Winer

Test Summary Report

Versione 1.0.0

Antica enoteca di Roma - Unsplash

Giovanni Prisco
0512106534

Indice

Revision History

Data	Versione	Descrizione
07/02/2022	1.0.0	Creazione Test Summary Report

1. Introduzione

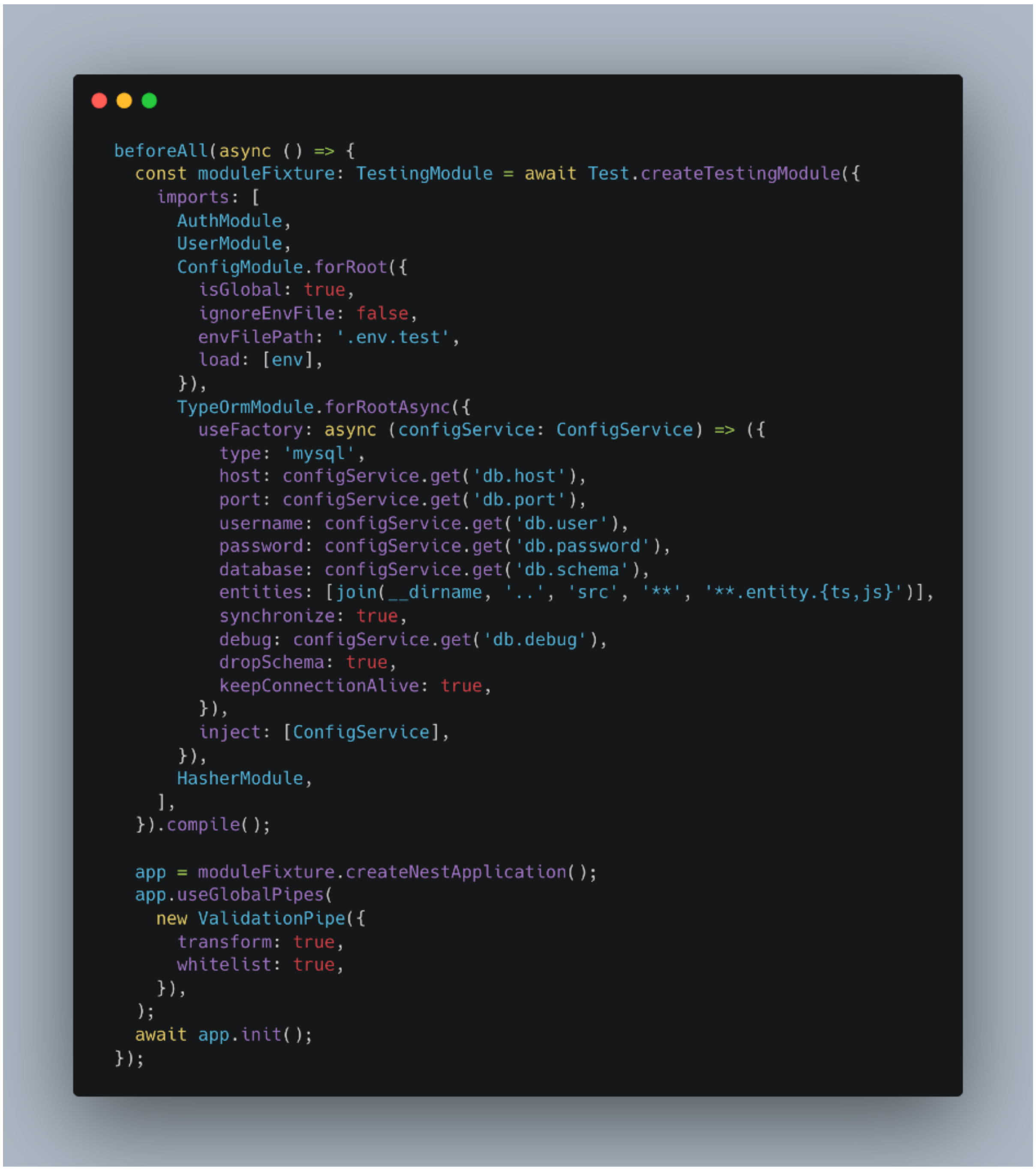
Questo documento si propone di fare un punto della situazione dei vari test implementati.

Grazie allo sviluppo del sistema secondo il dependency injection pattern, ci è stato possibile isolare i sottosistemi per testarli singolarmente andando ad iniettare nel modulo principale solo quelli desiderati o eventuali mock.

Per testare il corretto flusso dell'applicazione si è deciso di non "mockare" le chiamate al database nei test di integrazione, quindi abbiamo configurato le istanze di test affinché si connettessero ad un'istanza di database creata appositamente per eseguire i test.

