

**Nombre del QA aspirante:** Verónica Garcés

**Fecha:** jueves 18 de enero del 2024

**Requisitos de la evaluación:**

Git

Docker con Docker compose

**Tecnologías utilizadas**

Selenium (para pruebas automatizadas)

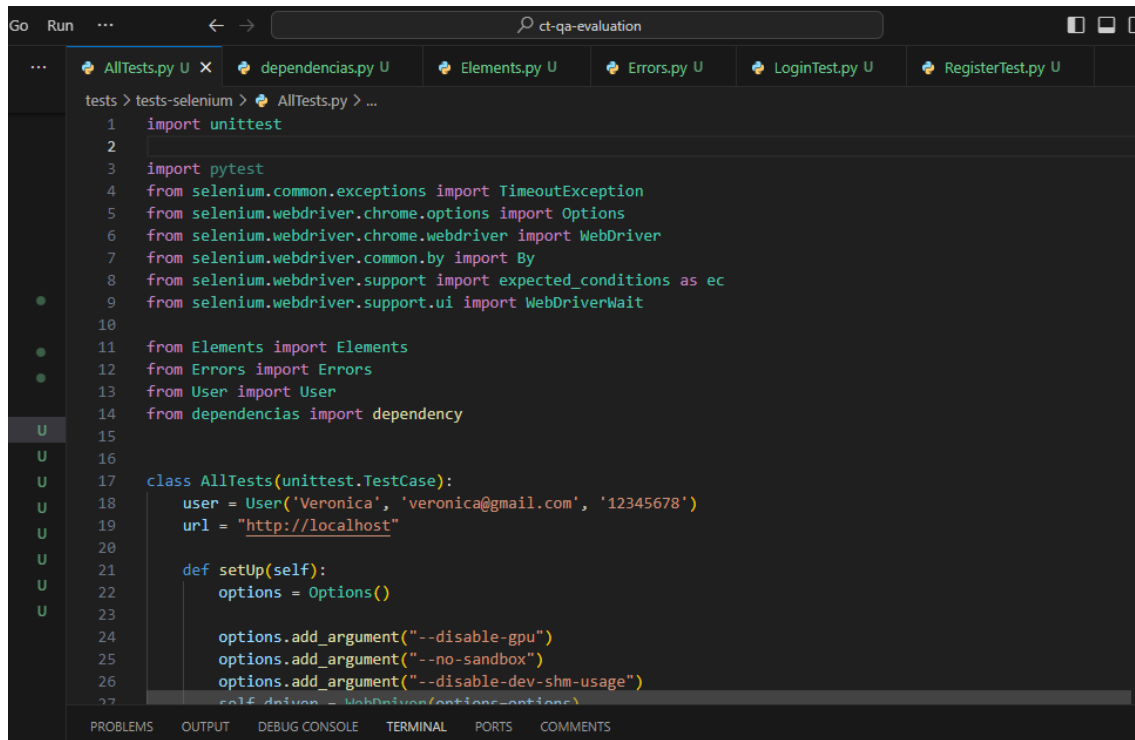
Visual

**Pasos para seguir:**

- Clonar el repositorio: el repositorio se clonará desde github por medio de gh repo clone para poder clonar el proyecto y poder tenerlo en Visual Studio Code
- Crear un archivo .env ajustándolo a cualquier variable deseable
- Iniciar el proyecto con Docker Compose Up -d : el cual hará que se inicie los servicios definidos en el archivo docker-compose.yml en segundo plano
- Acceder al contenedor de desarrollo (php.dev) utilizando Docker Compose Exec php.dev bash y ejecutar las siguientes líneas:
- Instalar las dependencias de php : composer install
- Generar una nueva clave de aplicación: php artisan key:generate
- Migrar la base de datos: php artisan migrate
- Correr el programa: php artisan service (para el Backend)
- Correr el programa: npm run dev (para el Frontend)

Luego de haber corrido el programa y poder observar detalladamente lo que contiene el sistema internamente, se procede a realizar los siguientes pasos solicitados para la respectiva evaluación:

