

Università degli Studi di Bergamo(Dalmine)
Corso di Ingegneria Informatica, laurea magistrale

Progetto testing e verifica software: django-ecommerce



Sudati Simone 1045936

Corso: Testing e verifica del software

Docente: Angelo Gargantini

Sessione: Settembre 2020

Indice

1	Applicazione	1
1.1	Setup e Run	1
1.2	Presentazione	2
2	Code testing	7
2.1	unittest	7
2.1.1	Model tests	8
2.1.2	Forms tests	9
2.1.3	Views tests	10
2.2	Test parametrici	11
2.3	Criteri copertura: coverage	12
2.4	MC/DC: Modified condition/Decision coverage	13
2.5	Continuos integration	17
2.6	Selenium	19
2.7	Mock	21
3	Verification	22
3.1	Uso di assertions	22
3.2	Design by contract	24
3.3	Analisi statica	26
3.3.1	MyPy	26
3.3.2	Pylint	27
3.3.3	Flake8	28
3.3.4	Bandit	29
3.3.5	Black	30
3.3.6	Pylama	31
3.3.7	Pyreverse e GraphViz	32
3.4	Eliminazione/giustificazione errori in compilazione	33
3.5	Code refactoring	33
3.6	Integrazione con CI	34
3.7	Code Inspection	37

4 Modeling	39
4.1 Modello ASM	39
4.2 Simulazione/animazione	41
4.3 Scenario validation	43
4.4 Model checking ASMETASMV	50
4.5 Model review & Model advisor	51
5 Model-based testing	54
5.1 Input domain modeling	54
5.2 Combinatorial testing	56
5.3 MBT yakindu	58

Capitolo 1

Applicazione

1.1 Setup e Run

Tool applicazione

- Django web framework
- Linguaggi: Python, HTML, CSS, JavaScript
- Pycharm IDE

Setup applicazione

1. Aprire il terminale
2. Creare ambiente virtuale e attivarlo
3. Cambiare directory: `cd django – ecommerce`
4. Installare le dipendenze: `pip install –r requirements.txt`
5. Run server: `python manage.py runserver`
6. Il deploy del server su <http://127.0.0.1:8000/>

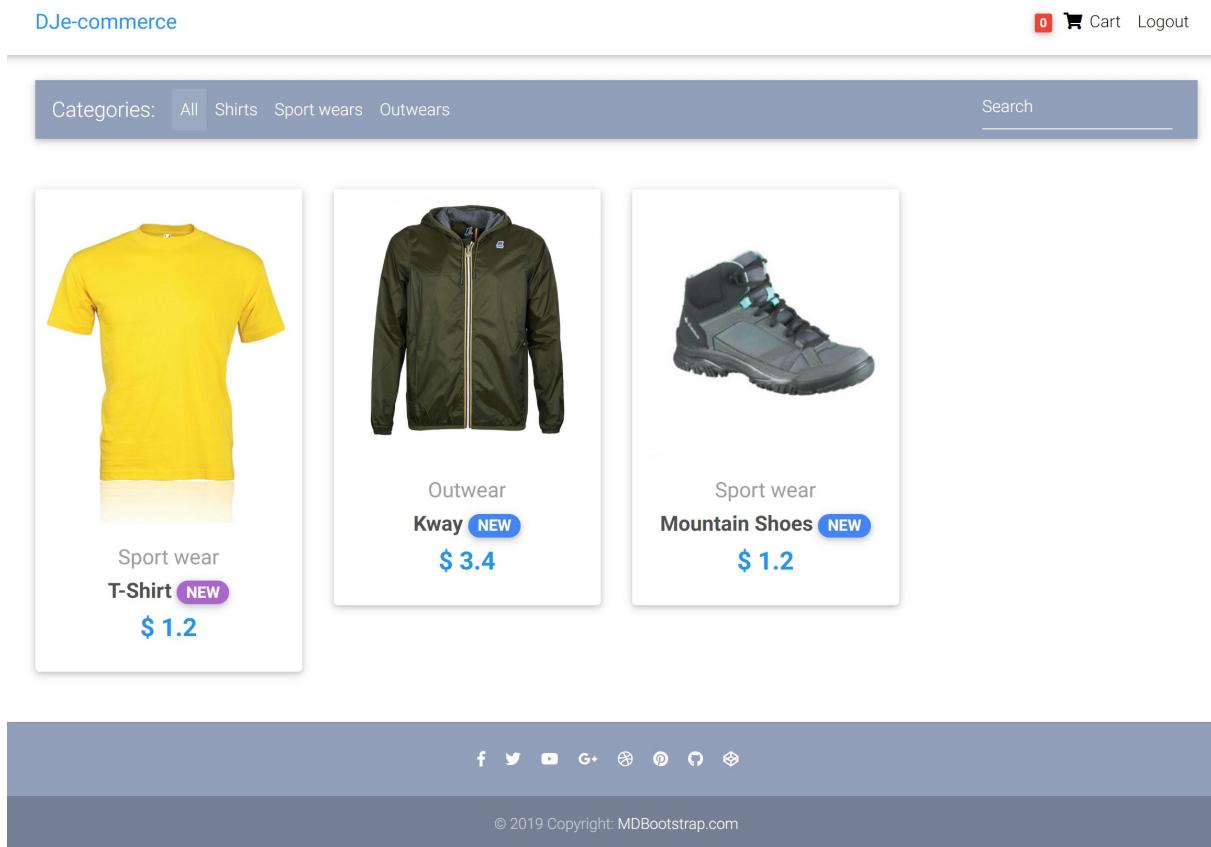
Run applicazione

1. Aprire il terminale
2. Cambiare directory: `cd django – ecommerce`
3. Run server: `python manage.py runserver`
4. Il deploy del server su <http://127.0.0.1:8000/>

1.2 Presentazione

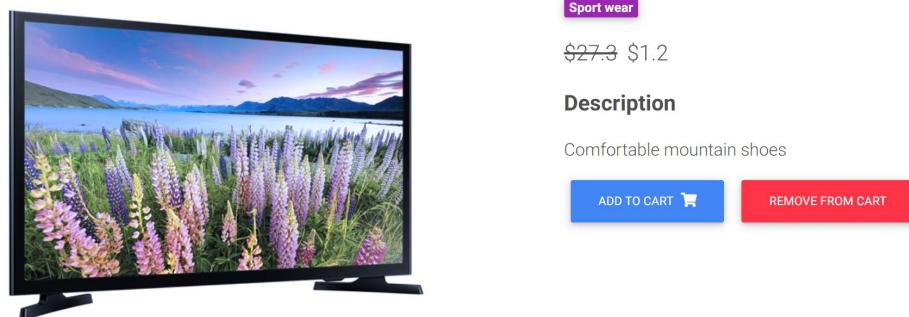
L'applicazione è una web-application di un sito ecommerce scritto in Django.

Questa è la home, la pagina principale



Si fare il get di questo url sia con login che senza. In questa pagina si visualizzano tutti i vari item disponibili suddivisi per 3 categorie : Shirts, Sport wears, Outwears. Cliccando sui prodotti si possono aggiungere in quantità desiderata al carrello.

Il sommario (order summary of item) ancora da implementare in future implemetazioni



Additional information

Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus suscipit modi sapiente illo soluta odit voluptates, quibusdam officia. Neque quibusdam quas a quis porro? Molestias illo neque eum in laborum.



Questa pagina ha l'obiettivo di rappresentare un breve recap del prodotto appena selezionato per decidere se aggiungerlo al carrello o deselectarlo.

Il carrello

The screenshot shows a shopping cart summary page with the following details:

#	Item title	Price	Quantity	Total Item Price
1	Kway	89.3	- 1 +	\$80.0 <small>Saving \$9.29999999999997</small>
2	T-Shirt	27.3	- 2 +	\$40.0 <small>Saving \$14.60000000000001</small>
Coupon			-\$4.0	
Order Total			\$116.0	

At the bottom, there are two buttons: "CONTINUE SHOPPING" (blue) and "PROCEED TO CHECKOUT" (yellow). The page also includes social media sharing icons and a copyright notice: © 2019 Copyright: MDBootstrap.com.

Mostra tutti i prodotti attualmente selezionati. Si può procedere con il pagamento cliccando su "proceed to checkout" oppure tornare alla homepage cliccando su "continue shipping".

La procedura per il pagamento

The screenshot shows a checkout form for a shopping cart. At the top right, there are links for 'DJe-commerce', 'Cart' (with a '2' notification), and 'Logout'. The main title 'Checkout form' is centered above the input fields.

Shipping address: This section contains fields for 'Address' (1234 Main St), 'Address 2 (optional)', 'Apartment or suite', 'Country' (dropdown menu), 'Zip' (text input), and three checkboxes for selecting the same shipping address, saving it as default, or using a specific default address.

Billing address: This section is identical to the shipping address section, containing fields for 'Address' (1234 Main St), 'Address 2 (optional)', 'Apartment or suite', 'Country' (dropdown menu), and 'Zip' (text input).

Your cart: A summary table showing the items in the cart:

Your cart	
1 x Kway Kway	\$80.0
2 x T-Shirt Comfortable mountain shoes	\$40.0
Promo code 1234	-\$4.0
Total (USD)	\$116.0

A 'Promo code' input field and a 'REDEEM' button are located at the bottom of the cart summary.

In questa pagina si inseriscono tutti i dettagli relativi all'indirizzo di spedizione e all'indirizzo di fatturazione. Inoltre nella parte finale del form si seleziona il metodo di pagamento. Nella parte destra del form vi è la possibilità di aggiungere attraverso l'inserimento di un promo code l'ottenimento di sconti.

Django pannello di amministrazione

Django administration

Site administration

ACCOUNTS

Email addresses	Add	Change
-----------------	-----	--------

AUTHENTICATION AND AUTHORIZATION

Groups	Add	Change
Users	Add	Change

CORE

Addresses	Add	Change
Coupons	Add	Change
Items	Add	Change
Order items	Add	Change
Orders	Add	Change
Payments	Add	Change
Refunds	Add	Change
User profiles	Add	Change

SITES

Sites	Add	Change
-------	-----	--------

SOCIAL ACCOUNTS

Social accounts	Add	Change
Social application tokens	Add	Change
Social applications	Add	Change

Recent actions

My actions

- `{self.pk}`
Refund
- Shoes
Item
- scarpa
Item
- Mountain Shoes
Item
- scarpa
Item
- Mountain Shoes
Item
- fsfssf
Item
- fsfssf
Item
- 1234
Coupon
- A shirt
Item

Capitolo 2

Code testing

2.1 unittest

I test sono inseriti nel file `test.py` all'interno dell'app core. Per sua struttura il web framework django predispone un file `test.py` all'interno delle singole app per poter testare la componente. Per runnarli i test bisogna cambiare directory entrando in quella del progetto `cd django – ecommerce` e lanciare il comando `python manage.py test -v 2.`

```
Applying socialaccount.0002_token_max_lengths... OK
Applying socialaccount.0003_extra_data_default_dict... OK
System check identified no issues (0 silenced).
test_valid_forms (core.tests.CouponFormTest) ... ok
test_form (core.tests.CouponFormTestParametrized_0_141234) ... ok
test_form (core.tests.CouponFormTestParametrized_1_12345) ... ok
test_form (core.tests.CouponFormTestParametrized_2_32423) ... ok
test_coupon_creation (core.tests.CouponModelTest) ... ok
test_item_creation (core.tests.ItemModelMockFile) ... ok
test_item_creation (core.tests.ItemModelTest) ... ok
test_valid_form (core.tests.PaymentFormTest) ... ok
test_valid_form (core.tests.RefundFormTest) ... ok
test_form (core.tests.RefundFormTestParametrized_0_141234) ... ok
test_form (core.tests.RefundFormTestParametrized_1_12345) ... ok
test_form (core.tests.RefundFormTestParametrized_2_32423) ... ok
test_mock (core.tests.RefundModelMock) ... ok
test_AddCoupon (core.tests.TestViews) ... ok
test_Checkout (core.tests.TestViews) ... ok
test_OrderSummary (core.tests.TestViews) ... ok
test_OrderSummary2 (core.tests.TestViews) ... ok
test_get_orderSummary (core.tests.TestViews) ... ok
test_miao (core.tests.TestViews) ... ok
test_tryClient (core.tests.TestViews) ... ok

-----
Ran 20 tests in 0.146s

OK
```

Essendo le app di django strutturate secondo MVT ovvero model view template si prestano ad essere testati i models, le views e le forms.