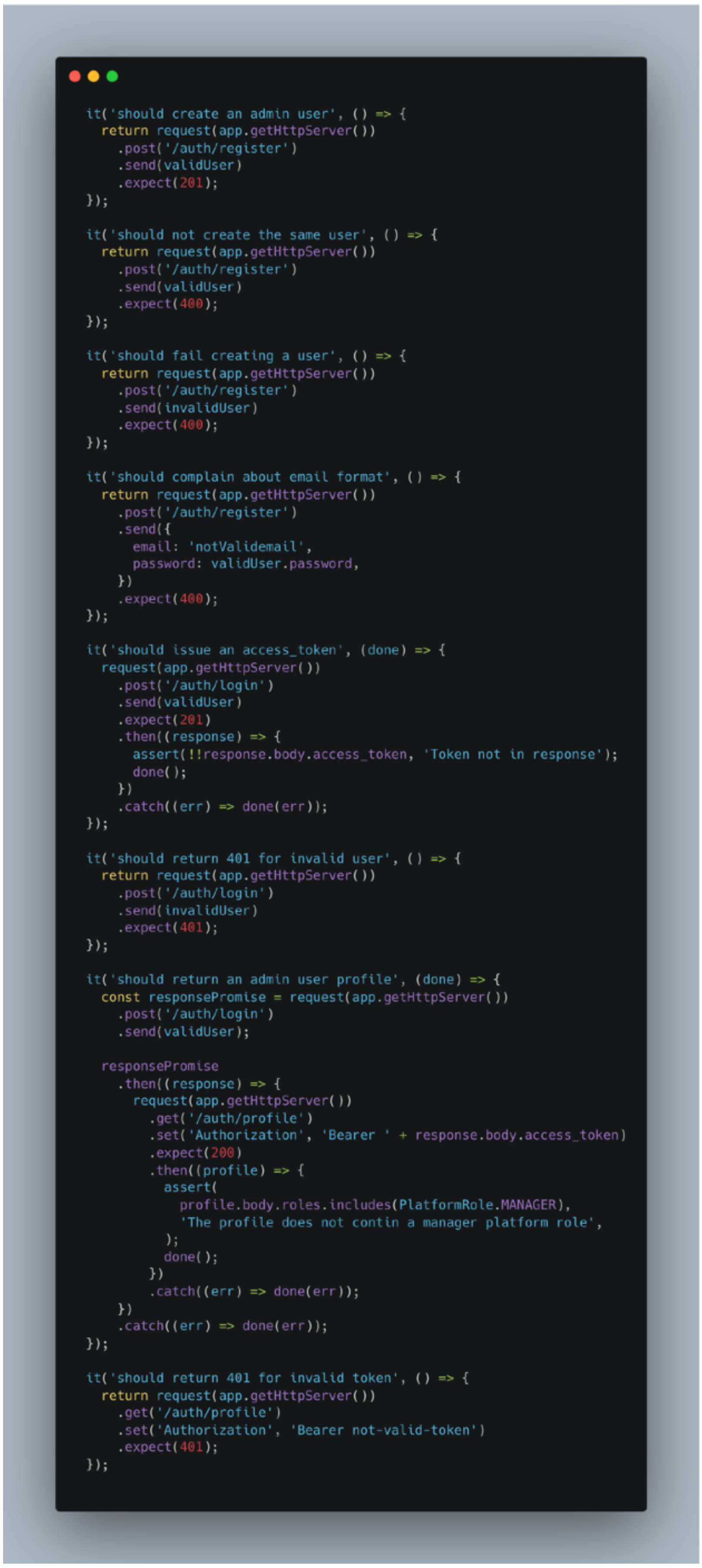
2. Resoconto

2.1 Test Autenticazione



```
beforeAll(async () => {
  const moduleFixture: TestingModule = await Test.createTestingModule({
    imports: [
      AuthModule,
      UserModule,
      ConfigModule.forRoot({
        isGlobal: true,
        ignoreEnvFile: false,
        envFilePath: '.env.test',
        load: [env],
      }),
      TypeOrmModule.forRootAsync({
        useFactory: async (configService: ConfigService) => ({
          type: 'mysql',
          host: configService.get('db.host'),
          port: configService.get('db.port'),
          username: configService.get('db.user'),
          password: configService.get('db.password'),
          database: configService.get('db.schema'),
          entities: [join(__dirname, '..', 'src', '**', '**.entity.{ts,js}')],
          synchronize: true,
          debug: configService.get('db.debug'),
          dropSchema: true,
          keepConnectionAlive: true,
        }),
        inject: [ConfigService],
      }),
      HasherModule,
    ],
  }).compile();

  app = moduleFixture.createNestApplication();
  app.useGlobalPipes(
    new ValidationPipe({
      transform: true,
      whitelist: true,
    }),
  );
  await app.init();
});
```

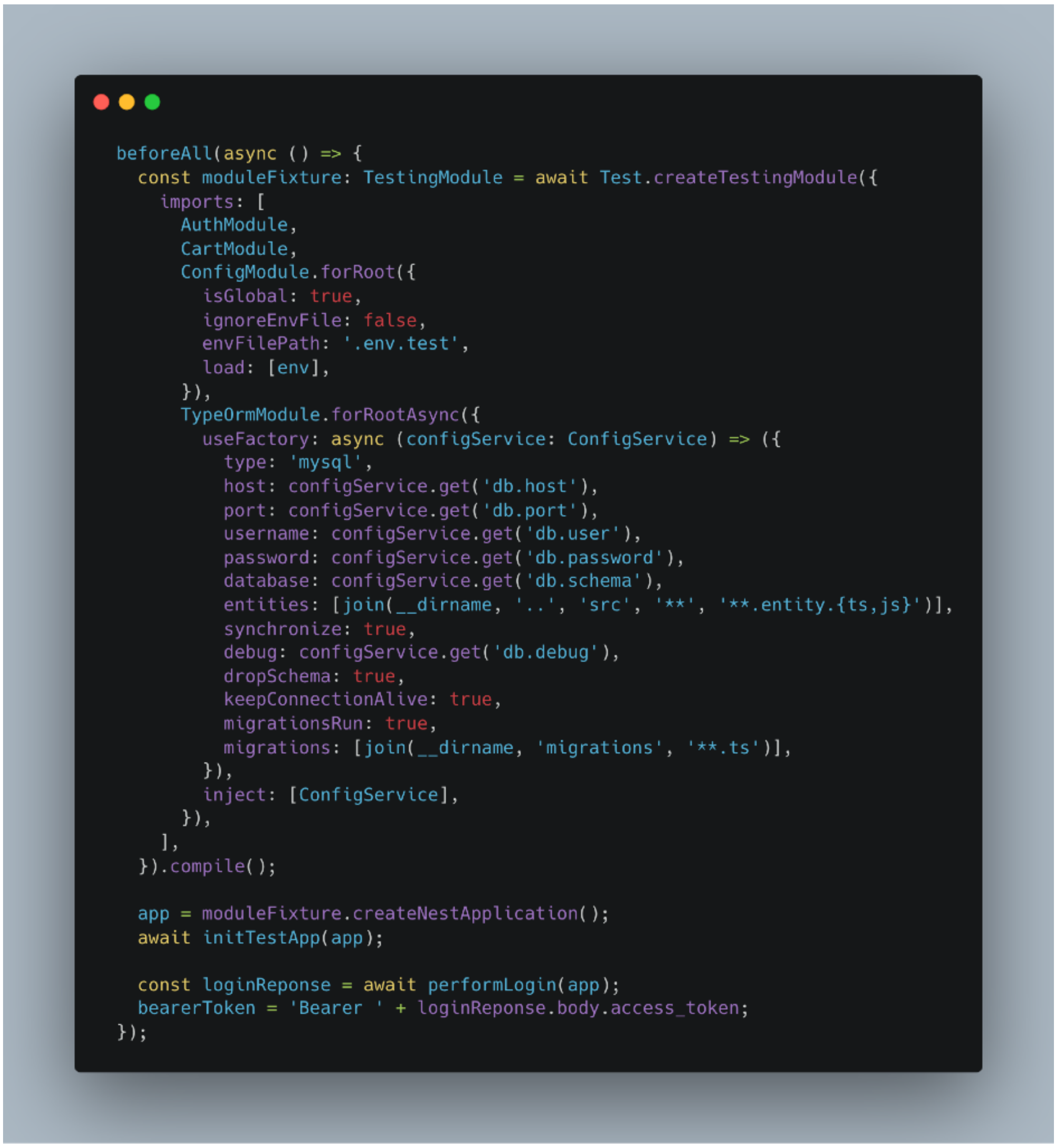
Inizializzazione modulo di test autenticazione



```
it('should create an admin user', () => {
  return request(app.getHttpServer())
    .post('/auth/register')
    .send(validUser)
    .expect(201);
});

it('should not create the same user', () => {
  return request(app.getHttpServer())
    .post('/auth/register')
    .send(validUser)
    .expect(400);
});