Antes de realizar cada prueba, se creó un archivo donde se irán almacenando las pruebas correspondientes a realizar, teniendo dentro de este archivo las respectivas importaciones por parte de selenium con Python, creando una clase con funciones, donde cada función pertenece a los pasos que se piden al evaluar.



```
1 import unittest
2
3 import pytest
4 from selenium.common.exceptions import TimeoutException
5 from selenium.webdriver.chrome.options import Options
6 from selenium.webdriver.chrome.webdriver import WebDriver
7 from selenium.webdriver.common.by import By
8 from selenium.webdriver.support import expected_conditions as ec
9 from selenium.webdriver.support.ui import WebDriverWait
10
11 from Elements import Elements
12 from Errors import Errors
13 from User import User
14 from dependencias import dependency
15
16
17 class AllTests(unittest.TestCase):
18     user = User('Veronica', 'veronica@gmail.com', '12345678')
19     url = "http://localhost"
20
21     def setUp(self):
22         options = Options()
23
24         options.add_argument("--disable-gpu")
25         options.add_argument("--no-sandbox")
26         options.add_argument("--disable-dev-shm-usage")
27         self.driver = WebDriver(options=options)
```

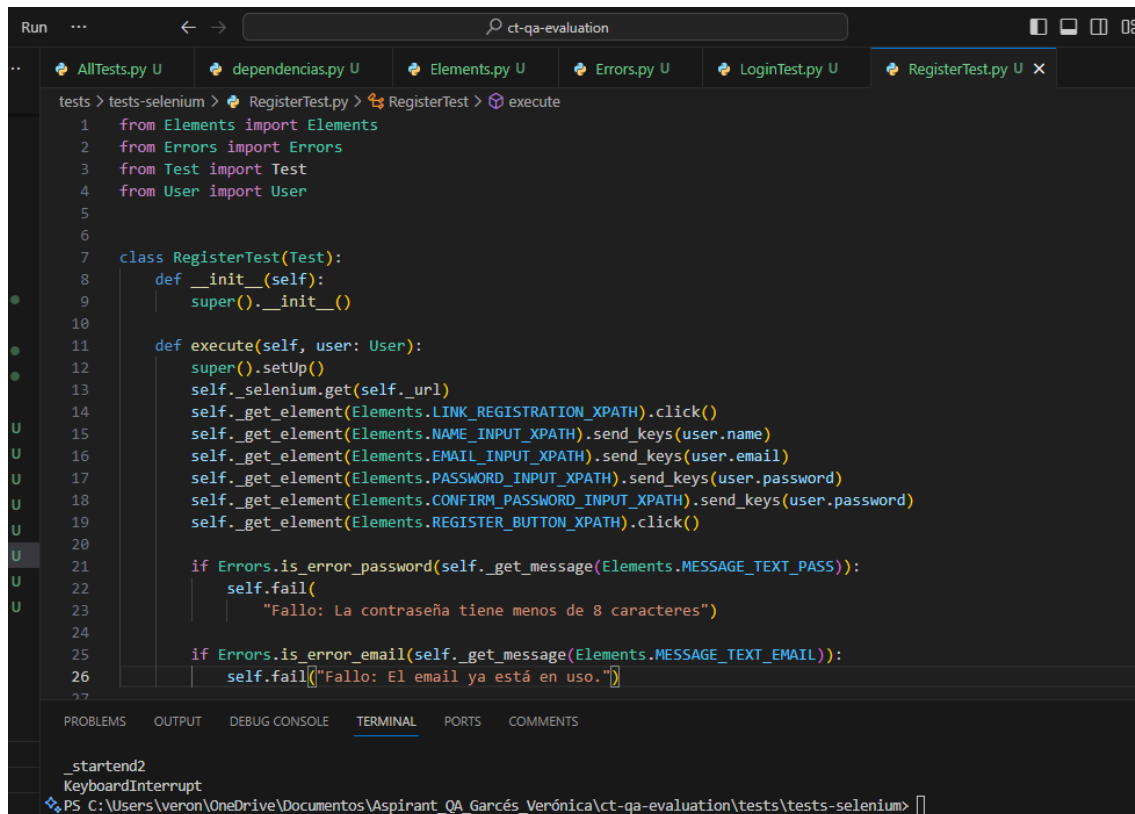
También se ha creado un archivo llamado Elements donde se almacena todos los Xpatch que se van a utilizar en cada prueba, estos Xpatch son localizadores que permiten buscar la dirección correcta de lo que se solicite al momento de hacer las respectivas pruebas. También se crea un archivo donde se almacenan los errores presentados en las deferentes pruebas solicitadas. Y finalmente se crear un archivo de Errores, donde se irán almacenando los mensajes de cada error que se presente al momento de hacer la prueba.

```
Run ... ct-qa-evaluation
AllTests.py U dependencias.py U Elements.py X Errors.py U LoginTest.py U RegisterTest.py U
tests > tests-selenium > Elements.py > EMAIL_FIELD_REGISTER
1 from enum import Enum
2
3
4 class Elements(Enum):
5     EMAIL_FIELD_REGISTER = "/html/body/div/div/div[2]/form/div[1]/input"
6     EMAIL_FIELD_LOGIN = "/html[1]/body[1]/div[1]/div[1]/div[2]/form[1]/div[1]/input[1]"
7     PASSWORD_FIELD_LOGIN = "/html/body/div/div/div[2]/form/div[2]/input"
8
9     PASSWORD_FIELD_REGISTER = "/html/body/div/div/div[2]/form/div[2]/input"
10    LOGIN_BUTTON = "/html[1]/body[1]/div[1]/div[1]/div[1]/a[1]"
11    SAVE_LOGIN_BUTTON = "/html/body/div/div/div[2]/form/div[4]/button"