2.1.1 Model tests

```

1 class ItemModelTest(TestCase):
2     def create_item(self, title="K-way", price=28.6, discount_price=5.1,
3                     category='SW', label='M', slug="", description="Hi", image="image.
4                         png"):
5         return Item.objects.create(title=title, price=price, discount_price=
6                         discount_price, category=category, label=label, slug=slug,
7                         description=description, image=image)
8
9     def test_item_creation(self):
10        w = self.create_item()
11        self.assertTrue(isinstance(w, Item))
12
13        fields = w.title, w.price, w.discount_price, w.category,
14        w.label, w.slug, w.description, w.image
15        self.assertEqual(w.__unicode__(), fields)

```

```

1 class CouponModelTest(TestCase):
2     def create_coupon(
3             self,
4             code="13243abcd",
5             amount=10.8
6         ):
7             return Coupon.objects.create(
8                 code=code,
9                 amount=amount,
10            )
11
12     def test_coupon_creation(self):
13        w = self.create_coupon()
14        self.assertTrue(isinstance(w, Coupon))
15
16        fields = (
17            w.code,
18            w.amount
19        )
20        self.assertEqual(w.__unicode__(), fields)

```

2.1.2 Forms tests

```
1 class PaymentFormTest(TestCase):
2     def test_valid_form(self):
3         data = {'stripeToken': "ciao", 'save': 'true', 'use_default': 'false'}
4         form = PaymentForm(data=data)
5         self.assertTrue(form.is_valid())
```

In questo secondo test a seconda se si lascia la "@" nell'email o meno il test risulterà corretto o meno.

```
1 class RefundFormTest(TestCase):
2     def test_valid_form(self):
3         data = {'ref_code': '234245', 'message': 'ciao', 'email': 'simone@mail.com'}
4         form = RefundForm(data=data)
5         self.assertTrue(form.is_valid())
```

```
1 class CouponFormTest(TestCase):
2     def test_valid_form(self):
3         data = {
4             "code": "1234"
5         }
6         form = CouponForm(data=data)
7         self.assertTrue(form.is_valid())
```

2.1.3 Views tests

```
1 def test_OrderSummary(self):
2     url = reverse('core:order-summary')
3     resp = self.client.get(url)
4     self.assertEqual(resp.status_code, 302) #Redirect
```

```
1 def test_Checkout(self):
2     url = reverse('core:checkout')
3     resp = self.client.get(url)
4     self.assertEqual(resp.status_code, 302) #Redirect
```

In add_coupon siccome non è stato ancora impletato totalmente e è negato a un normale user

```
1 def test_AddCoupon(self):
2     url = reverse("core:add-coupon")
3     resp = self.client.get(url)
4     self.assertEqual(resp.status_code, 405) # Method not allowed
```

2.2 Test parametrici

Per svolgerli bisogna importare il seguente package *from parameterized import parameterized, parameterized class*

```

1 @parameterized_class(
2     ("ref_code", "message", "email", "expected_result"),
3     [
4         ("141234", "ciao", "simone@gmail.com", True), #giusta
5         ("12345", "ciao", "simone.com", False),       #senza chiocciola
6         ("32423", "ciao", "simone@gmail", False),    #senza punto
7
8     class RefundFormTestParametrized(TestCase):
9         def test_form(self):
10             data = { 'ref_code': self.ref_code, 'message': self.message,
11                     'email': self.email, }
12             form = RefundForm(data=data)
13             self.assertEqual(form.is_valid(), self.expected_result)

```

```

1 @parameterized_class(
2     ("code", "amount", "expected_result"),
3     [
4         ("141234", 12.3, True),
5         ("12345", 2.9, True),
6         ("32423", 3.7, True),
7     ],
8 )
9
10
11 class CouponFormTestParametrized(TestCase):
12     def test_form(self):
13         data = {
14             "code": self.code,
15             "amount": self.amount,
16         }
17         form = CouponForm(data=data)
18         self.assertEqual(form.is_valid(), self.expected_result)

```

2.3 Criteri copertura: coverage

1. Installazione: `pip install coverage`
2. Run dei test: `coverage run manage.py test -v 2`
3. Genera report output: `coverage report -m`
4. Genera report html: `coverage html`
5. Elimina coperture precedenti: `coverage erase`

Coverage report: 84%

<i>Module ↑</i>	<i>statements</i>	<i>missing</i>	<i>excluded</i>	<i>coverage</i>
FunctionalTesting__init__.py	0	0	0	100%
FunctionalTesting\test_functional.py	13	0	0	100%
core__init__.py	0	0	0	100%
core\admin.py	22	1	0	95%
core\forms.py	29	0	0	100%
core\migrations\0001_initial.py	8	0	0	100%
core\migrations__init__.py	0	0	0	100%
core\models.py	114	26	0	77%
core\templatetags\cart_template_tags.py	9	5	0	44%
core\tests.py	101	6	0	94%
core\urls.py	7	0	0	100%
core\views.py	508	66	0	74%
djecommerce__init__.py	0	0	0	100%
djecommerce\settings__init__.py	0	0	0	100%
djecommerce\settings\base.py	23	0	0	100%
djecommerce\settings\development.py	12	0	0	100%
djecommerce\urls.py	10	3	0	70%
manage.py	9	2	0	78%
Total	865	109	0	84%

coverage.py v5.2.1, created at 2020-08-27 02:57 +0100

2.4 MC/DC: Modified condition/Decision coverage

L'idea chiave sta nel testare solamente le combinazioni più rilevanti e importanti per limitare i costi di testing. Ogni branch attua la propria decisione indipendentemente dalle altre e si vogliono trovare quelle che influenzano da sole e indipendentemente dalle altre l'outcome. Da una tabella che indica tutte le combinazioni da testare (ogni riga con un caso di testing) come prodotto cartesiano tra booleani vogliamo estrarre i casi più rilevanti da testare escludendo i casi non interessanti.

NB: Arriveremo a trovare che date N condizioni avremo $N+1$ test da provare, che sono sicuramente in minor numero degli 2^n casi di test iniziali. La funzione è la compilazione del RefundForm per chiedere il rimborso. Le condizioni sono:

- reasonValid: booleano che controlla che la ragione inserita sia accettabile
- emailValid: booleano che controlla che l'email inserita abbia la struttura corretta
- phoneValid: booleano che controlla che il telefono inserito sia corretto
- accepted: booleano settato dall'utente per far partire la transazione

Questo scenario verrà analizzato più avanti approfonditamente nel capitolo 5.2 combinatorial testing.

L'outcome che vogliamo verificare è che la compilazione del form sia avvenuta nella maniera corretta e che quindi :

accepted and reasonValid and (emailValid or phoneValid) .

Queste sono tutte e 16 le combinazioni e quindi i potenziali casi di test da dover svolgere.

Tests	reasonValid	emailValid	phoneValid	accepted	Outcome
1	T	T	T	T	T
2	T	T	T	F	F
3	T	T	F	T	T
4	T	T	F	F	F
5	T	F	T	T	T
6	T	F	T	F	F
7	T	F	F	T	F
8	T	F	F	F	F
9	F	T	T	T	F
10	F	T	T	F	F
11	F	T	F	T	F
12	F	T	F	F	F
13	F	F	T	T	F
14	F	F	T	F	F
15	F	F	F	T	F
16	F	F	F	F	F

Qui vengono evidenziati i casi di test in cui l'outcome cambia al variare del SOLO parametro reasonValid

Tests	reasonValid	emailValid	phoneValid	accepted	Outcome
1	T	T	T	T	T
2	T	T	T	F	F
3	T	T	F	T	T
4	T	T	F	F	F
5	T	F	T	T	T
6	T	F	T	F	F
7	T	F	F	T	F
8	T	F	F	F	F
9	F	T	T	T	F
10	F	T	T	F	F
11	F	T	F	T	F
12	F	T	F	F	F
13	F	F	T	T	F
14	F	F	T	F	F
15	F	F	F	T	F
16	F	F	F	F	F

Qui vengono evidenziati i casi di test in cui l'outcome cambia al variare del SOLO parametro emailValid

Tests	reasonValid	emailValid	phoneValid	accepted	Outcome	
1	T	T	T	T	T	
2	T	T	T	F	F	
3	T	T	F	T	T	
4	T	T	F	F	F	
5	T	F	T	T	T	
6	T	F	T	F	F	
7	T	F	F	T	F	{3,7}
8	T	F	F	F	F	
9	F	T	T	T	F	
10	F	T	T	F	F	
11	F	T	F	T	F	
12	F	T	F	F	F	
13	F	F	T	T	F	
14	F	F	T	F	F	
15	F	F	F	T	F	
16	F	F	F	F	F	

Qui vengono evidenziati i casi di test in cui l'outcome cambia al variare del SOLO parametro phoneValid

Tests	reasonValid	emailValid	phoneValid	accepted	Outcome	
1	T	T	T	T	T	
2	T	T	T	F	F	
3	T	T	F	T	T	
4	T	T	F	F	F	
5	T	F	T	T	T	
6	T	F	T	F	F	
7	T	F	F	T	F	{5,7}
8	T	F	F	F	F	
9	F	T	T	T	F	
10	F	T	T	F	F	
11	F	T	F	T	F	
12	F	T	F	F	F	
13	F	F	T	T	F	
14	F	F	T	F	F	
15	F	F	F	T	F	
16	F	F	F	F	F	

Qui vengono evidenziati i casi di test in cui l'outcome cambia al variare del SOLO parametro accepted

Tests	reasonValid	emailValid	phoneValid	accepted	Outcome	
1	T	T	T	T	T	
2	T	T	T	F	F	
3	T	T	F	T	T	
4	T	T	F	F	F	{1,2}
5	T	F	T	T	T	{3,4}
6	T	F	T	F	F	{5,6}
7	T	F	F	T	F	
8	T	F	F	F	F	
9	F	T	T	T	F	
10	F	T	T	F	F	
11	F	T	F	T	F	
12	F	T	F	F	F	
13	F	F	T	T	F	
14	F	F	T	F	F	
15	F	F	F	T	F	
16	F	F	F	F	F	

Ora dobbiamo scegliere dei casi di test in modo da coprire almeno una coppia (una relazione) per ogni variabile.

reasonValid	{1,9}	
	{3,11}	
	{5,13}	
emailValid	{3,7}	Casi di test scelti : 3,4,5,7,11
phoneValid	{5,7}	
accepted	{1,2}	
	{3,4}	
	{5,6}	

Scegliendo questi 5 casi test da applicare abbiamo risolto il problema.

