

TESTING

SINTESI

Con il presente documento si volge all'analisi e alla relativa argomentazione di ogni test automatico considerato all'interno del progetto. I metodi test sono stati raggruppati secondo il Controller in cui essi testano il comportamento dell'applicativo e sono stati segnati secondo uno standard preciso definito con il pattern NomeDelMetodoTestato_RisultatoAspettato.

Per motivi di tempo sono stati implementati casi di test esclusivamente sulla classe ClienteController per il metodo EffettuaPrenotazione. I restanti test sono stati implementati in fase di sviluppo manualmente da ogni membro del gruppo.

• CLIENTE CONTROLLER

afterEffettuaPrenotazioneTest_tornaTemplate

Verifica il corretto funzionamento del metodo, passando alla funzione, nel momento in cui viene richiesto, un iterable di Voli con ad esso associati Aerei aventi disponibilità a prenotazioni.

```
@Test
public void afterEffettuaPrenotazioneTest_tornaTemplate() {

    when(voloRepo.findAll()).thenReturn(Arrays.asList(new Volo(new Long(1), "A", "B", new Date(2323223232L), new Long(11), 3.0, 10)));
    when(aereoRepo.findByCodiceAereo(anyLong())).thenReturn(new Aereo(new Long(11), 3));

    clienteModel = new Cliente("abelo", "alessandro", "belotti", "alebelo@gmail", "123");
    String result = c.effettuaPrenotazione(modelModel, clienteModel);

    assertEquals(result, "/cliente/effettuaPrenotazione");
}
```

afterEffettuaPrenotazioneTest_removeVolo

Viene testato il corretto funzionamento del sistema che in automatico, nel momento della prenotazione dell'ultimo posto su un volo, non rende più disponibile questo volo per prenotazioni future.

```
@Test
public void afterEffettuaPrenotazioneTest_removeVolo() {

    when(voloRepo.findAll()).thenReturn(Arrays.asList(new Volo(new Long(1), "A", "B", new Date(2323223232L), new Long(11), 3.0, 10)));
    when(aereoRepo.findByCodiceAereo(anyLong())).thenReturn(new Aereo(new Long(11), 0));

    Volo v = new Volo();
    v.setCodiceVolo(new Long(2));
    v.setCodiceAereo(new Long(1));
    voloRepo.save(v);

    clienteModel = new Cliente("abelo", "alessandro", "belotti", "alebelo@gmail", "123");
    String result = c.effettuaPrenotazione(modelModel, clienteModel);

    assertEquals(result, "/cliente/error");
}
```

beforeEffettuaPrenotazioneTest_nessunVolo

Nessun volo è stato ancora inserito dagli operatori, il sistema richiamerà quindi la pagina di errore.

```
@Test
public void beforeEffettuaPrenotazioneTest_nessunVolo() {

    // non avendo ancora inserito alcun volo il richiamo della effettuaPrenotazione
    // è inutile perchè non sarà possibile selezionare alcun codiceVolo
    when(voloRepo.findAll()).thenReturn(Arrays.asList());

    Volo v = new Volo();
    v.setCodiceVolo(new Long(2));
    v.setCodiceAereo(new Long(1));
    voloRepo.save(v);

    clienteModel = new Cliente("abelo", "alessandro", "belotti", "alebelo@gmail", "123");
    String result = c.effettuaPrenotazione(modelModel, clienteModel);

    assertEquals(result, "/cliente/error");
}
```

afterEffettuaPrenotazioneTest_esitoAffermativo

Controllo di prenotazione in modo corretto, specificando il Volo selezionato e l'Aereo corrispondente al Volo. Il Cliente è già presente nel database

```
@Test
public void afterEffettuaPrenotazioneTest_esitoAffermativo() {

    when(voloRepo.findByCodiceVolo(anyLong())).thenReturn(new Volo(new Long(1), "A", "B", new Date(2323223232L), new Long(11), 3.0, 10));
    when(aereoRepo.findByCodiceAereo(anyLong())).thenReturn(new Aereo(new Long(11), 10));
    when(clienteRepo.findById(anyString())).thenReturn(Optional.of(new Cliente()));

    Prenotazione p = new Prenotazione();
    p.setCodiceFiscale("CODICEFISCALE");
    p.setCodiceVolo(new Long(1));

    Cliente cliente = new Cliente();
    cliente.setCodiceFiscale("CODICEFISCALE");
    cliente.setNome("NOME");
    cliente.setCognome("COGNOME");

    PrenotazioneDTO pDTO = new PrenotazioneDTO();
    pDTO.setPrenotazione(p);
    pDTO.setCliente(cliente);

    String result = c.postEffettuaPrenotazione(pDTO, modelModel);

    assertEquals(result, "/cliente/esitoPrenotazione");
}
```