12    MESSAGE_TEXT = "/html/body/div/div/div[2]/form/div[1]/div/p"
13    PROFILE_DROPDOWN = "/html/body/div/div/div[2]/nav/div[1]/div/div[2]/div[2]"
14    PROFILE_BUTTON = "/html/body/div/div/div[2]/nav/div[1]/div/div[2]/div[2]/div/div[3]/div/div[2]/a"
15    LOGOUT_BUTTON = "/html/body/div/div/div[2]/nav/div[1]/div/div[2]/div[2]/div/div[3]/div/form/div/button"
16    CURRENT_PASSWORD_INPUT = ('/html/body/div[1]/div/div[2]/main/div/div/div[2]/div[1]/div[2]/form/div['
17                               '1]/div/div[1]/input')
18    NEW_PASSWORD_INPUT = ('/html/body/div[1]/div/div[2]/main/div/div/div[2]/div[1]/div[2]/form/div[1]/div/div['
19                           '2]/input')
20    CONFIRM_NEW_PASSWORD_INPUT = ('/html/body/div[1]/div/div[2]/main/div/div/div[2]/div[1]/div[2]/form/div['
21                                  '1]/div/div[3]/input')
22    SAVE_BUTTON_XPATH = "/html/body/div[1]/div/div[2]/main/div/div/div[2]/div[1]/div[2]/form/div[2]/button"
23    NEW_NAME_INPUT_XPATH = "/html/body/div[1]/div/div[2]/main/div/div/div[1]/div[1]/div[2]/form/div[1]/div/div[1]/i
24    SAVE_PROFILE_BUTTON_XPATH = "/html/body/div[1]/div/div[2]/main/div/div/div[1]/div[1]/div[2]/form/div[2]/button"
25    NAME_CHECK_XPATH = "/html/body/div[1]/div/div[2]/nav/div[1]/div/div[2]/div[2]"
26    LINK_REGISTRATION_XPATH = "/html/body/div/div/div[1]/a[2]"
```

```
Run ... ct-qa-evaluation
AllTests.py U dependencias.py U Elements.py U Errors.py X LoginTest.py U RegisterTest.py U
tests > tests-selenium > Errors.py > ...
1 from enum import Enum
2
3
4 class Errors(Enum):
5     ERROR_PASSWORD_IN_REGISTER = "The password must be at least 8 characters."
6     ERROR_EMAIL_IN_REGISTER = "The email has already been taken."
7     ERROR_EMAIL_IN_LOGIN = "These credentials do not match our records."
8
9     @staticmethod
10    def is_error_password(message):
11        return Errors.ERROR_PASSWORD_IN_REGISTER.value in message
12
13    @staticmethod
14    def is_error_email(message):
15        return Errors.ERROR_EMAIL_IN_REGISTER.value in message
16
17    @staticmethod
18    def is_error_email_in_login(message):
19        return Errors.ERROR_EMAIL_IN_LOGIN.value in message
20
21
```

## Registrar nuevo usuario de prueba

Se crea un archivo de Registro nuevo de usuario de prueba donde se declara las importaciones necesarias y se crea una clase donde tiene funciones que permite llamar a los localizadores (Xpatch), y muestra un mensaje si se comete un error previamente ya programado.