2.5 Continuos integration

Per il continuous integration è stato usato GitHub Actions gestito da github e integrato nel repository del progetto stesso. Settando a piacimento il file .yml che si trova .github/workflows/django.yml è possibile installare le dipendenze e eseguire i test ad ogni nuovo push.

The screenshot shows the GitHub Actions page for the repository SudatiSimone / TVSWProject. The repository is private. The Actions tab is selected. A pull request named 'req2' is shown, with the status 'master' and a commit hash 'aef5a32'. On the left, there is a list of workflows: 'Django CI' (on: push), 'build (3.6)' (selected), 'build (3.7)', and 'build (3.8)'. The 'build (3.6)' workflow has a detailed view on the right. The view shows a job named 'Django CI / build (3.6)' that succeeded 1 hour ago in 44s. The steps of the job are listed as follows:

- ▶ ✓ Set up job
- ▶ ✓ Run actions/checkout@v2
- ▶ ✓ Set up Python 3.6
- ▶ ✓ Install Dependencies
- ▶ ✓ Run Tests
- ▶ ✓ Post Run actions/checkout@v2
- ▶ ✓ Complete job

Django CI / build (3.8)
succeeded 1 hour ago in 34s

- ▶ ✓ Set up job
- ▶ ✓ Run actions/checkout@v2
- ▶ ✓ Set up Python 3.8
- ▶ ✓ Install Dependencies
- ▼ ✓ Run Tests
 - 1 ► Run python manage.py test -v 2
 - 6 Creating test database for alias 'default' ('file:memorydb_default?mode=memory&cache=shared')...
 - 7 test_item_creation (core.tests.ItemModelMockFile) ... ok
 - 8 test_item_creation (core.tests.ItemModelTest) ... ok
 - 9 test_valid_form (core.tests.PaymentFormTest) ... ok
 - 10 test_valid_form (core.tests.RefundFormTest) ... ok
 - 11 test_form (core.tests.RefundFormTestParametrized_0_141234) ... ok
 - 12 test_form (core.tests.RefundFormTestParametrized_1_12345) ... ok
 - 13 test_form (core.tests.RefundFormTestParametrized_2_32423) ... ok
 - 14 test_AddCoupon (core.tests.ViewTest) ... ok
 - 15 test_Checkout (core.tests.ViewTest) ... ok
 - 16 test_OrderSummary (core.tests.ViewTest) ... ok
 - 17

I repo badge nel readme segnalano se tutto corretto

README.md

TVSWProject



Applicazione web scritta in django di un sito ecommerce e completa fase di testing e verifica del codice. CI con github actions.

Vedi documentazione nel file "Documentazione.pdf".

2.6 Selenium

Servono due prompt! Per esempio da IDE Pycharm ne puoi gestire due.

1. Installazione: *pip install selenium*
2. Sul primo prompt si lancia il comando *python manage.py runserver*
3. Sul secondo prompt *python manage.py test FunctionalTesting*

Ho simulato un utente che si trova sulla pagina principale del sito. Clicca sul tasto login e entra nella pagina di accesso, inserisce il proprio username e poi la propria password. Clicca sul pulsante accedi e così ha effettuato l'accesso con il proprio account e si trova nella pagine principale.

```

1 class LoginViewSeleniumTest(LiveServerTestCase):
2
3     def test_selenium_login(self):
4
5         #Driver
6         self.driver = webdriver.Chrome('FunctionalTesting/chromedriver.
7             exe')
8
9         # 1) Aprire home page sito
10        self.driver.get("http://localhost:8000/")
11        sleep(3)
12
13        # 2) Cliccare sul tasto "login" della navbar
14        button = self.driver.find_element_by_xpath("//span[contains(text
15            (), 'Login')]")
16        self.driver.execute_script("arguments[0].click()", button)
17
18        # Test che sia nella corretta pagina
19        self.assertIn("http://localhost:8000/accounts/login", self.driver
20            .current_url)
21        sleep(3)
22
23        # 3) Compilare i campi per fare login
24        usernameField = WebDriverWait(self.driver, 10).until(EC.
25            element_to_be_clickable((By.XPATH, "//input[@id='id_login']")))
26
27        passwordField = WebDriverWait(self.driver, 10).until(EC.
28            element_to_be_clickable((By.XPATH, "//input[@id='id_password'])))
29
30        usernameField.send_keys("simone_sudati")
31        passwordField.send_keys("miaomiao")
32        sleep(3)
33
34        # 4) Cliccare il tasto "Sign In"
35        button = self.driver.find_element_by_xpath("//button[@class='btn
36            btn-primary waves-effect waves-light']")

```

```
29         self.driver.execute_script("arguments[0].click()", button)
30
31
32     # Test che sia nella corretta pagina
33     self.assertIn("http://localhost:8000/", self.driver.current_url)
34
35     #chiusura driver
36     sleep(5)
37     self.driver.quit()
```

2.7 Mock

La libreria unittest.mock permette di sostituire parti del sistema con oggetti mock e fare assertions sul loro funzionamento.

```
1 class ItemModelMockFile(TestCase):
2
3     def create_item_image(self, image, title="K-way", price=28.6,
4                         discount_price=5.1, category='SW', label='M', slug="",
5                         description="Hi"):
6         file_mock = mock.MagicMock(spec=File)
7         file_mock.name = image
8         return Item.objects.create(title=title, price=price,
9                         discount_price=discount_price, category=category,
10                        label=label,
11                        slug=slug, description=description,
12                        image=file_mock)
13
14     def test_item_creation(self):
15         w = self.create_item_image("image.png")
16         self.assertTrue(isinstance(w, Item))
17
18         fields = w.title, w.price, w.discount_price, w.category, w.label,
19                     w.slug, w.description, w.image
20         self.assertEqual(w.__unicode__(), fields)
21
22
23 class RefundModelMock(TestCase):
24     @mock.patch('core.models.Item', autospec=True)
25     @mock.patch('core.models.Item.__str__', autospec=True)
26     def test_mock(self, MockItem, MockItemMetodo):
27         item= MockItem
28         MockItemMetodo.return_value= "ciao"
```

Capitolo 3

Verification

3.1 Uso di assertions

Gli asserts sono inseriti all'interno del codice per controllare e confermare che determinati valori/condizioni non si verifichino. Per esempio si controlla che il prezzo totale da pagare sia positivo

```
1  def post(self, *args, **kwargs):
2      order = Order.objects.get(user=self.request.user, ordered=False)
3      form = PaymentForm(self.request.POST)
4      userprofile = UserProfile.objects.get(user=self.request.user)
5      if form.is_valid():
6          token = form.cleaned_data.get('stripeToken')
7          save = form.cleaned_data.get('save')
8          use_default = form.cleaned_data.get('use_default')
9
10     if save:
11         if userprofile.stripe_customer_id != '' and userprofile.
12             stripe_customer_id is not None:
13             customer = stripe.Customer.retrieve(
14                 userprofile.stripe_customer_id)
15             customer.sources.create(source=token)
16
17     else:
18         customer = stripe.Customer.create(
19             email=self.request.user.email,
20         )
21         customer.sources.create(source=token)
22         userprofile.stripe_customer_id = customer['id']
23         userprofile.one_click_purchasing = True
24         userprofile.save()
25
26     amount = int(order.get_total() * 100)
27     assert amount >= 0.0, "il prezzo da pagare non puo' essere
28                         negativo!"
```

In questo esempio prima di aggiungere un ordine si controlla che il nuovo ordine `order_qs[0]` sia effettivamente un'istanza di ordine.

```
1 @login_required
2 def add_to_cart(request, slug):
3     item = get_object_or_404(Item, slug=slug)
4     order_item, created = OrderItem.objects.get_or_create(
5         item=item,
6         user=request.user,
7         ordered=False
8     )
9     order_qs = Order.objects.filter(user=request.user, ordered=False)
10    if order_qs.exists():
11        assert isinstance(
12            order_qs[0], Order
13        )
14        order = order_qs[0]
15        # check if the order item is in the order
16        if order.items.filter(item__slug=item.slug).exists():
17            order_item.quantity += 1
18            order_item.save()
19            messages.info(request, "This item quantity was updated.")
20            return redirect("core:order-summary")
21        else:
22            order.items.add(order_item)
23            messages.info(request, "This item was added to your cart.")
24            return redirect("core:order-summary")
```

3.2 Design by contract

Si usa il package *iContract* che fornisce

- *require* per le precondizioni
- *ensure* per le postcondizioni
- *invariant* per gli invarianti

Data la struttura del programma in cui le funzioni sono tutte views, ho creato delle funzioni ad ora separate dal sito ecommerce ma in futuro integrabili. In modo da avere funzioni strutturate a cui poter applicare per bene le funzionalità del design by contract. La prima funzione calcola il costo di spedizione del prodotto comprato online. Poi ho creato una classe fittizia Prodotto per usare gli invarianti.

```

1
2 @icontract.require(lambda y_ShippingPosition: y_ShippingPosition >0,
3                     " deve essere positiva la coordinata y di spedizione")
4 @icontract.require(lambda x_ShippingPosition: x_ShippingPosition >0,
5                     " deve essere positiva la coordinata x di spedizione")
6 @icontract.require(lambda x_shop: 0 < x_shop < 50,
7                     " indirizzo coordinata x del negozio ")
8 @icontract.require(lambda y_shop: 0 < y_shop < 50,
9                     " deve essere positiva la coordinata x di spedizione")
10 @icontract.ensure(lambda result: 0 < result <= 10,
11                     " il costo dev'essere >0 e al massimo=10 ")
12 def CalculateShipping(x_ShippingPosition: int , y_ShippingPosition: int ,
13                         x_shop: int , y_shop: int ):
14     # posizione del magazzino
15     x_magazzino = 5
16     y_magazzino = 23
17
18     # calcolo della distance
19     distance1 = sqrt(pow((x_shop - x_magazzino) , 2) +
20                       pow((y_shop - y_magazzino) , 2))
21     distance2 = sqrt(pow((x_magazzino - x_ShippingPosition) , 2) +
22                       pow((y_magazzino - y_ShippingPosition) , 2))
23     distance = distance1 + distance2
24
25     # calcola la distanza totale tra magazzino e l'indirizzo di
26     # spedizione
27     costo_al_metro = 0.2
28     full_cost = costo_al_metro * distance
29
30     assert full_cost > 0
31     if full_cost > 10:
32         full_cost = 10
33     elif full_cost < 1:
34         full_cost = 0