afterEffettuaPrenotazioneTest_clienteNonPresente

Test volto a verificare il corretto inserimento di un nuovo Cliente non ancora presente nel database

```
@Test
public void afterEffettuaPrenotazioneTest_clienteNonPresente() {

    when(voloRepo.findByCodiceVolo(anyLong())).thenReturn(new Volo(new Long(1), "A", "B", new Date(2323223232L), new Long(11), 3.0, 10));
    when(aereoRepo.findByCodiceAereo(anyLong())).thenReturn(new Aereo(new Long(11), 10));

    Prenotazione p = new Prenotazione();
    p.setCodiceFiscale("CODICEFISCALE");
    p.setCodiceVolo(new Long(1));

    Cliente cliente = new Cliente();
    cliente.setCodiceFiscale("CODICEFISCALE");
    cliente.setNome("NOME");
    cliente.setCognome("COGNOME");

    PrenotazioneDTO pDTO = new PrenotazioneDTO();
    pDTO.setPrenotazione(p);
    pDTO.setCliente(cliente);

    String result = c.postEffettuaPrenotazione(pDTO, modelModel);

    assertEquals(result, "/cliente/esitoPrenotazione");
}
```

afterEffettuaPrenotazioneTest_voloNonPresente

Verifica che sia stato selezionato un Volo e che non sia stata svolta una POST sospetta, ossia viene richiamata la funzione con un codiceVolo non presente nel database (es con Postman).

```
@Test
public void afterEffettuaPrenotazioneTest_voloNonPresente() {

    Prenotazione p = new Prenotazione();
    p.setCodiceFiscale("CODICEFISCALE");
    p.setCodiceVolo(new Long(1));


    Cliente cliente = new Cliente();
    cliente.setCodiceFiscale("CODICEFISCALE");
    cliente.setNome("NOME");
    cliente.setCognome("COGNOME");













    PrenotazioneDTO pDTO = new PrenotazioneDTO();
    pDTO.setPrenotazione(p);
    pDTO.setCliente(cliente);

    // viene richiamata la postEffettuaPrenotazione con un codiceVolo non presente nel repo:
    // questo capita se la chiamata viene fatto da un tool esterno (Postman)
    String result = c.postEffettuaPrenotazione(pDTO, modelModel);

    assertEquals(result, "/cliente/error");
}
```

Il test di coverage sulla ClienteController ha riguardato la EffettuaPrenotazione e ha dato come esito le seguenti statistiche:



| Element | Coverage |
|-------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| ▼  Aeroporto |  59,7 % |
| ▼  src/main/java |  44,4 % |
| >  com.pack.aeroporto.controller |  40,0 % |
| >  com.pack.aeroporto.entity |  55,6 % |
| >  com.pack.aeroporto.object | 45,0 % |
| >  com.pack.aeroporto | 0,0 % |
| >  src/test/java |  91,0 % |

• **ALTRI TEST**

Gli altri metodi e il corretto funzionamento delle chiamate ai template sono stati testati manualmente in fase di sviluppo dai membri del team.

Riassumendo possiamo vedere che i metodi delle classi controller sono state testate nel seguente modo:

| CLIENTE CONTROLLER | | |
|--------------------------|---------------------------|------------------------|
| effettuaPrenotazione | test automatico + manuale | bug corretto da Tironi |
| controlloStorico | test manuale | esito ok |
| faiLogin | test manuale | esito ok |
| faiRegistrazione | test manuale | esito ok |
| getModificaPrenotazione | test manuale | esito ok |
| getVoli | test manuale | esito ok |
| postModificaPrenotazione | test manuale | esito ok |
| postEffettuaPrenotazione | test manuale | esito ok |

| OPERATORE CONTROLLER | | |
|----------------------|--------------|-------------------------|
| faiLogin | test manuale | esito ok |
| getInserisciVoli | test manuale | esito ok |
| getVisualizzaVolo | test manuale | esito ok |
| postInserisciVolo | test manuale | bug corretto da Belotti |
| getVisualizzaMenu | test manuale | esito ok |