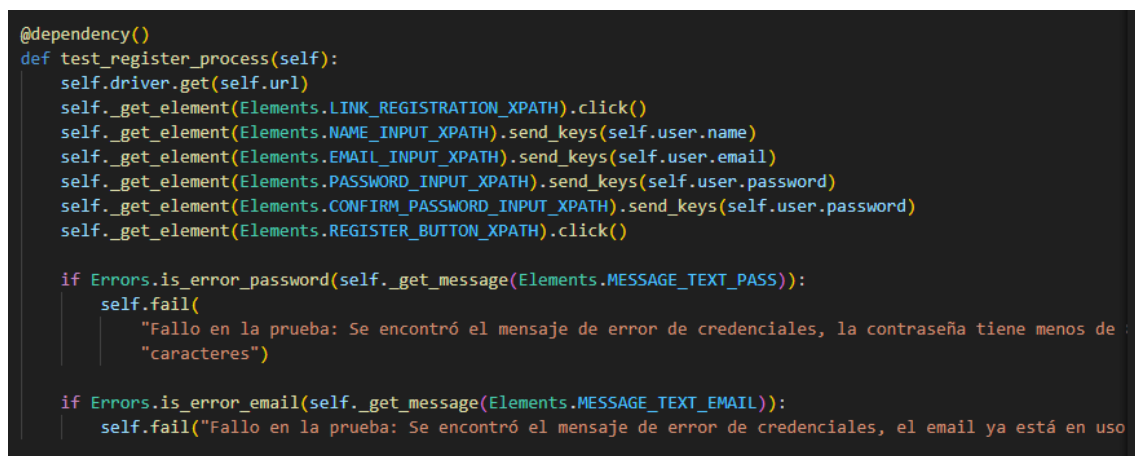


The screenshot shows an IDE window titled 'ct-qa-evaluation'. The top bar includes a 'Run' button and a search icon. Below the bar, there are tabs for 'AllTests.py U', 'dependencias.py U', 'Elements.py U', 'Errors.py U', 'LoginTest.py U', and 'RegisterTest.py U X'. The main editor displays the code for 'RegisterTest.py'. The code imports 'Elements', 'Errors', 'Test', and 'User' from a package named 'tests'. It defines a 'RegisterTest' class that inherits from 'Test'. The class has an 'execute' method that performs a registration process: it sets up the Selenium driver, clicks the registration link, and enters user details (name, email, password, confirm password) into the respective input fields. After clicking the register button, it checks for error messages. If the password is too short, it fails with the message 'Fallo: La contraseña tiene menos de 8 caracteres'. If the email is already in use, it fails with the message 'Fallo: El email ya está en uso.'.

```
1 from Elements import Elements
2 from Errors import Errors
3 from Test import Test
4 from User import User
5
6
7 class RegisterTest(Test):
8     def __init__(self):
9         super().__init__()
10
11     def execute(self, user: User):
12         super().setUp()
13         self.selenium.get(self.url)
14         self._get_element(Elements.LINK_REGISTRATION_XPATH).click()
15         self._get_element(Elements.NAME_INPUT_XPATH).send_keys(user.name)
16         self._get_element(Elements.EMAIL_INPUT_XPATH).send_keys(user.email)
17         self._get_element(Elements.PASSWORD_INPUT_XPATH).send_keys(user.password)
18         self._get_element(Elements.CONFIRM_PASSWORD_INPUT_XPATH).send_keys(user.password)
19         self._get_element(Elements.REGISTER_BUTTON_XPATH).click()
20
21         if Errors.is_error_password(self._get_message(Elements.MESSAGE_TEXT_PASS)):
22             self.fail(
23                 "Fallo: La contraseña tiene menos de 8 caracteres")
24
25         if Errors.is_error_email(self._get_message(Elements.MESSAGE_TEXT_EMAIL)):
26             self.fail("Fallo: El email ya está en uso.")
27
```

The bottom panel shows the 'TERMINAL' tab with the following output:

```
_startend2
KeyboardInterrupt
PS C:\Users\veron\OneDrive\Documentos\Aspirant_QA_Garcés_Verónica\ct-qa-evaluation\tests\tests-selenium>
```



The screenshot shows a Selenium test script for registration. It starts with a '@dependency()' decorator. The 'test\_register\_process' method uses 'self.driver' to get the URL and perform the registration steps: clicking the registration link, entering user details (name, email, password, confirm password) into the respective input fields, and clicking the register button. After clicking the register button, it checks for error messages. If the password is too short, it fails with the message 'Fallo en la prueba: Se encontró el mensaje de error de credenciales, la contraseña tiene menos de 8 caracteres'. If the email is already in use, it fails with the message 'Fallo en la prueba: Se encontró el mensaje de error de credenciales, el email ya está en uso.'.