```

```
34     return full_cost

---



---

1 @icontract.invariant(lambda self: len(str(self.codice)) == 5)
2 @icontract.invariant(lambda self: self.codice > 0)
3 class Prodotto:
4     def __init__(self, nome, codice, prezzo):
5         self.nome = nome
6         self.codice = codice
7         self.prezzo = prezzo
8
9     def recap(self):
10        return f"Prodotto\nNome:{self.nome}\nCodice:{self.codice}\nPrezzo:{self.prezzo}"

---



---

1 def main():
2     costo1 = CalculateShipping(2, 10, 25, 37)
3     print("Il costo della spedizione e': " + str(costo1))
4
5     # Viola pre-condizione perch y shop e' > 50
6     # costo2 = CalculateShipping(2, 10, 25, 51)
7
8     prodotto1 = Prodotto("shampoo", 12345, 10)
9     print(prodotto1.recap())
10
11    # Viola l'invariante perch fatto da 6 cifre
12    # prodotto2 = Prodotto("shampo", 123456, 25)
13
14
15 if __name__ == "__main__":
16     main()
```

3.3 Analisi statica

3.3.1 MyPy

1. *pip install mypy*
2. Run prima app: *mypy --config setup.cfg core*
3. Run seconda app: *mypy --config setup.cfg djecommerce*

Situazione iniziale

```
core\views.py:10: error: Skipping analyzing 'django.shortcuts': found module but no type hints or library stubs
core\views.py:12: error: Skipping analyzing 'django.utils': found module but no type hints or library stubs
core\views.py:13: error: Skipping analyzing 'django.views.generic': found module but no type hints or library stubs
core\urls.py:1: error: Skipping analyzing 'django.urls': found module but no type hints or library stubs
core\urls.py:2: error: Skipping analyzing 'django.conf': found module but no type hints or library stubs
core\urls.py:3: error: Skipping analyzing 'django.conf.urls.static': found module but no type hints or library stubs
core\urls.py:4: error: Skipping analyzing 'django.contrib': found module but no type hints or library stubs
core\tests.py:4: error: Skipping analyzing 'django.test': found module but no type hints or library stubs
core\tests.py:6: error: Skipping analyzing 'django.urls': found module but no type hints or library stubs
core\tests.py:7: error: Skipping analyzing 'django.utils.text': found module but no type hints or library stubs
core\tests.py:13: error: Skipping analyzing 'django.core.files': found module but no type hints or library stubs
core\tests.py:14: error: Skipping analyzing 'parameterized': found module but no type hints or library stubs
core\tests.py:16: error: Skipping analyzing 'selenium.webdriver.common.keys': found module but no type hints or library stubs
core\tests.py:17: error: Skipping analyzing 'selenium.webdriver.common.keys': found module but no type hints or library stubs
Found 46 errors in 11 files (checked 13 source files)
```

```
C:\Users\susim\Desktop\CloneProgetto\TVSWProject>mypy djecommerce
djecommerce\wsgi.py:3: error: Skipping analyzing 'django.core.wsgi': found module but no type hints or library stubs
djecommerce\urls.py:1: error: Skipping analyzing 'django.conf': found module but no type hints or library stubs
djecommerce\urls.py:2: error: Skipping analyzing 'django.conf.urls.static': found module but no type hints or library stubs
djecommerce\urls.py:3: error: Skipping analyzing 'django.contrib': found module but no type hints or library stubs
djecommerce\urls.py:4: error: Skipping analyzing 'django.urls': found module but no type hints or library stubs
djecommerce\urls.py:13: error: Skipping analyzing 'debug_toolbar': found module but no type hints or library stubs
djecommerce\settings\base.py:2: error: Skipping analyzing 'decouple': found module but no type hints or library stubs
djecommerce\settings\base.py:2: note: See https://mypy.readthedocs.io/en/latest/running\_mypy.html#missing-imports
Found 7 errors in 3 files (checked 7 source files)
```

Situazione finale

```
C:\Users\susim\Desktop\CloneProgetto\TVSWProject>mypy --config setup.cfg core
Success: no issues found in 13 source files
```

```
C:\Users\susim\Desktop\CloneProgetto\TVSWProject>python -m mypy --config setup.cfg djecommerce
Success: no issues found in 7 source files
```

3.3.2 Pylint

1. *pip install pylint*
2. Run prima app: *pylint core*
3. Run seconda app: *pylint --rcfile=.pylintrc ./core*

Ogni volta che si fa il run del comando applicando delle modifiche per togliere gli errori visualizza il nuovo rate ottenuto. Ottenendo un 8.70 finale.

Situazione iniziale

```
***** Module core.views
core\views.py:119:0: C0330: Wrong hanging indentation before block (add 4 spaces).
    [shipping_address1, shipping_country, shipping_zip]
    ^ | (bad-continuation)
core\views.py:189:0: C0330: Wrong hanging indentation before block (add 4 spaces).
    [billing_address1, billing_country, billing_zip]
    ^ | (bad-continuation)
core\views.py:271:0: C0330: Wrong hanging indentation before block (add 4 spaces).
    userprofile.stripe_customer_id is not None
    ^ | (bad-continuation)
core\views.py:272:0: C0330: Wrong hanging indentation before block (add 4 spaces).
    and userprofile.stripe_customer_id != ""
    ^ | (bad-continuation)
core\views.py:1:0: C0114: Missing module docstring (missing-module-docstring)
core\views.py:30:0: C0116: Missing function or method docstring (missing-function-docstring)
core\views.py:36:0: C0116: Missing function or method docstring (missing-function-docstring)
core\views.py:37:24: E1101: Class 'Item' has no 'objects' member (no-member)
core\views.py:41:0: C0116: Missing function or method docstring (missing-function-docstring)
core\views.py:49:0: C0115: Missing class docstring (missing-class-docstring)
```

Your code has been rated at 3.70/10 (previous run: 1.89/10, +1.81)

Situazione finale

```
core\views.py:508:0: W0613: Unused argument 'args' (unused-argument)
core\views.py:508:0: W0613: Unused argument 'kwargs' (unused-argument)
core\views.py:513:4: R1710: Either all return statements in a function should return an expression
core\views.py:513:0: W0613: Unused argument 'args' (unused-argument)
core\views.py:513:0: W0613: Unused argument 'kwargs' (unused-argument)

-----
Your code has been rated at 8.70/10 (previous run: 8.70/10, +0.00)
```

3.3.3 Flake8

1. *pip install flake8*
2. Run prima app: *python -m flake8 core*
3. Run seconda app: *python -m flake8 djecommerce*

Per risolvere i seguenti errori:

- Linee troppo lunghe (> 79 caratteri per riga)
- Moduli importati ma non utilizzati
- Errori nelle indentazioni
- Spazi aggiuntivi superflui

Flake8 su app core

Un solo errore perchè pensa che la variabile sia inutilizzata ma serve

```
C:\Users\susim\Desktop\CloneProgetto\TVSWProject>flake8 core
core\models.py:194:9: F841 local variable 'userprofile' is assigned to but never used
```

3.3.4 Bandit

1. *pip install bandit*
2. Run prima app: *python -m bandit core -r*
3. Run seconda app: *python -m bandit djecommerce -r*

```
Code scanned:
    Total lines of code: 1083
    Total lines skipped (#nosec): 0

Run metrics:
    Total issues (by severity):
        Undefined: 0.0
        Low: 2.0
        Medium: 0.0
        High: 0.0
    Total issues (by confidence):
        Undefined: 0.0
        Low: 0.0
        Medium: 0.0
        High: 2.0
Files skipped (0):
```

```
Test results:
>> Issue: [B101:assert_used] Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.
   Severity: Low   Confidence: High
   Location: core\views.py:272
   More Info: https://bandit.readthedocs.io/en/latest/plugins/b101\_assert\_used.html
271         amount = int(order.get_total() * 100)
272         assert amount >= 0.0, "il prezzo da pagare non può "
273             "essere negativo!"

-----
>> Issue: [B101:assert_used] Use of assert detected. The enclosed code will be removed when compiling to optimised byte code.
   Severity: Low   Confidence: High
   Location: core\views.py:395
   More Info: https://bandit.readthedocs.io/en/latest/plugins/b101\_assert\_used.html
394     if order_qs.exists():
395         assert isinstance(
396             order_qs[0], Order
397         )
398         order = order_qs[0]
```

In cui compaiono due problemi ma che sono semplicemente dei controlli aggiunti con gli assert. Durante il run di bandit quando si incontrano degli assert vengono rimossi in modo da ottimizzare la compilazione del byte code.

3.3.5 Black

1. *pip install black*
2. Run prima app: *python -m black core*
3. Check: *python -m black core --check*
4. Run seconda app: *python -m black djecommerce --check*
5. Max 79 caratteri per linea: *python -m black --line-length 79 core*

Black è un formattatore di codice python. Ottimizza e sistema le indentazioni, spaziature e le formattazioni adeguate.

```
C:\Users\susim\Desktop\CloneProgetto\TVSWProject>black core
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\apps.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\migrations\0004_auto_20190630_1408.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\migrations\0002_auto_20190616_2144.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\migrations\0003_userprofile.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\management\commands\rename.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\admin.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\forms.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\urls.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\tests.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\models.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\migrations\0001_initial.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\core\views.py
All done! ✨♦✨
12 files reformatted, 3 files left unchanged.
```

```
C:\Users\susim\Desktop\CloneProgetto\TVSWProject>black djecommerce
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\djecommerce\wsgi.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\djecommerce\settings\production.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\djecommerce\urls.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\djecommerce\settings\development.py
reformatted C:\Users\susim\Desktop\CloneProgetto\TVSWProject\djecommerce\settings\base.py
All done! ✨♦✨
5 files reformatted, 2 files left unchanged.
```

3.3.6 Pylama

Pylama è un ulteriore linter che ne incorpora al suo interno diversi altri tra i quali pep8, pep257, pyflakes, pylint e mccabe.