it('should fail creating a user', () => {
  return request(app.getHttpServer())
    .post('/auth/register')
    .send(invalidUser)
    .expect(400);
});

it('should complain about email format', () => {
  return request(app.getHttpServer())
    .post('/auth/register')
    .send({
      email: 'notValidemail',
      password: validUser.password,
    })
    .expect(400);
});

it('should issue an access_token', (done) => {
  request(app.getHttpServer())
    .post('/auth/login')
    .send(validUser)
    .expect(201)
    .then((response) => {
      assert(!!response.body.access_token, 'Token not in response');
      done();
    })
    .catch((err) => done(err));
});

it('should return 401 for invalid user', () => {
  return request(app.getHttpServer())
    .post('/auth/login')
    .send(invalidUser)
    .expect(401);
});

it('should return an admin user profile', (done) => {
  const responsePromise = request(app.getHttpServer())
    .post('/auth/login')
    .send(validUser);

  responsePromise
    .then((response) => {
      request(app.getHttpServer())
        .get('/auth/profile')
        .set('Authorization', 'Bearer ' + response.body.access_token)
        .expect(200)
        .then((profile) => {
          assert(
            profile.body.roles.includes(PlatformRole.MANAGER),
            'The profile does not contin a manager platform role',
          );
          done();
        })
        .catch((err) => done(err));
    })
    .catch((err) => done(err));
});

it('should return 401 for invalid token', () => {
  return request(app.getHttpServer())
    .get('/auth/profile')
    .set('Authorization', 'Bearer not-valid-token')
    .expect(401);
});
```