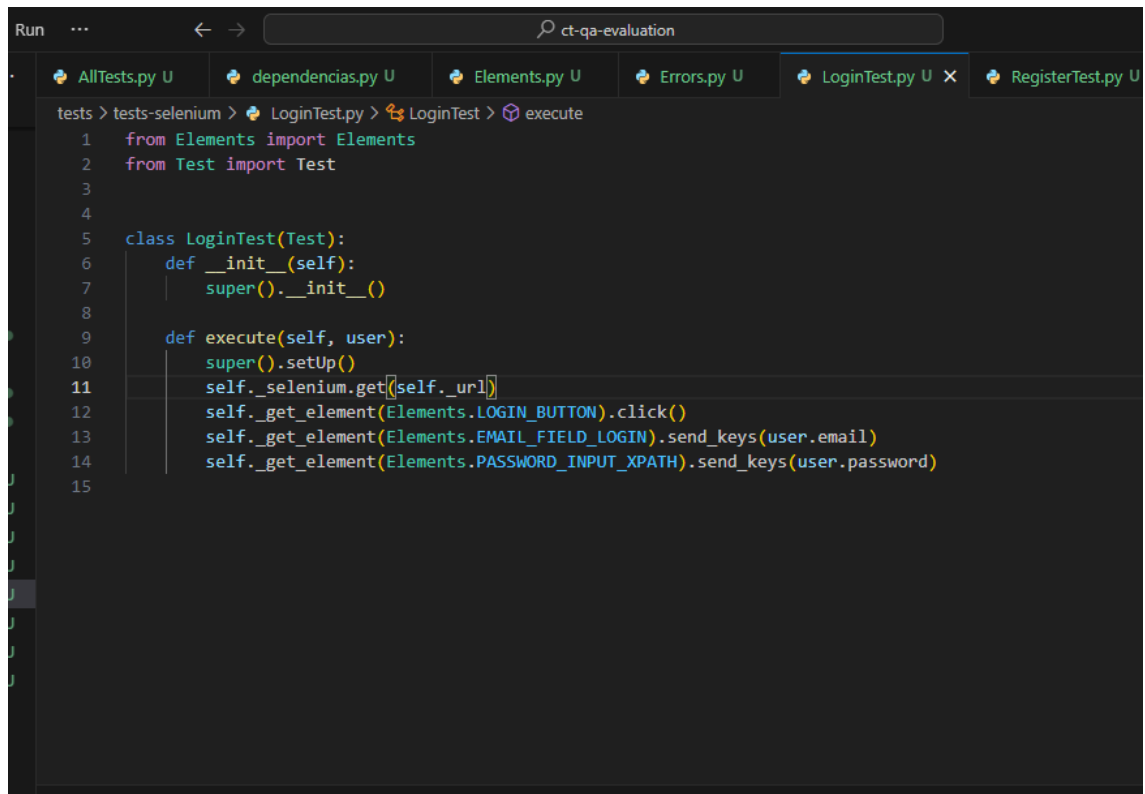
```
@dependency()
def test_register_process(self):
    self.driver.get(self.url)
    self._get_element(Elements.LINK_REGISTRATION_XPATH).click()
    self._get_element(Elements.NAME_INPUT_XPATH).send_keys(self.user.name)
    self._get_element(Elements.EMAIL_INPUT_XPATH).send_keys(self.user.email)
    self._get_element(Elements.PASSWORD_INPUT_XPATH).send_keys(self.user.password)
    self._get_element(Elements.CONFIRM_PASSWORD_INPUT_XPATH).send_keys(self.user.password)
    self._get_element(Elements.REGISTER_BUTTON_XPATH).click()

    if Errors.is_error_password(self._get_message(Elements.MESSAGE_TEXT_PASS)):
        self.fail(
            "Fallo en la prueba: Se encontró el mensaje de error de credenciales, la contraseña tiene menos de 8 caracteres")

    if Errors.is_error_email(self._get_message(Elements.MESSAGE_TEXT_EMAIL)):
        self.fail("Fallo en la prueba: Se encontró el mensaje de error de credenciales, el email ya está en uso")
```