1. Installazione di Pyreverse: *pip install pylama*
2. Run con: *pylama module*

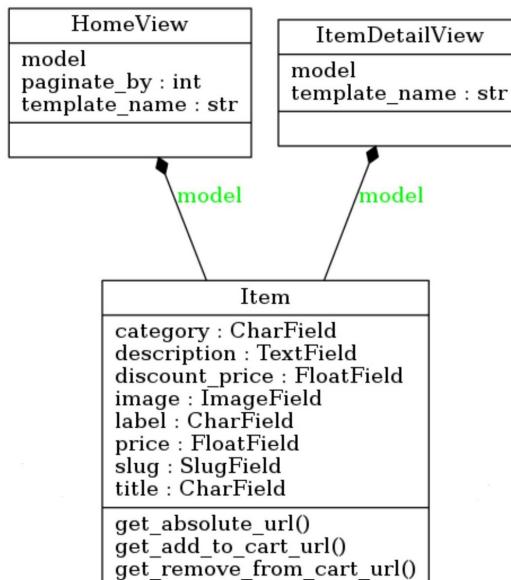
```
core\tests.py:210:13: E225 missing whitespace around operator [pycodestyle]
core\tests.py:210:1: W0612 local variable 'item' is assigned to but never used [pyflakes]
core\tests.py:211:36: E225 missing whitespace around operator [pycodestyle]
core\tests.py:256:1: W391 blank line at end of file [pycodestyle]
core\urls.py:46:6: W292 no newline at end of file [pycodestyle]
core\views.py:82:1: C901 'CheckoutView.post' is too complex (15) [mccabe]
core\views.py:191:25: E125 continuation line with same indent as next logical line [pycodestyle]
core\views.py:261:1: C901 'PaymentView.post' is too complex (14) [mccabe]
core\views.py:274:5: E125 continuation line with same indent as next logical line [pycodestyle]
core\migrations\0001_initial.py:21:80: E501 line too long (114 > 79 characters) [pycodestyle]
```

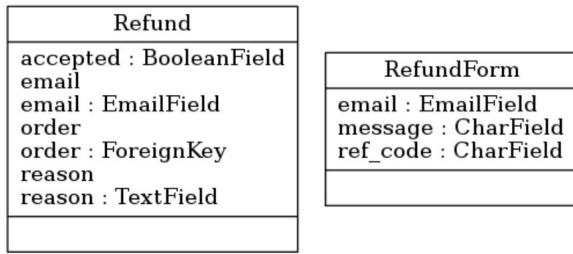
Come si può vedere tra quadre identifica il tool che ha trovato l'errore. Per esempio notiamo [maccabe] o [pycodestyle]

3.3.7 Pyreverse e GraphViz

Usando pyreverse e graphViz si possono generare diagrammi di classe UML delle applicazioni del progetto.

1. Installazione di Pyreverse: `pip install pyreverse`
2. Installazione di GraphViz e settare la variabile di ambiente
3. Generare file .dot: `pyreverse core`
4. Conversione del file .dot in png tramite apposito comando : `dot -Tpng classes.dot -o file.png`
5. (Alternativa) Passo 5 compiuto tramite tool online





3.4 Eliminazione/giustificazione errori in compilazione

Vedi sopra nel capitolo dell’analisi statica 3.3 e in particolare l’utilizzo di mypy.

3.5 Code refactoring

Vedi capitolo analisi statica 3.3 e in particolare l’uso del tool black che è un code formatter.

Si sono sistematate tutte le spaziature e indentazioni, si sono sistematate tutte le lunghezze delle linee contententi più di 79 caratteri che davano errore e anche il renaming delle variabili/funzioni/metodi/classi/test per la case notation. Avendo utilizzato Django di default il template dell’applicazione presenta il file ”test.py”. Visto il numero non eccessivo di test può bastare questo file ma nel caso aumentassero bisognerà separare in più file. Per esempio: test_models.py, test_views.py, test_forms.py per separare i test a seconda se sono indirizzati a testare i models, le views o i forms. Risulta comodo e utile separare i test all’interno di una cartella che contenga tutti i casi di test, con tutti i casi di test (metodi) che hanno struttura ”test_xxxx” in modo che django li riconosca e possano essere eseguiti con un unico comando. Nel mio caso ho deciso di separare solo il test di selenium che ho inserito in una cartella separata dedicata al suo test (visto che necessita anche del driver del browser per testarlo); tenendo invece tutti gli altri casi di test nel file ”test.py” dell’app core che è l’app cardine dell’applicazione. Il motivo è perchè i casi di test sono ben strutturati ma non sono tanti e essendo tutti svolti sull’app core era già perfetto il file ”test.py” presente nell’app core.

3.6 Integrazione con CI

```
Django CI / build (3.8)
succeeded 1 hour ago in 1m 5s

▶ ✓ Set up job
▶ ✓ Run actions/checkout@v2
▶ ✓ Set up Python 3.8
▶ ✓ Install Dependencies
▶ ✓ Run Tests
▶ ✓ Black code formatting
▶ ✓ Black line length
▶ ✓ Bandit
▶ ✓ Flake8
▶ ✓ mypy lint
▶ ✓ pylint
▶ ✓ Post Run actions/checkout@v2
▶ ✓ Complete job
```

Solamente pylint segnerebbe come "rosso" perchè non si sono tolti tutti gli elementi che il linter considera errori ma si è raggiunto un buon punteggio di 8.70 su 10. Quindi si ottiene il 10 su 10 disabilitando alcuni errori di poco conto.

```
▼ ✓ Black code formatting
  1 ► Run python -m black core --check
  6 All done! ✨ 🎉 ✨
  7 15 files would be left unchanged.

▼ ✓ Bandit
  1 ► Run python -m bandit core -r
  6 [main] INFO profile include tests: None
  7 [main] INFO profile exclude tests: None
  8 [main] INFO cli include tests: None
  9 [main] INFO cli exclude tests: None
 10 [main] INFO running on Python 3.7.8
 11 Run started:2020-08-22 08:55:47.353756
 12
 13 Test results:
 14     No issues identified.
 15
 16 Code scanned:
 17     Total lines of code: 1201
 18     Total lines skipped (#nosec): 2
 19
 20 Run metrics:
 21     Total issues (by severity):
 22         Undefined: 0.0
 23         Low: 0.0
 24         Medium: 0.0
 25         High: 0.0
 26     Total issues (by confidence):
 27         Undefined: 0.0
 28         Low: 0.0
 29         Medium: 0.0
 30         High: 0.0
 31 Files skipped (0):
```

```
▼ ✓ Flake8
  1 ► Run flake8 core --ignore F841,W503,E231
  6 Success: no issues found in 10 source files

▼ ✓ mypy lint
  1 ► Run python -m mypy --config setup.cfg core
  6 Success: no issues found in 10 source files

▼ ✓ pylint
  1 ► Run python -m pylint --rcfile=.pylintrc ./core --disable=W
  6
  7 -----
  8 Your code has been rated at 10.00/10
  9
```

I repo badge nel readme segnalano se tutto corretto

README.md

TVSWProject



Applicazione web scritta in django di un sito ecommerce e completa fase di testing e verifica del codice. CI con github actions.

Vedi documentazione nel file "**Documentazione.pdf**".

3.7 Code Inspection

Una checklist contiene una serie di domande che aiutano a identificare i difetti del modulo ispezionato. La check list che verrà utilizzata, visto l'assenza di checklist per python dopo alcune ricerche su internet, è stata presa prendendo spunto da diverse checklist anche di linguaggi differenti come java per essere adattate a python. Il modulo analizzato è core.

PER TUTTI I CASI DI TEST

- The CI jobs on the pull request have passed. Sì, come si può vedere dall'ultimo push ma anche dalla cronologia dei precedenti push.
- It is obvious what the test is trying to test. Sì, già solo dai nomi dei test che sono stati scelti in maniera significativa.
- The test passes when it's supposed to pass. Sì passa nei casi positivi.
- The test fails when it's supposed to fail. Sì fallisce nei casi negativi.
- The test is testing what it thinks it's testing. Sì
- The spec backs up the expected behavior in the test. Sì come da specifiche.
- The test does not use external resources. No, solamente package importati
- The test does not use proprietary features (vendor-prefixed or otherwise). No
- The test is automated as either reftest or a script test unless there's a very good reason for it not to be. Sì il test è automatizzato e dal terminale si possono lanciare tutti i test contemporaneamente o una parte di essi.
- The test does not contain commented-out code. No nessun codice commentato nel test. Ad ora selenium è stato commentato solo per i tempi lunghi di esecuzione ma si può prontamente decommentare.
- The test is placed in the relevant directory. Sì, la directory del test è l'app in questione. L'app in questione che deve essere testata contiene un file "test.py" che si occuperà di contenere tutti i relativi test di quell'app.
- The test has a reasonable and concise filename. Tutti i test dell'app core no perchè sono semplicemente nel file test.py invece selenium sì che si trova nella directory FunctionalTesting.
- If the test needs code running on the server side, the server code must be written in Python, and the Python code must not do anything potentially unsafe. No non utilizza server esterni funziona in locale (localhost).

- If the test needs to be run in some non-standard configuration or needs user interaction, it is a manual test. No è tutto automatizzato.
- The title is descriptive but not too wordy. Sì, si sono scelti titoli efficaci e concisi

Capitolo 4

Modeling

4.1 Modello ASM

Ho creato un modello che simulasse il funzionamento dell'applicazione. Con il modello è possibile loggarsi e accedere al menu principale dalla quale si possono scegliere diversi funzioni. Tra le più importanti ci sono Add_product con la quale è possibile aggiungere un prodotto con la relativa quantità e costo e Cart con la quale si accede al carrello e si possono aggiungere dei prodotti. Qui sotto riporto il codice della signature per elencare i domini utilizzati.

```
1 signature:
2
3     abstract domain Utente
4     domain Prodotto subsetof String
5     domain SubInteger subsetof Integer
6
7     enum domain State = {START | INSERIREID | CHECKID | INSERIREPWD |
8         CHECKPWD | CHOOSESERVICE | CARRELLO | ADD_PRODUCT }
9     enum domain Service = {CART | CHANGEPWS | ADD_PRODUCT_OR_EXIT | EXIT}
10    enum domain Decision = {PAY | CONTINUE }

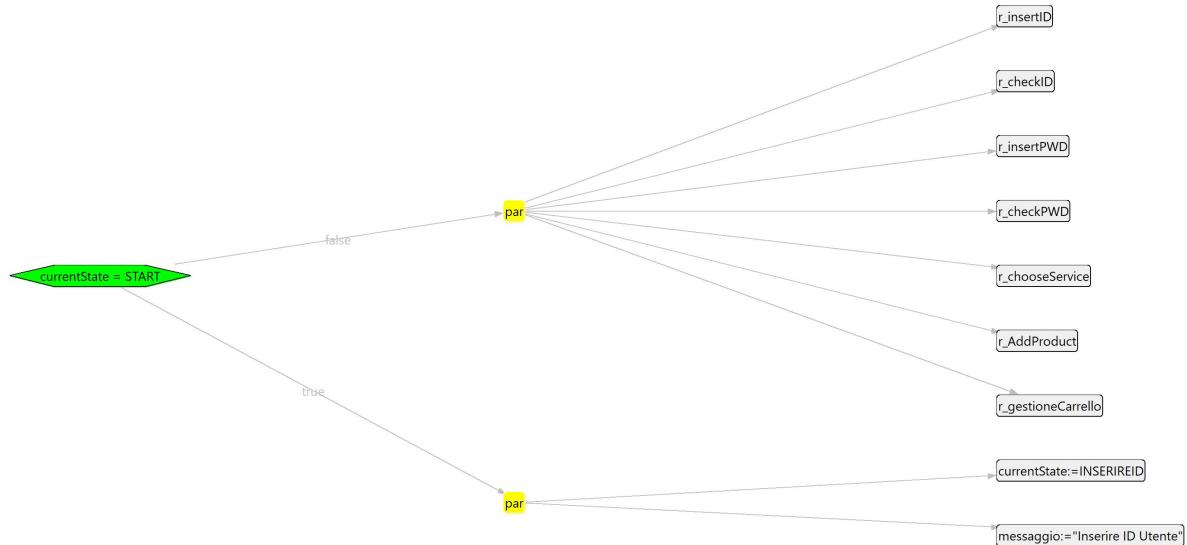
11    dynamic controlled currentState : State
12    dynamic controlled total: SubInteger
13    dynamic controlled messaggio : String
14    dynamic controlled currentUserID : Utente
15    dynamic controlled currentPwd : String
16    dynamic controlled prodotto : Prodotto
17    dynamic controlled currentRFT : Integer
18    dynamic controlled listaProdotti : Utente -> Seq(Prodotto)
19    dynamic controlled numProducts: SubInteger
20
21    dynamic monitored id_utente : Utente
22    dynamic monitored prod : String
23    dynamic monitored password : String
24    dynamic monitored selectedService: Service
25    dynamic monitored decision : Decision
```

```

26      dynamic monitored insertQuantity: SubInteger
27      dynamic monitored insertPrice: SubInteger
28
29
30      derived utentePassword: Utente -> String
31
32      static isValidID: Utente -> Boolean
33      static isValidPwd: String -> Boolean
34      static utente1 : Utente
35      static utente2 : Utente

```

Questa è la struttura del modello asmeta completo poi per poter applicare numSMV verrà semplificato restringendo i domini a quelli permessi.