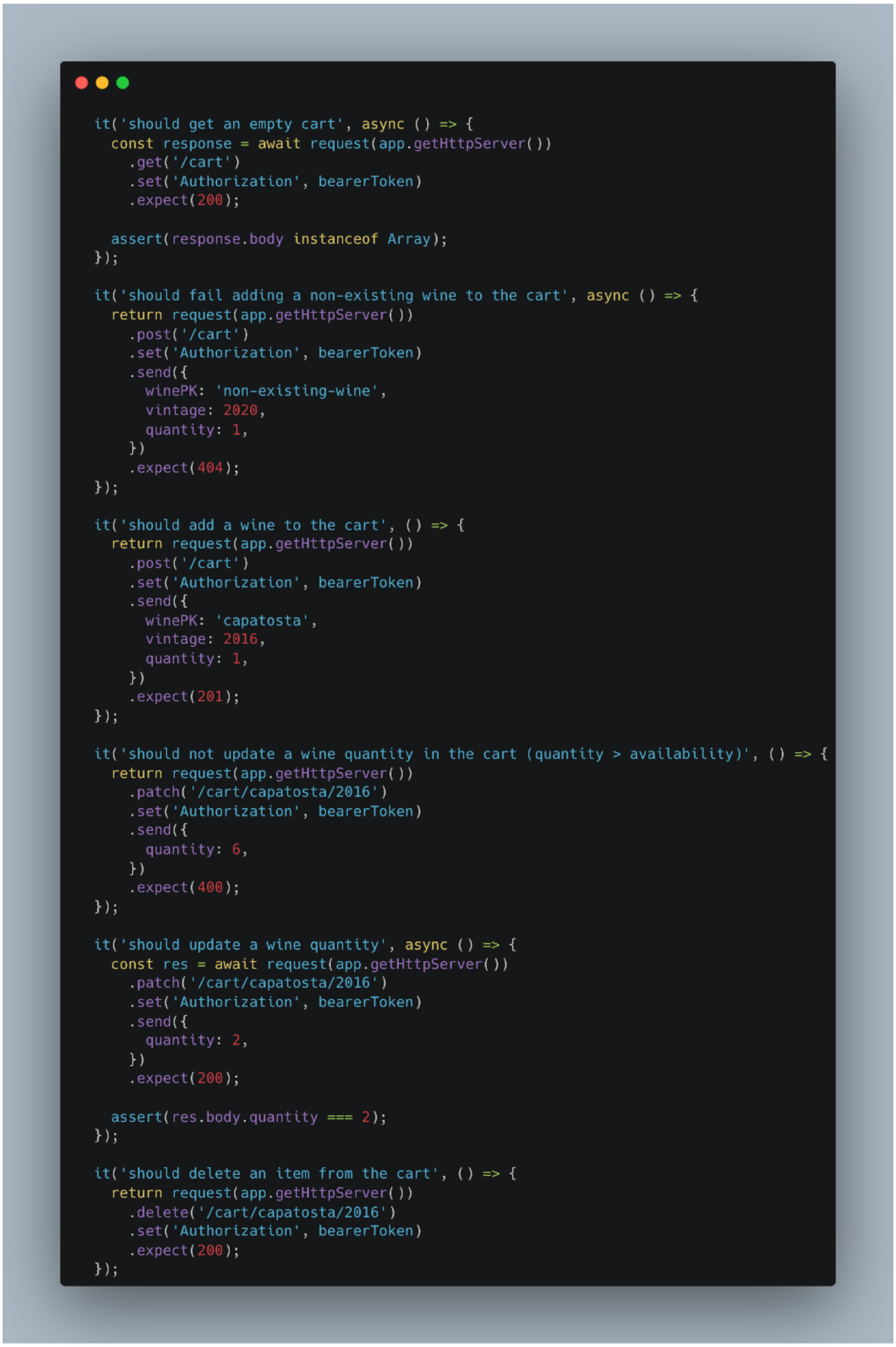
Test Cases Autenticazione

2.2 Test Carrello



```
beforeAll(async () => {
  const moduleFixture: TestingModule = await Test.createTestingModule({
    imports: [
      AuthModule,
      CartModule,
      ConfigModule.forRoot({
        isGlobal: true,
        ignoreEnvFile: false,
        envFilePath: '.env.test',
        load: [env],
      }),
      TypeOrmModule.forRootAsync({
        useFactory: async (configService: ConfigService) => ({
          type: 'mysql',
          host: configService.get('db.host'),
          port: configService.get('db.port'),
          username: configService.get('db.user'),
          password: configService.get('db.password'),
          database: configService.get('db.schema'),
          entities: [join(__dirname, '..', 'src', '**', '**.entity.{ts,js}')],
          synchronize: true,
          debug: configService.get('db.debug'),
          dropSchema: true,
          keepConnectionAlive: true,
          migrationsRun: true,
          migrations: [join(__dirname, 'migrations', '**.ts')],
        }),
        inject: [ConfigService],
      }),
    ],
  }).compile();

  app = moduleFixture.createNestApplication();
  await initTestApp(app);

  const loginReponse = await performLogin(app);
  bearerToken = 'Bearer ' + loginReponse.body.access_token;
});
```

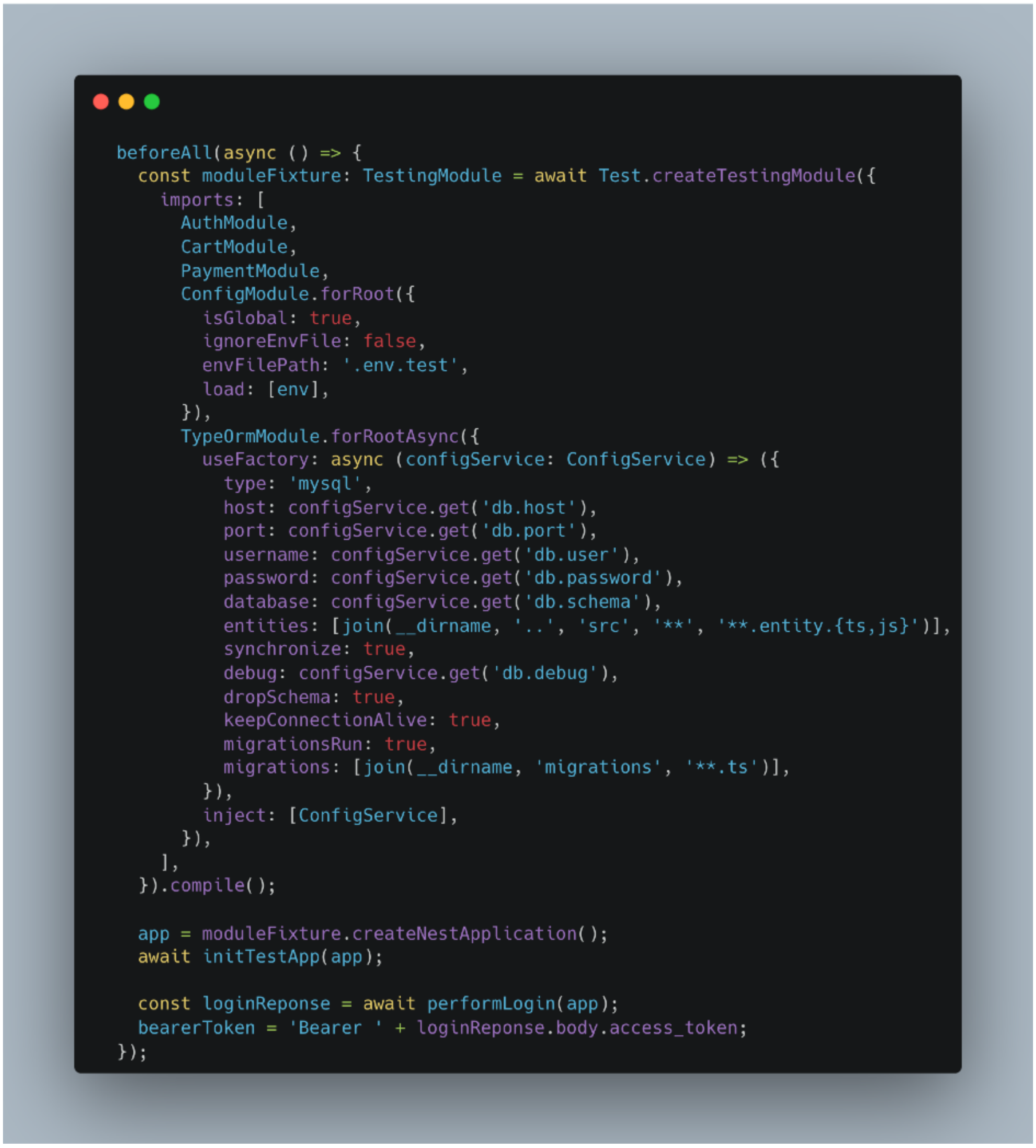
Inizializzazione modulo di test carrello



```
it('should get an empty cart', async () => {
  const response = await request(app.getHttpServer())
    .get('/cart')
    .set('Authorization', bearerToken)
    .expect(200);

  assert(response.body instanceof Array);
});

it('should fail adding a non-existing wine to the cart', async () => {
  return request(app.getHttpServer())
    .post('/cart')
    .set('Authorization', bearerToken)
    .send({
      winePK: 'non-existing-wine',
      vintage: 2020,
      quantity: 1,
    })
    .expect(404);
});

it('should add a wine to the cart', () => {
  return request(app.getHttpServer())
    .post('/cart')
    .set('Authorization', bearerToken)
    .send({
      winePK: 'capatosta',
      vintage: 2016,
      quantity: 1,
    })
    .expect(201);
});

it('should not update a wine quantity in the cart (quantity > availability)', () => {
  return request(app.getHttpServer())
    .patch('/cart/capatosta/2016')
    .set('Authorization', bearerToken)
    .send({
      quantity: 6,
    })
    .expect(400);
});

it('should update a wine quantity', async () => {
  const res = await request(app.getHttpServer())
    .patch('/cart/capatosta/2016')
    .set('Authorization', bearerToken)
    .send({
      quantity: 2,
    })
    .expect(200);

  assert(res.body.quantity === 2);
});

it('should delete an item from the cart', () => {
  return request(app.getHttpServer())
    .delete('/cart/capatosta/2016')
    .set('Authorization', bearerToken)
    .expect(200);
});
```

