11/11)	100% (21/21)
 Comanda	100% (1/1)	100% (14/14)	100% (27/27)
 Notifica	50% (1/2)	94% (17/18)	97% (33/34)
 Piatto	100% (1/1)	100% (18/18)	100% (41/41)
 PiattoOrdinato	100% (1/1)	100% (17/17)	93% (44/47)
 Tavolo	100% (1/1)	100% (21/21)	100% (44/44)

85% classes, 98% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
 it.unisa.smartrestaurantapp...	85% (6/7)	98% (98/99)	98% (210/214)

Il team



Offertucci Mario



Santarpia Valeria

Grazie per l'attenzione