Nell'immagine soprastante viene rappresentata la semantic visualization delle strutture del programma.

4.2 Simulazione/animazione

Viene rappresentata una simulazione (animazione) completa di uno scenario con avalla

State 0	State 1	State 2	State 3	State 4	State 5
	Inserire ID Utente	Inserire ID Utente	Inserire la Password	Inserire la Password	Di che servizio vuoi usufruire? CARRELLO o SOMMARIO o CHANGEPWS o EXIT
	INSERIREID	CHECKID	INSERIREPWD	CHECKPWD	CHOOSESERVICE
	utente2	utente2	utente2	utente2	utente2
		utente2	utente2	utente2	utente2
			"miao2"	"miao2"	"miao2"
				miao2	miao2
					CART
					"sciarpa"

State 6	State 7	State 8
Quale prodotto vuoi aggiungere al carrello? scarpe, sciarpa, pantaloni o maglia CARRELLO	prodotto aggiunto: sciarpa CHOOSESERVICE	Quale prodotto vuoi aggiungere al carrello? scarpe, sciarpa, pantaloni o maglia CARRELLO
utente2	utente2	utente2
utente2	utente2	utente2
"miao2"	"miao2"	"miao2"
miao2	miao2	miao2
CART	CART	CART
"sciarpa"	"maglia"	"maglia"
sciarpa	sciarpa	maglia
	[sciarpa]	[sciarpa]

State 9	State 10	State 11
prodotto aggiunto: maglia	Questo è il sommario dell'attuale ordine(prodotto selezionato)	Questo è il sommario dell'attuale ordine(prodotto selezionato)
CHOOSESERVICE	ADD_PRODUCT	CHOOSESERVICE
utente2	utente2	utente2
utente2	utente2	utente2
"miao2"	"miao2"	"miao2"
miao2	miao2	miao2
ADD_PRODUCT_OR_EXIT	ADD_PRODUCT_OR_EXIT	ADD_PRODUCT_OR_EXIT
"maglia"	"maglia"	"maglia"
maglia	maglia	maglia
[sciarpa,maglia]	[sciarpa,maglia]	[sciarpa,maglia]
	2	2
	1	1
		2
		1

State 12	State 13	State 14
Questo è il sommario dell'attuale ordine(prodotto selezionato)	Questo è il sommario dell'attuale ordine(prodotto selezionato)	Uscita dal sistema!
ADD_PRODUCT	CHOOSESERVICE	START
utente2	utente2	
utente2	utente2	utente2
"miao2"	"miao2"	
miao2	miao2	miao2
ADD_PRODUCT_OR_EXIT	EXIT	
"maglia"	"maglia"	
maglia	maglia	maglia
[sciarpa,maglia]	[sciarpa,maglia]	[sciarpa,maglia]
1	1	
3	3	
2	5	5
1	2	2

4.3 Scenario validation

La struttura di uno scenario completo è la seguente.

```

1 ** Simulation **
2
3 check succeeded: currentState = START
4 <UpdateSet - 0>
5 currentState=INSERIREID
6 id_utente=utente2
7 messaggio=Inserire ID Utente
8 result=1
9 step_=1
10 </UpdateSet>
11 <State 1 (controlled)>
12 Utente={utente1 ,utente2}
13 currentState=INSERIREID
14 id_utente=utente2
15 messaggio=Inserire ID Utente
16 result=1
17 step_=1
18 </State 1 (controlled)>
19 check succeeded: currentState = INSERIREID
20 <UpdateSet - 1>
21 currentState=CHECKID
22 currentUserID=utente2
23 id_utente=utente2
24 result=1
25 step_=2
26 </UpdateSet>
27 <State 2 (controlled)>
28 Utente={utente1 ,utente2}
29 currentState=CHECKID
30 currentUserID=utente2
31 id_utente=utente2
32 messaggio=Inserire ID Utente
33 result=1
34 step_=2
35 </State 2 (controlled)>
36 check succeeded: currentState = CHECKID
37 check succeeded: currentUserID = utente2
38 <UpdateSet - 2>
39 currentState=INSERIREPWD
40 id_utente=utente2
41 messaggio=Inserire la Password
42 password=miao2
43 result=1
44 step_=3
45 </UpdateSet>
46 <State 3 (controlled)>
47 Utente={utente1 ,utente2}
48 currentState=INSERIREPWD

```

```
49 currentUserID=utente2
50 id_utente=utente2
51 messaggio=Inserire la Password
52 password=miao2
53 result=1
54 step_=3
55 </State 3 (controlled)>
56 check succeeded: currentState = INSERIREPWD
57 check succeeded: currentUserID = utente2
58 <UpdateSet - 3>
59 currentPwd=miao2
60 currentState=CHECKPWD
61 id_utente=utente2
62 password=miao2
63 result=1
64 step_=4
65 </UpdateSet>
66 <State 4 (controlled)>
67 Utente={utente1 , utente2}
68 currentPwd=miao2
69 currentState=CHECKPWD
70 currentUserID=utente2
71 id_utente=utente2
72 messaggio=Inserire la Password
73 password=miao2
74 result=1
75 step_=4
76 </State 4 (controlled)>
77 check succeeded: currentState = CHECKPWD
78 check succeeded: currentUserID = utente2
79 check succeeded: currentPwd = \"miao2\"
80 <UpdateSet - 4>
81 currentState=CHOOSESERVICE
82 id_utente=utente2
83 messaggio=Di che servizio vuoi usufruire? CARRELLO o SOMMARIO o CHANGEPWS
     o EXIT
84 password=miao2
85 result=1
86 selectedService=ADD_PRODUCT_OR_EXIT
87 step_=5
88 </UpdateSet>
89 <State 5 (controlled)>
90 Utente={utente1 , utente2}
91 currentPwd=miao2
92 currentState=CHOOSESERVICE
93 currentUserID=utente2
94 id_utente=utente2
95 messaggio=Di che servizio vuoi usufruire? CARRELLO o SOMMARIO o CHANGEPWS
     o EXIT
96 password=miao2
97 result=1
```

```
98 selectedService=ADD_PRODUCT_OR_EXIT
99 step_-=5
100 </State 5 (controlled)>
101 check succeeded: currentState = CHOOSERVICE
102 check succeeded: currentUserID = utente2
103 check succeeded: currentPwd = \"miao2\"
104 <UpdateSet - 5>
105 currentState=ADD_PRODUCT
106 id_utente=utente2
107 insertPrice=2
108 insertQuantity=1
109 messaggio=Questo      il sommario dell'attuale ordine(prodotto selezionato)
110 password=miao2
111 result=1
112 selectedService=ADD_PRODUCT_OR_EXIT
113 step_-=6
114 </UpdateSet>
115 <State 6 (controlled)>
116 Utente={utente1 , utente2}
117 currentPwd=miao2
118 currentState=ADD_PRODUCT
119 currentUserID=utente2
120 id_utente=utente2
121 insertPrice=2
122 insertQuantity=1
123 messaggio=Questo      il sommario dell'attuale ordine(prodotto selezionato)
124 password=miao2
125 result=1
126 selectedService=ADD_PRODUCT_OR_EXIT
127 step_-=6
128 </State 6 (controlled)>
129 check succeeded: currentState = ADD_PRODUCT
130 check succeeded: currentUserID = utente2
131 check succeeded: currentPwd = \"miao2\"
132 check succeeded: total = 0
133 check succeeded: numProducts = 0
134 <UpdateSet - 6>
135 currentState=CHOOSERVICE
136 id_utente=utente2
137 insertPrice=2
138 insertQuantity=1
139 numProducts=1
140 password=miao2
141 result=1
142 selectedService=ADD_PRODUCT_OR_EXIT
143 step_-=7
144 total=2
145 </UpdateSet>
146 <State 7 (controlled)>
147 Utente={utente1 , utente2}
148 currentPwd=miao2
```

```
149 currentState=CHOOSESERVICE
150 currentUserID=utente2
151 id_utente=utente2
152 insertPrice=2
153 insertQuantity=1
154 messaggio=Questo      il sommario dell'attuale ordine(prodotto selezionato)
155 numProducts=1
156 password=miao2
157 result=1
158 selectedService=ADD_PRODUCT_OR_EXIT
159 step_-=7
160 total=2
161 </State 7 (controlled)>
162 check succeeded: currentState = CHOOSESERVICE
163 check succeeded: currentUserID = utente2
164 check succeeded: currentPwd = \\"miao2\\"
165 check succeeded: total = 2
166 check succeeded: numProducts = 1
167 <UpdateSet - 7>
168 currentState=ADD_PRODUCT
169 id_utente=utente2
170 insertPrice=3
171 insertQuantity=1
172 messaggio=Questo      il sommario dell'attuale ordine(prodotto selezionato)
173 password=miao2
174 result=1
175 selectedService=ADD_PRODUCT_OR_EXIT
176 step_-=8
177 </UpdateSet>
178 <State 8 (controlled)>
179 Utente={utente1 ,utente2}
180 currentPwd=miao2
181 currentState=ADD_PRODUCT
182 currentUserID=utente2
183 id_utente=utente2
184 insertPrice=3
185 insertQuantity=1
186 messaggio=Questo      il sommario dell'attuale ordine(prodotto selezionato)
187 numProducts=1
188 password=miao2
189 result=1
190 selectedService=ADD_PRODUCT_OR_EXIT
191 step_-=8
192 total=2
193 </State 8 (controlled)>
194 check succeeded: currentState = ADD_PRODUCT
195 check succeeded: currentUserID = utente2
196 check succeeded: currentPwd = \\"miao2\\"
197 check succeeded: total = 2
198 check succeeded: numProducts = 1
199 <UpdateSet - 8>
```

```
200 currentState=CHOOSESERVICE
201 id_utente=utente2
202 insertPrice=3
203 insertQuantity=1
204 numProducts=2
205 password=miao2
206 result=1
207 selectedService=EXIT
208 step_=9
209 total=5
210 </UpdateSet>
211 <State 9 (controlled)>
212 Utente={utente1 ,utente2}
213 currentPwd=miao2
214 currentState=CHOOSESERVICE
215 currentUserID=utente2
216 id_utente=utente2
217 insertPrice=3
218 insertQuantity=1
219 messaggio=Questo      il sommario dell'attuale ordine(prodotto selezionato)
220 numProducts=2
221 password=miao2
222 result=1
223 selectedService=EXIT
224 step_=9
225 total=5
226 </State 9 (controlled)>
227 check succeeded: currentState = CHOOSESERVICE
228 check succeeded: currentUserID = utente2
229 check succeeded: currentPwd = \\"miao2\\"
230 check succeeded: total = 5
231 check succeeded: numProducts = 2
232 <UpdateSet - 9>
233 currentState=START
234 messaggio=Uscita dal sistema!
235 result=1
236 step_=10
237 </UpdateSet>
238 <State 10 (controlled)>
239 Utente={utente1 ,utente2}
240 currentPwd=miao2
241 currentState=START
242 currentUserID=utente2
243 id_utente=utente2
244 insertPrice=3
245 insertQuantity=1
246 messaggio=Uscita dal sistema!
247 numProducts=2
248 password=miao2
249 result=1
250 selectedService=EXIT
```

```
251 step_-=10
252 total=5
253 </State 10 (controlled)>
254 check succeeded: currentState = START
255 check succeeded: currentUserID = utente2
256 check succeeded: currentPwd = \\"miao2\\"
257 check succeeded: total = 5
258 check succeeded: numProducts = 2
259 <UpdateSet - 10>
260 currentState=INSERIREID
261 messaggio=Inserire ID Utente
262 result=1
263 step_-=11
264 </UpdateSet>
265 <State 11 (controlled)>
266 Utente={utente1 ,utente2}
267 currentPwd=miao2
268 currentState=INSERIREID
269 currentUserID=utente2
270 id_utente=utente2
271 insertPrice=3
272 insertQuantity=1
273 messaggio=Inserire ID Utente
274 numProducts=2
275 password=miao2
276 result=1
277 selectedService=EXIT
278 step_-=11
279 total=5
280 </State 11 (controlled)>
281 <UpdateSet - 11>
282 </UpdateSet>
283 <State 12 (controlled)>
284 Utente={utente1 ,utente2}
285 currentPwd=miao2
286 currentState=INSERIREID
287 currentUserID=utente2
288 id_utente=utente2
289 insertPrice=3
290 insertQuantity=1
291 messaggio=Inserire ID Utente
292 numProducts=2
293 password=miao2
294 result=1
295 selectedService=EXIT
296 step_-=11
297 total=5
298 </State 12 (controlled)>
299
300 ** Coverage Info: **
301
```

```
302 r_checkID  
303 r_AddProduct  
304 r_chooseService  
305 r_gestioneCarrello  
306 r_checkPWD  
307 r_insertID  
308 r_insertPWD  
309 r_Main
```