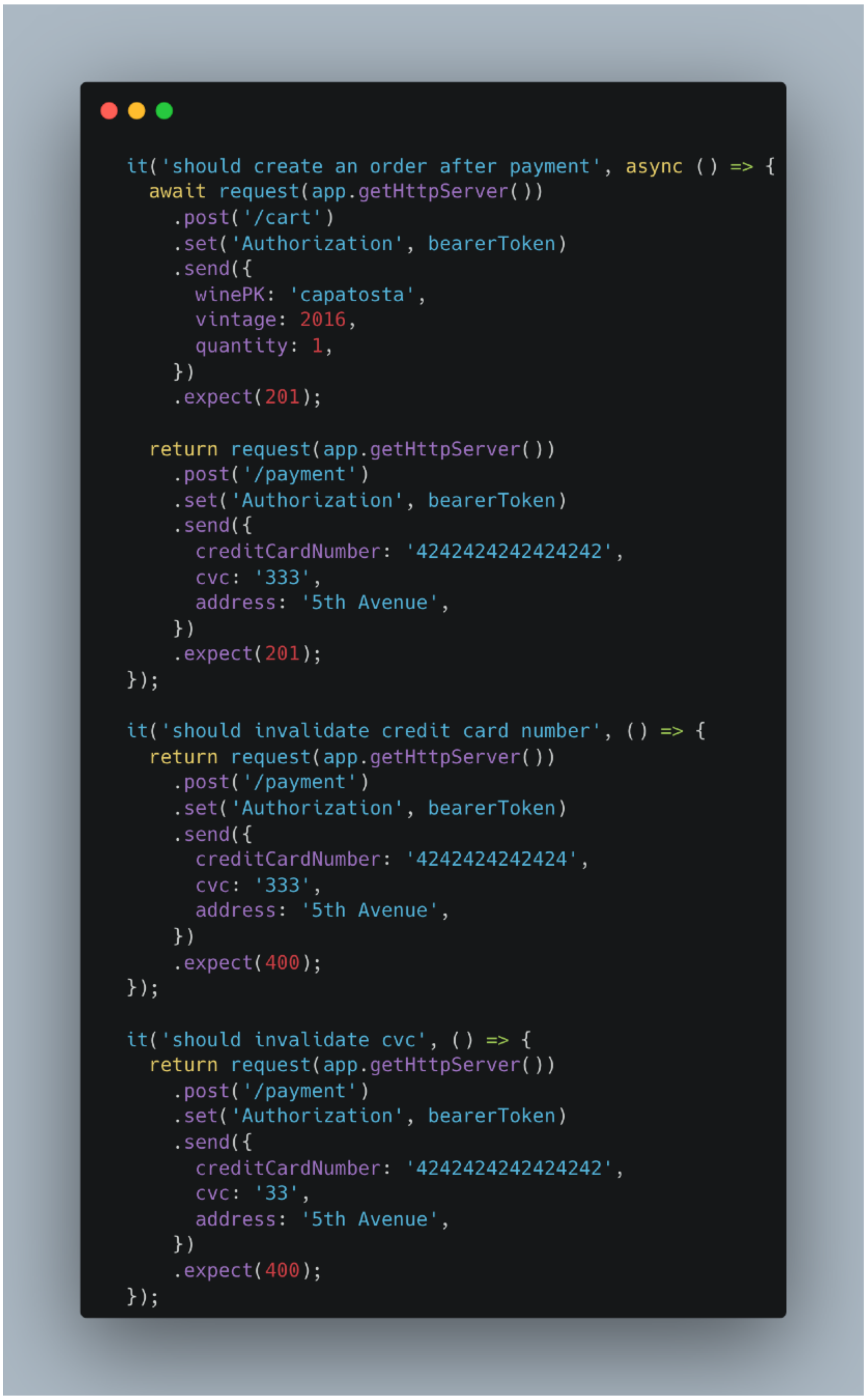
Test Cases Carrello

2.3 Test Pagamenti



```
beforeAll(async () => {
  const moduleFixture: TestingModule = await Test.createTestingModule({
    imports: [
      AuthModule,
      CartModule,
      PaymentModule,
      ConfigModule.forRoot({
        isGlobal: true,
        ignoreEnvFile: false,
        envFilePath: '.env.test',
        load: [env],
      }),
      TypeOrmModule.forRootAsync({
        useFactory: async (configService: ConfigService) => ({
          type: 'mysql',
          host: configService.get('db.host'),
          port: configService.get('db.port'),
          username: configService.get('db.user'),
          password: configService.get('db.password'),
          database: configService.get('db.schema'),
          entities: [join(__dirname, '..', 'src', '**', '**.entity.{ts,js}')],
          synchronize: true,
          debug: configService.get('db.debug'),
          dropSchema: true,
          keepConnectionAlive: true,
          migrationsRun: true,
          migrations: [join(__dirname, 'migrations', '**.ts')],
        }),
        inject: [ConfigService],
      }),
    ],
  }).compile();

  app = moduleFixture.createNestApplication();
  await initTestApp(app);

  const loginReponse = await performLogin(app);
  bearerToken = 'Bearer ' + loginReponse.body.access_token;
});
```

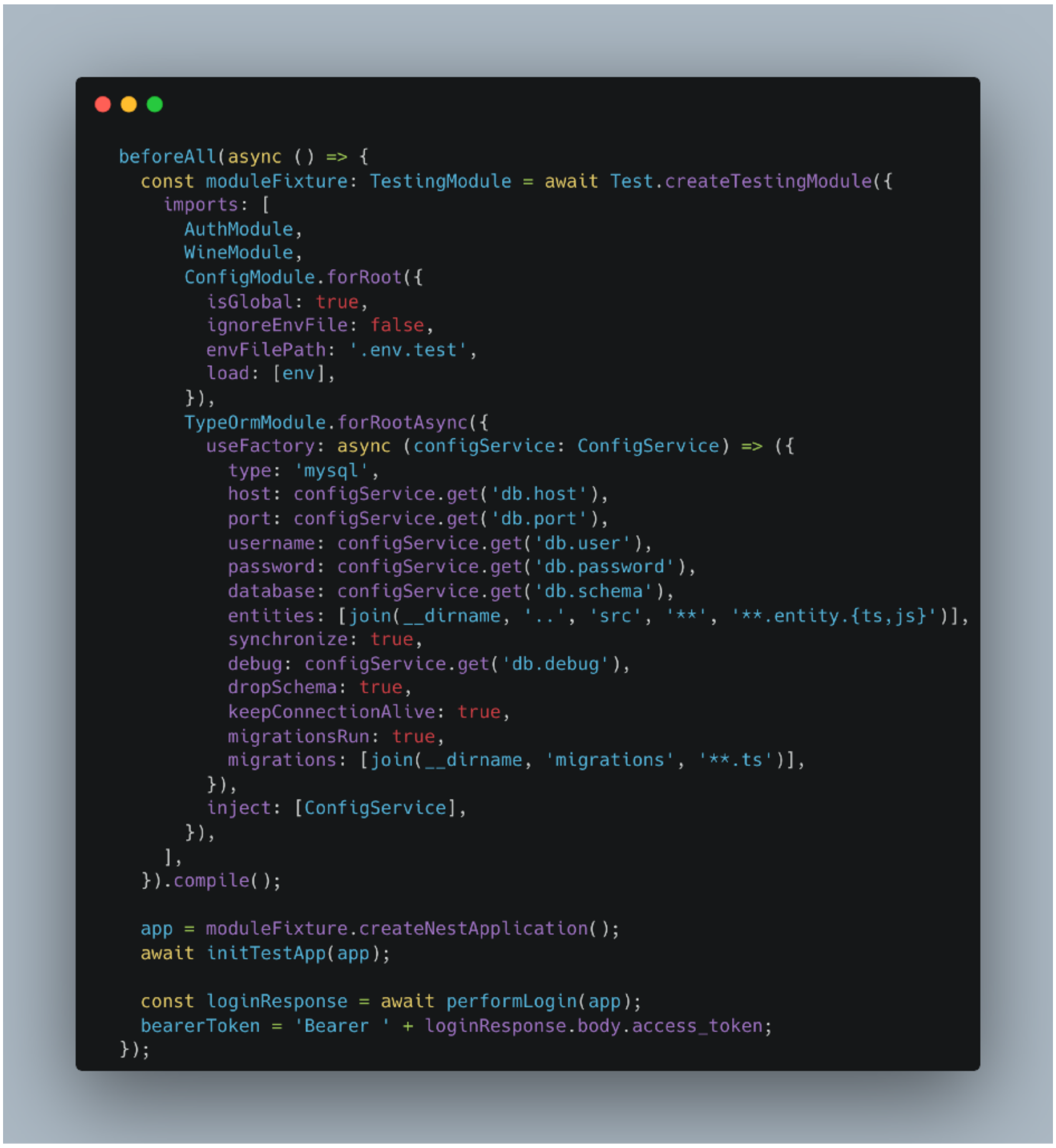
Inizializzazione modulo di test pagamenti



```
it('should create an order after payment', async () => {
  await request(app.getHttpServer())
    .post('/cart')
    .set('Authorization', bearerToken)
    .send({
      winePK: 'capatosta',
      vintage: 2016,
      quantity: 1,
    })
    .expect(201);

  return request(app.getHttpServer())
    .post('/payment')
    .set('Authorization', bearerToken)
    .send({
      creditCardNumber: '4242424242424242',
      cvc: '333',
      address: '5th Avenue',
    })
    .expect(201);
});

it('should invalidate credit card number', () => {
  return request(app.getHttpServer())
    .post('/payment')
    .set('Authorization', bearerToken)
    .send({
      creditCardNumber: '4242424242424',
      cvc: '333',
      address: '5th Avenue',
    })
    .expect(400);
});

it('should invalidate cvc', () => {
  return request(app.getHttpServer())
    .post('/payment')
    .set('Authorization', bearerToken)
    .send({
      creditCardNumber: '4242424242424242',
      cvc: '33',
      address: '5th Avenue',
    })
    .expect(400);
});
```