4.4 Model checking ASMETASMV

1. Scrittura del codice AsmetaL.
2. Traduzione del codice AsmetaL in un codice NuSMV.
3. Esecuzione del codice NuSMV con il model checker.

NuSMV fornisce controesempi concreti per dimostrare la non validità delle condizioni. Per esempio qua dimostra che il numero di prodotti selezionati può essere maggiore di 2 mostrando gli stati sequenziali che bisogna percorrere.

```
> NuSMV -dynamic -coi -quiet C:\Users\susim\Documents\WorkspaceTesting\ProgettoAsmetaTVSW\ModelloAsmetaSemplificato.smv
-- specification AG numProducts <= 2  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  numProducts = 0
  currentState = CHOOSESERVICE
  selectedService = ADD_PRODUCT_OR_EXIT
-> State: 1.2 <-
  currentState = ADD_PRODUCT
-> State: 1.3 <-
  numProducts = 1
  currentState = CHOOSESERVICE
-> State: 1.4 <-
  currentState = ADD_PRODUCT
-> State: 1.5 <-
  numProducts = 2
  currentState = CHOOSESERVICE
-> State: 1.6 <-
  currentState = ADD_PRODUCT
-> State: 1.7 <-
  numProducts = 3
  currentState = CHOOSESERVICE
-- specification AG total <= 20  is true
```

Questo l'output di NuSMV con le CTL specification scelte.

AsmetaSMV console	Execution of NuSMV code ...
<pre>> NuSMV -dynamic -coi -quiet C:\Users\susim\Documents\WorkspaceTesting\ProgettoAsmetaTVSW\ModelloAsmeta! -- specification AG numProducts <= 3 is true -- specification AG (currentState = OUT -> AG currentState = OUT) is true -- specification EF currentState = OUT is true -- specification AG ((numProducts = 3 & currentState = ADD_PRODUCT) -> AF currentState = OUT) is true -- specification AG ((currentState = DECISION & decision2 = EXIT2) -> AX currentState = OUT) is true -- specification AG (currentState = ADD_PRODUCT -> AX currentState = CHOOSESERVICE) is true -- specification AG total <= 300 is true -- specification EF ((numProducts = 0 & currentState = START) & total = 0) is true</pre>	

4.5 Model review & Model advisor

Si può utilizzare sia come integrato in eclispe sia da terminale con i seguenti comandi.

1. Cambiare directory e andare in quella del progetto
2. Lanciare il comando `java -jar AsmetaMA.jar -mpAll asmetalFileName.asm`

Gli errori individuati al primo run erano i seguenti.

```

1 MP1: No inconsistent update is ever performed
2 NONE
3
4 MP2: Every conditional rule must be complete
5 ConditionalRule if (and(eq(currentState,START),correct)) is not complete.
6 ConditionalRule if (and(and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,CHOOSESERVICE)),eq(selectedService,
    ADD_PRODUCT_OR_EXIT))) is not complete.
7 ConditionalRule if (and(and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,ENTER)),eq(decision1,GO))) is not complete.
8 ConditionalRule if (and(and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,DECISION)),eq(decision2,EXIT2))) is not
    complete.
9 ConditionalRule if (and(and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,ENTER)),eq(decision1,EXIT1))) is not complete
.
10 ConditionalRule if (and(not(and(eq(currentState,START),correct)),not(correct))) is not complete.
11 ConditionalRule if (and(and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,DECISION)),eq(decision2,PROCEDE))) is not
    complete.
12 ConditionalRule if (and(and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,CHOOSESERVICE)),eq(selectedService,EXIT))) is
    not complete.
13 ConditionalRule if (and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,ADD_PRODUCT))) is not complete.
14 ConditionalRule if (and(and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,CHOOSESERVICE)),eq(selectedService,CHANGE_PWS)
    )) is not complete.
15 []
16 []
17 MP2: Every case rule without otherwise must be complete
18 NONE
19 []
20 []
21 MP3: Choose rule is always/sometimes/never not empty
22 NONE
23 []
24 []
25 MP3: Forall rule is always/sometimes/never not empty
26 NONE

```

27
 28 MP3: Conditional rule eval to true
 29 ConditionalRule "if and(and(and(not(and(eq(currentState,START),correct)),
 correct),eq(currentState,DECISION)),eq(decision2,EXIT2)) then endif"
 never executes the then branch
 30 ConditionalRule "if and(and(and(not(and(eq(currentState,START),correct)),
 correct),eq(currentState,DECISION)),eq(decision2,PROCEDE)) then endif"
 never executes the then branch
 31
 32 MP4: No assignment is always trivial
 33 NONE
 34
 35 MP5: For every domain element e, there exists a location which has value
 e
 36 Domain Prodotto is unuseful. All its values are neither used in a domain
 nor in a codomain.
 37 Domain State could be reduced in size. Elements {DECISION, CARRELLO}
 could be removed.
 38
 39 MP6: Every controlled location can take any value in its codomain
 40 Function currentRFT should be removed, because it's never used.
 41 Function enter should be removed, because it's never used.
 42 Function prodotto should be removed, because it's never used.
 43 Function currentState does not take the values {CARRELLO, DECISION} of
 its domain. It could be defined over the smaller domain {ADD_PRODUCT,
 CHOOSESERVICE, ENTER, OUT, START}.
 44 Function numProducts does not take the values {10, 4, 5, 6, 7, 8, 9} of
 its domain. It could be defined over the smaller domain {0, 1, 2, 3}.
 45
 46 MP7: a location could be removed
 47 Controlled location currentRFT is never used. It could be removed.
 48 Controlled location enter is never used. It could be removed.
 49 Controlled location prodotto is never used. It could be removed.
 50 Monitored location prod is never used. It could be removed.
 51
 52 MP7: a controlled location is never updated
 53 Location currentRFT is never updated.
 54 Location enter is never updated.
 55 Location prodotto is never updated.
 56
 57 MP7: a controlled location could be static
 58 NONE

Poi dopo aver sistemato il codice. In cui gli unici errori sono gli MP2 che sono "puramente estetici" e sono delle richieste di inserire degli else per ogni if per esempio inserendo "else skip".

- ¹
 - ² MP1: No inconsistent update is ever performed
 - ³ NONE
 - ⁴ MP2: Every conditional rule must be complete
 - ⁵ ConditionalRule if (and(and(not(and(eq(currentState,START),correct)),correct),not(eq(currentState,ENTER)))) is not complete.
 - ⁶ ConditionalRule if (and(and(not(and(eq(currentState,START),correct)),correct),not(eq(currentState,CHOOSESERVICE)))) is not complete.
 - ⁷ ConditionalRule if (and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,CHOOSESERVICE)),eq(selectedService,CHANGEPWS))) is not complete.
 - ⁸ ConditionalRule if (and(and(not(and(eq(currentState,START),correct)),correct),eq(currentState,CHOOSESERVICE)),eq(selectedService,EXIT))) is not complete.
 - ⁹ ConditionalRule if (and(not(and(eq(currentState,START),correct)),not(correct))) is not complete.
 - ¹⁰ ConditionalRule if (and(not(and(eq(currentState,START),correct)),not(correct)),eq(currentState,ADDPRODUCT))) is not complete.
 - ¹¹ MP2: Every case rule without otherwise must be complete
 - ¹² NONE
 - ¹³ MP3: Choose rule is always/sometimes/never not empty
 - ¹⁴ NONE
 - ¹⁵ MP3: Forall rule is always/sometimes/never not empty
 - ¹⁶ NONE
 - ¹⁷ MP3: Conditional rule eval to true
 - ¹⁸ NONE
 - ¹⁹ MP4: No assignment is always trivial
 - ²⁰ NONE
 - ²¹ MP5: For every domain element e, there exists a location which has value e
 - ²² NONE
 - ²³ MP6: Every controlled location can take any value in its codomain
 - ²⁴ NONE
 - ²⁵ MP7: a location could be removed
 - ²⁶ NONE
 - ²⁷ MP7: a controlled location is never updated
 - ²⁸ NONE
 - ²⁹ MP7: a controlled location could be static
 - ³⁰ NONE
-

Capitolo 5

Model-based testing

5.1 Input domain modeling

Modelliamo l'input del seguente modello refund presente nell'applicazione per poter avere il rimborso sul prodotto.

```
1 class Refund(models.Model):
2     order = models.ForeignKey(Order, on_delete=models.CASCADE)
3     reason = models.TextField()
4     accepted = models.BooleanField(default=False)
5     email = models.EmailField()
6
7     def __str__(self):
8         return "{self.pk}"
```

Usiamo un approccio di tipo Interface-based. In cui consideriamo meccanicamente ogni parametro in maniera isolata ignorando inizialmente le relazioni reciproche. Dobbiamo coprire l'intero dominio con partizioni che non si intersichino. I parametri da analizzare sono reason, accepted e email.