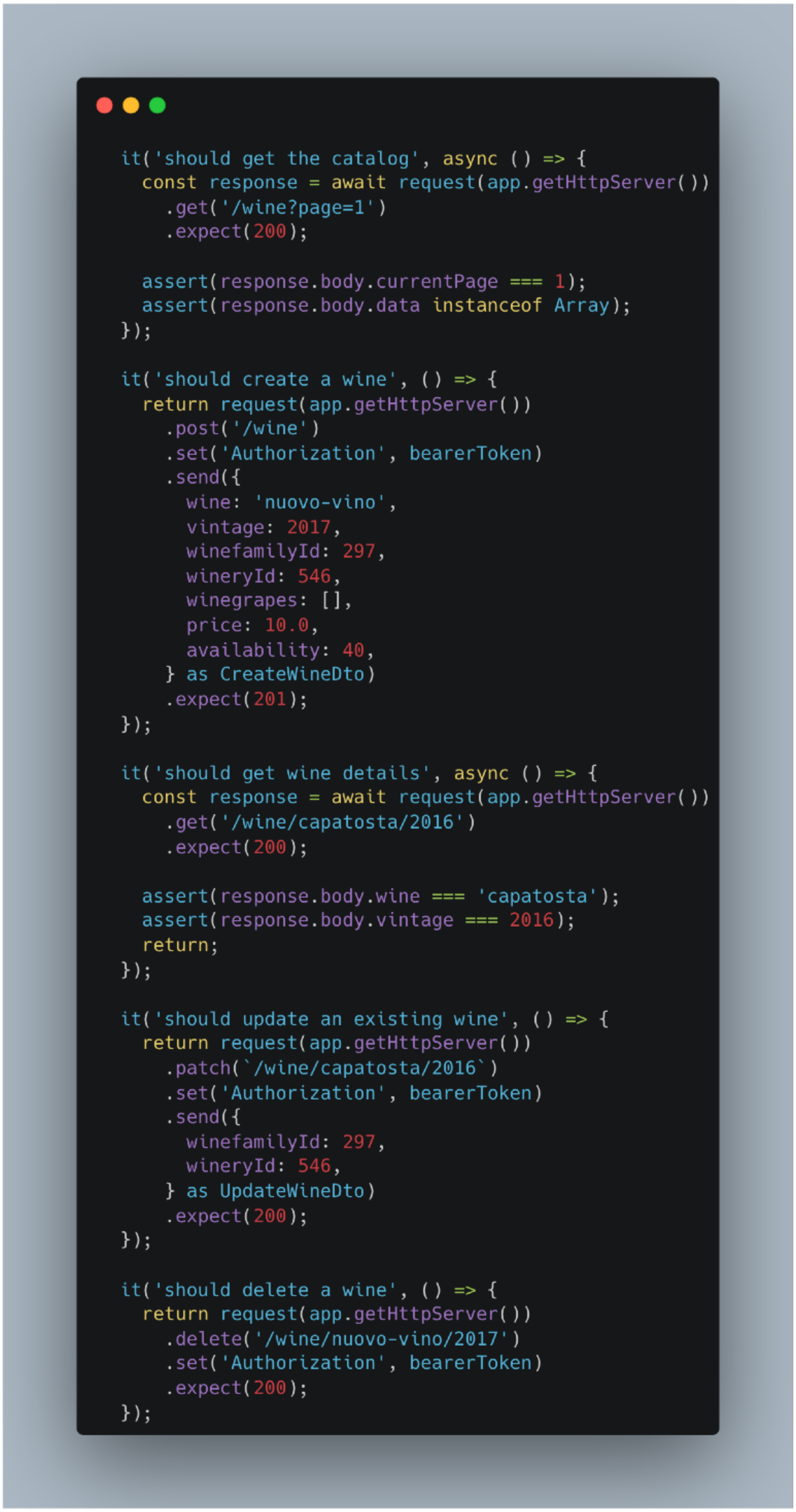
Test Cases Pagamenti

2.4 Test Catalogo



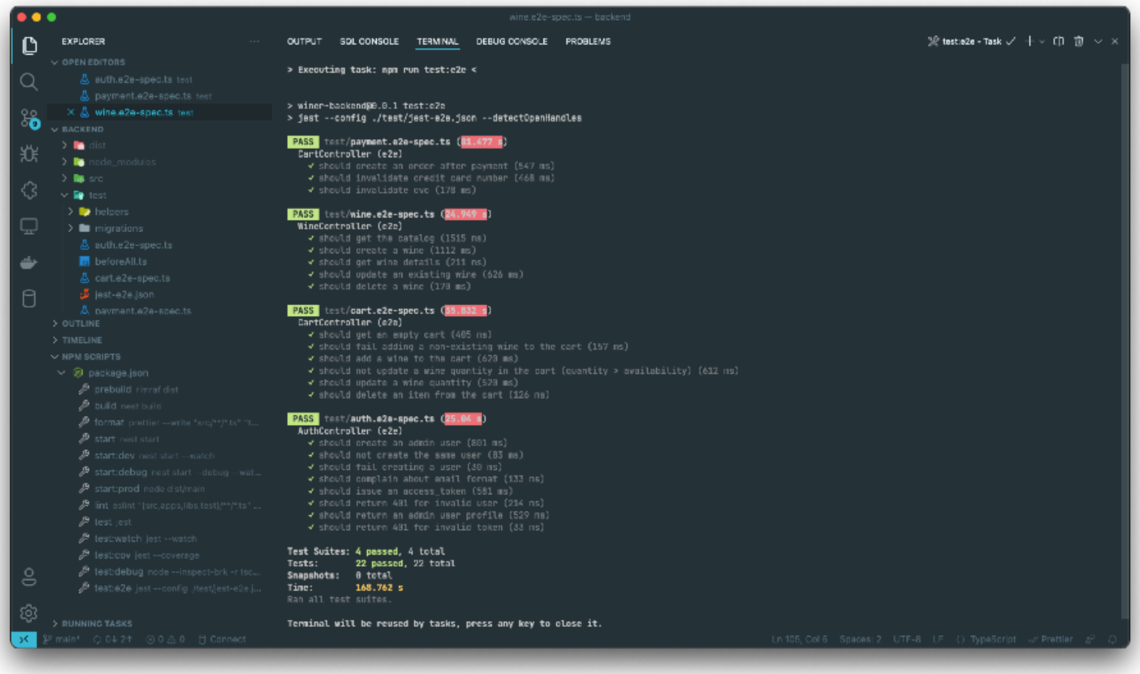
```
beforeAll(async () => {
  const moduleFixture: TestingModule = await Test.createTestingModule({
    imports: [
      AuthModule,
      WineModule,
      ConfigModule.forRoot({
        isGlobal: true,
        ignoreEnvFile: false,
        envFilePath: '.env.test',
        load: [env],
      }),
      TypeOrmModule.forRootAsync({
        useFactory: async (configService: ConfigService) => ({
          type: 'mysql',
          host: configService.get('db.host'),
          port: configService.get('db.port'),
          username: configService.get('db.user'),
          password: configService.get('db.password'),
          database: configService.get('db.schema'),
          entities: [join(__dirname, '..', 'src', '**', '**.entity.{ts,js}')],
          synchronize: true,
          debug: configService.get('db.debug'),
          dropSchema: true,
          keepConnectionAlive: true,
          migrationsRun: true,
          migrations: [join(__dirname, 'migrations', '**.ts')],
        }),
        inject: [ConfigService],
      }),
    ],
  }).compile();

  app = moduleFixture.createNestApplication();
  await initTestApp(app);

  const loginResponse = await performLogin(app);
  bearerToken = 'Bearer ' + loginResponse.body.access_token;
});
```

Inizializzazione modulo di test catalogo



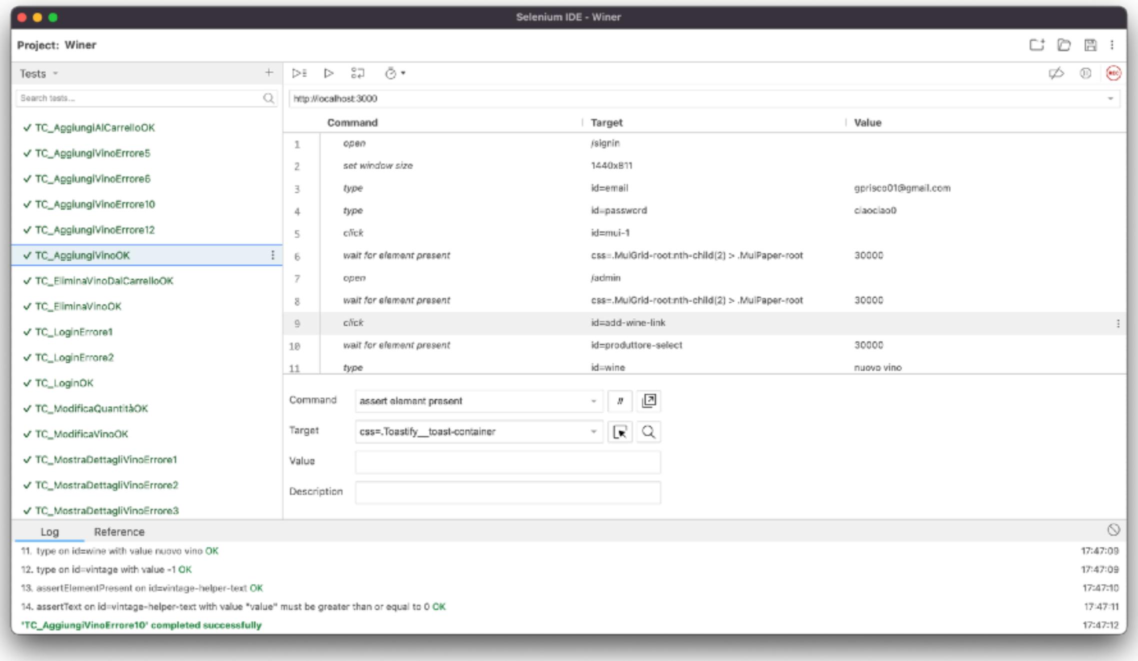
```
it('should get the catalog', async () => {
  const response = await request(app.getHttpServer())
    .get('/wine?page=1')
    .expect(200);

  assert(response.body.currentPage === 1);
  assert(response.body.data instanceof Array);
});

it('should create a wine', () => {
  return request(app.getHttpServer())
    .post('/wine')
    .set('Authorization', bearerToken)
    .send({
      wine: 'nuovo-vino',
      vintage: 2017,
      winefamilyId: 297,
      wineryId: 546,
      winegrapes: [],
      price: 10.0,
      availability: 40,
    } as CreateWineDto)
    .expect(201);
});

it('should get wine details', async () => {
  const response = await request(app.getHttpServer())
    .get('/wine/capatosta/2016')
    .expect(200);

  assert(response.body.wine === 'capatosta');
  assert(response.body.vintage === 2016);
  return;
});

it('should update an existing wine', () => {
  return request(app.getHttpServer())
    .patch(`/wine/capatosta/2016`)
    .set('Authorization', bearerToken)
    .send({
      winefamilyId: 297,
      wineryId: 546,
    } as UpdateWineDto)
    .expect(200);
});

it('should delete a wine', () => {
  return request(app.getHttpServer())
    .delete('/wine/nuovo-vino/2017')
    .set('Authorization', bearerToken)
    .expect(200);
});
```

Test Cases Catalogo



```
> Executing task: npm run test:e2e <


> winer-backend@0.0.1 test:e2e
> jest --config ./test/jest-e2e.json --detectOpenHandles

PASS  test/payment.e2e-spec.ts (81.477 s)
  CartController (e2e)
    ✓ should create an order after payment (547 ms)
    ✓ should invalidate credit card number (468 ms)
    ✓ should invalidate cvc (178 ms)

PASS  test/wine.e2e-spec.ts (24.949 s)
  WineController (e2e)
    ✓ should get the catalog (1515 ms)
    ✓ should create a wine (1112 ms)
    ✓ should get wine details (211 ms)
    ✓ should update an existing wine (626 ms)
    ✓ should delete a wine (178 ms)

PASS  test/cart.e2e-spec.ts (35.832 s)
  CartController (e2e)
    ✓ should get an empty cart (405 ms)
    ✓ should fail adding a non-existing wine to the cart (157 ms)
    ✓ should add a wine to the cart (628 ms)
    ✓ should not update a wine quantity in the cart (quantity > availability) (612 ms)
    ✓ should update a wine quantity (528 ms)
    ✓ should delete an item from the cart (126 ms)

PASS  test/auth.e2e-spec.ts (25.04 s)
  AuthController (e2e)
    ✓ should create an admin user (881 ms)
    ✓ should not create the same user (83 ms)
    ✓ should fail creating a user (30 ms)
    ✓ should complain about email format (133 ms)
    ✓ should issue an access_token (581 ms)
    ✓ should return 401 for invalid user (214 ms)
    ✓ should return an admin user profile (529 ms)
    ✓ should return 401 for invalid token (33 ms)

Test Suites: 4 passed, 4 total
Tests:       22 passed, 22 total
Snapshots:   0 total
Time:        168.762 s
Ran all test suites.

Terminal will be reused by tasks, press any key to close it.
```

Output dell'esecuzione dei test



Project: Winer

✓ TC_AggiungiAlCarrelloOK
✓ TC_AggiungiVinoErrore5
✓ TC_AggiungiVinoErrore6
✓ TC_AggiungiVinoErrore10
✓ TC_AggiungiVinoErrore12
✓ TC_AggiungiVinoOK
✓ TC_EliminaVinoDalCarrelloOK
✓ TC_EliminaVinoOK
✓ TC_LoginErrore1
✓ TC_LoginErrore2
✓ TC_LoginOK
✓ TC_ModificaQuantitàOK
✓ TC_ModificaVinoOK
✓ TC_MostraDettagliVinoErrore1
✓ TC_MostraDettagliVinoErrore2
✓ TC_MostraDettagliVinoErrore3

http://localhost:3000

	Command	Target	Value
1	open	/signin	
2	set window size	1440x811	
3	type	id=email	gprisco01@gmail.com
4	type	id=password	ciaociao0
5	click	id=mui-1	
6	wait for element present	css=.MuiGrid-root:nth-child(2) > .MuiPaper-root	30000
7	open	/admin	
8	wait for element present	css=.MuiGrid-root:nth-child(2) > .MuiPaper-root	30000
9	click	id=add-wine-link	
10	wait for element present	id=produttore-select	30000
11	type	id=wine	nuovo vino

Command: assert element present
Target: css=.Toastify__toast-container
Value:
Description:

```
11. type on id=wine with value nuovo vino OK                                                    17:47:09
12. type on id=vintage with value -1 OK                                                         17:47:09
13. assertElementPresent on id=vintage-helper-text OK                                           17:47:10
14. assertText on id=vintage-helper-text with value "value" must be greater than or equal to 0 OK 17:47:11
'TC_AggiungiVinoErrore10' completed successfully                                                17:47:12
```

Test con Selenium

L'esecuzione degli integration tests ci ha permesso, oltre a verificare che non ci fossero bug nei sottosistemi, di testare l'intera logica di business della nostra applicazione.