Dominio 1: reason

Il campo reason è un insieme di char e lo separatoremo in 3 partizioni a seconda del numero di caratteri presenti. Il numero di caratteri non può essere negativo infatti non avrebbe senso inserire e non fattibile inserire -5 caratteri per esempio.

Dominio 1	Stringa vuota ""	"Text"	"LongText"
reason	x=0	0 ≤ x ≤ 30	x > 30

Dominio 2: accepted

Il campo accepted è un booleano e può essere settato solamente a true o false.

Dominio 2	Caso positivo	Caso negativo
accepted	x=True	x=False

Dominio 3: email Il campo email per poterlo analizzare in maniera completa ho ipotizzato che abbia la seguente struttura **nomeUtente@dominio.suffisso** come se il suffisso fosse non legato al dominio.

Dominio 3	Stringa vuota ""	"Text"
nomeUtente	x=0	x>0

Dominio 3	Valida	Non valida
@	x=True	x=False

Dominio 3	Stringa vuota ""	"Text"
dominio	x=0	x>0

Dominio 3	.it	.com	altro
suffisso	x=.it	x=.com	x=Error

Ora procediamo con il calcolo di tutte le combinazioni attraverso il prodotto cartesiano.

$$D = D1 * D2 * D3 = 144$$

con

$$D3 = D3_1 * D3_2 * D3_3 * D3_4 = 24$$

Per un totale di 144 combinazioni possibili per il modello refund.

NB: le combinazioni sono calcolate come se fossero enumerativi i char ovvero per esempio in reason Text va tra $0 < x < 30$ e sarebbero 30 valori ma viene considerato un solo valore per comodità.

5.2 Combinatorial testing

La quantità di casistiche trovate con il prodotto cartesiano (per coprire tutti i casi) risulta eccessiva e richiederebbe troppi casi di test. Riusciremo a ridurre il numero di test sfruttando il combinatorial testing e usando dei criteri per non testare tutte le combinazioni possibili ma per esempio analizzando per coppie o triple di valori. Tramite Combinatorial Testing Web EDitor and Generator (CTWEDGE) definiamo i vincoli sui parametri per la generazione di casi di test su refund.

```

1 Model combinatorialTesting
2
3 Parameters:
4
5   reason : {VUOTA, TEXT, LONG_TEXT}
6   accepted : Boolean
7   nomeUtente: [0..12]
8   mailSuffisso : {COM, IT , ERROR}
9   mailDominio : [0..12]
10  mailChiocciola : Boolean
11  reasonValid: Boolean
12  emailValid: Boolean
13  noError: Boolean
14
15 Constraints:
16  # noError => (reasonValid and emailValid )#
17  # accepted = true #
18  # not(reasonValid) <=> reason=VUOTA #
19  # emailValid => ( (nomeUtente>0 and nomeUtente<10) and (mailChiocciola
     ==true) and (mailSuffisso=IT or mailSuffisso=COM) and (mailDominio>0
     and mailDominio<10) ) #

```

La configurazione scelta e il criterio scelto è per coppie N-wise=2.

Name:	ACTS
Time:	0.373
Size:	171
Export	

CTWedge TestSuite

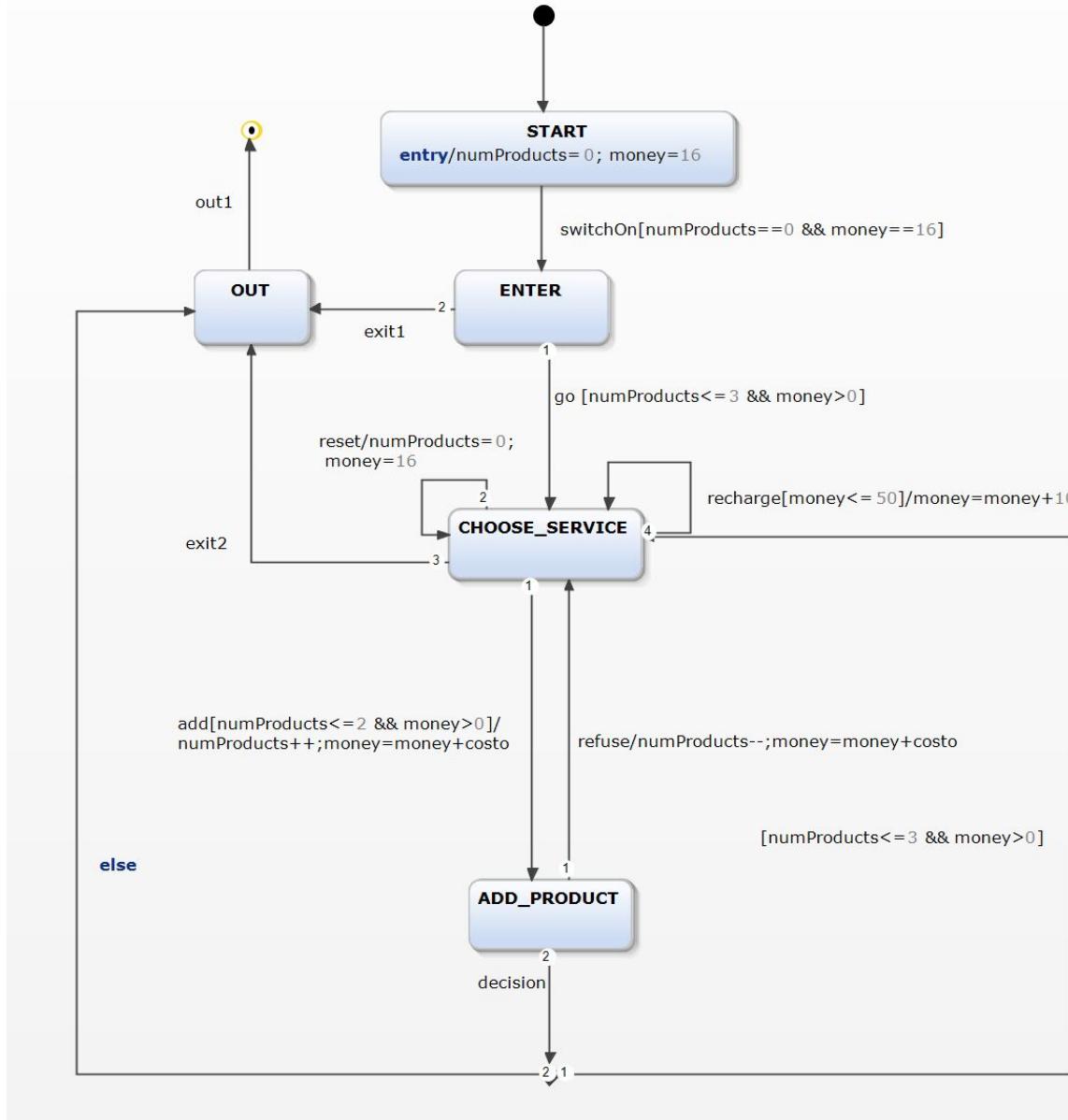
 N-WISE= 2

Test	reasonVal...	reason	mailDomi...	emailValid	mailChioc...	accepted	nomeUte...	mailSu...	noError
1	true	TEXT	0	false	false	true	0	IT	false
2	true	LONG_TEXT	1	false	true	true	0	ERROR	false
3	false	VUOTA	2	false	false	true	0	COM	false
4	true	TEXT	3	false	true	true	0	ERROR	false
5	true	LONG_TEXT	4	false	false	true	0	COM	false
6	false	VUOTA	5	false	true	true	0	IT	false
7	true	TEXT	6	false	true	true	0	COM	false
8	true	LONG_TEXT	7	false	false	true	0	IT	false
9	false	VUOTA	8	false	false	true	0	ERROR	false
10	true	TEXT	9	false	true	true	0	COM	false
11	true	LONG_TEXT	10	false	false	true	0	IT	false
12	false	VUOTA	11	false	true	true	0	ERROR	false
13	true	TEXT	12	false	false	true	0	COM	false
14	true	LONG_TEXT	0	false	true	true	1	ERROR	false
15	false	VUOTA	1	false	false	true	1	IT	false
16	true	TEXT	2	false	true	true	1	ERROR	false
17	true	LONG_TEXT	3	false	false	true	1	COM	false
18	false	VUOTA	4	true	true	true	1	IT	false
19	true	TEXT	5	false	false	true	1	ERROR	false
20	true	LONG_TEXT	6	false	false	true	1	IT	false
21	false	VUOTA	7	true	true	true	1	COM	false
22	true	TEXT	8	true	true	true	1	IT	true
23	true	LONG_TEXT	9	false	false	true	1	ERROR	false
24	false	VUOTA	10	false	true	true	1	COM	false

5.3 MBT yakindu

Modello a macchine a stati finiti

main region



Caso di test

```
1 testclass scenario_base for statechart ModelloEcommerce {  
2  
3     @Test  
4     operation scenario1() {  
5         enter  
6             assert active (main_region.START)  
7             raise switchOn  
8             assert (numProducts ==0)  
9             assert (money == 16)  
10            assert active (main_region.ENTER)  
11            raise go  
12            assert active (main_region.CHOOSERVICE)  
13            raise add  
14            assert active (main_region.ADDPRODUCT)  
15            raise decision  
16            assert active (main_region.CHOOSERVICE)  
17            assert (numProducts ==1)  
18            assert (money == 11)  
19            raise add  
20            assert active (main_region.ADDPRODUCT)  
21            raise decision  
22            assert active (main_region.CHOOSERVICE)  
23            assert (numProducts ==2)  
24            assert (money == 6)  
25            raise recharge  
26            assert active (main_region.CHOOSERVICE)  
27            assert (money == 16)  
28            raise add  
29            assert active (main_region.ADDPRODUCT)  
30            raise decision  
31            assert active (main_region.CHOOSERVICE)  
32            assert (numProducts ==3)  
33            assert (money == 11)  
34            raise exit2  
35            assert active (main_region.OUT)  
36            raise out1  
37        exit  
38    }  
39 }
```

Codice Java del modello

Riporto solamente la struttura dell'interfaccia

```
1  /** Generated by YAKINDU Statechart Tools code generator. */
2  package yakinduproject.modelloecommerce;
3
4  import yakinduproject.IStatemachine;
5
6
7  public interface IModelloEcommerceStatemachine extends IStatemachine {
8      public interface SCInterface {
9
10         public void raiseSwitchOn();
11
12         public void raiseGo();
13
14         public void raiseAdd();
15
16         public void raiseExit1();
17
18         public void raiseExit2();
19
20         public void raiseProcede();
21
22         public void raiseDecision();
23
24         public void raiseReset();
25
26         public void raiseRefuse();
27
28         public void raiseOut1();
29
30         public void raiseRecharge();
31
32         public long getNumProducts();
33
34         public void setNumProducts(long value);
35
36         public long getMoney();
37
38         public void setMoney(long value);
39
40         public long getCosto();
41
42         public void setCosto(long value);
43
44     }
45     public SCInterface getSCInterface();
46
47 }
```