Ingresa usuario de prueba creado

En este procedimiento, se crea un archivo donde almacena las importaciones correspondientes y también los Xpatch que son los localizadores que permitirá que al momento de hacer las pruebas con selenium vaya validando cada enlace



```
Run ... ct-qa-evaluation
AllTests.py U  dependencias.py U  Elements.py U  Errors.py U  LoginTest.py U X  RegisterTest.py U
tests > tests-selenium > LoginTest.py > LoginTest > execute
1  from Elements import Elements
2  from Test import Test
3
4
5  class LoginTest(Test):
6      def __init__(self):
7          super().__init__()
8
9      def execute(self, user):
10         super().setUp()
11         self._selenium.get(self._url)
12         self._get_element(Elements.LOGIN_BUTTON).click()
13         self._get_element(Elements.EMAIL_FIELD_LOGIN).send_keys(user.email)
14         self._get_element(Elements.PASSWORD_INPUT_XPATH).send_keys(user.password)
15
```

```
@dependency(depends=['test_register_process'], scope='class')
def test_login_process(self, delay_time=10):

    self.driver.get(self.url)
    self._get_element(Elements.LOGIN_BUTTON).click()
    self._get_element(Elements.EMAIL_FIELD_REGISTER).send_keys(self.user.email)
    self._get_element(Elements.PASSWORD_FIELD_LOGIN).send_keys(self.user.password)
    self._get_element(Elements.SAVE_LOGIN_BUTTON, delay_time).click()

def _get_element(self, element: Elements, delay_time=10):
    return WebDriverWait(self.driver, delay_time).until(
        ec.visibility_of_element_located((By.XPATH, element.value))
    )

def _get_message(self, element: Elements, delay_time=5):
    try:
        return self._get_element(element, delay_time).text
    except TimeoutException:
        return ""
```

Cambiar la contraseña

Dentro del archivo donde se encuentran todas las pruebas, se crea una función donde se llaman a los xpath que van a ser recorridos para poder realizar las pruebas

```

    return

@dependency(depends=['test_login_process'], scope='class')
def test_change_password(self):
    self.driver.get(self.url)
    new_password = "12345ABCD"
    self.test_login_process(100)
    self._get_element(Elements.PROFILE_DROPDOWN, delay_time=10).click()
    self._get_element(Elements.PROFILE_BUTTON).click()
    self._get_element(Elements.CURRENT_PASSWORD_INPUT).send_keys(self.user.password)
    self._get_element(Elements.NEW_PASSWORD_INPUT).send_keys(new_password)
    self._get_element(Elements.CONFIRM_NEW_PASSWORD_INPUT).send_keys(new_password)

    if Errors.is_error_password(self._get_message(Elements.MESSAGE_TEXT_PASS)):
        self.fail("Fallo en la prueba: Se encontró el mensaje de error de credenciales, la contraseña tiene men
        "caracteres")

```

## Anexos

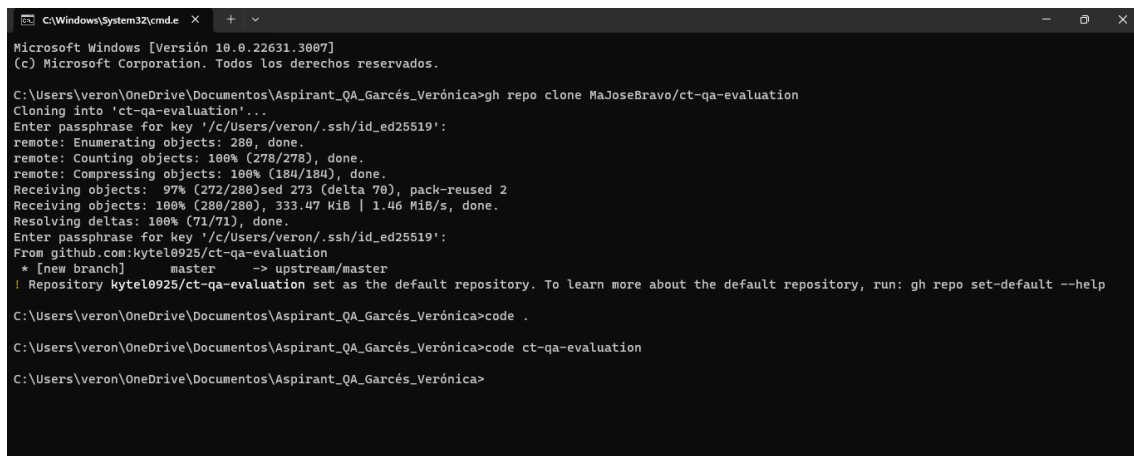
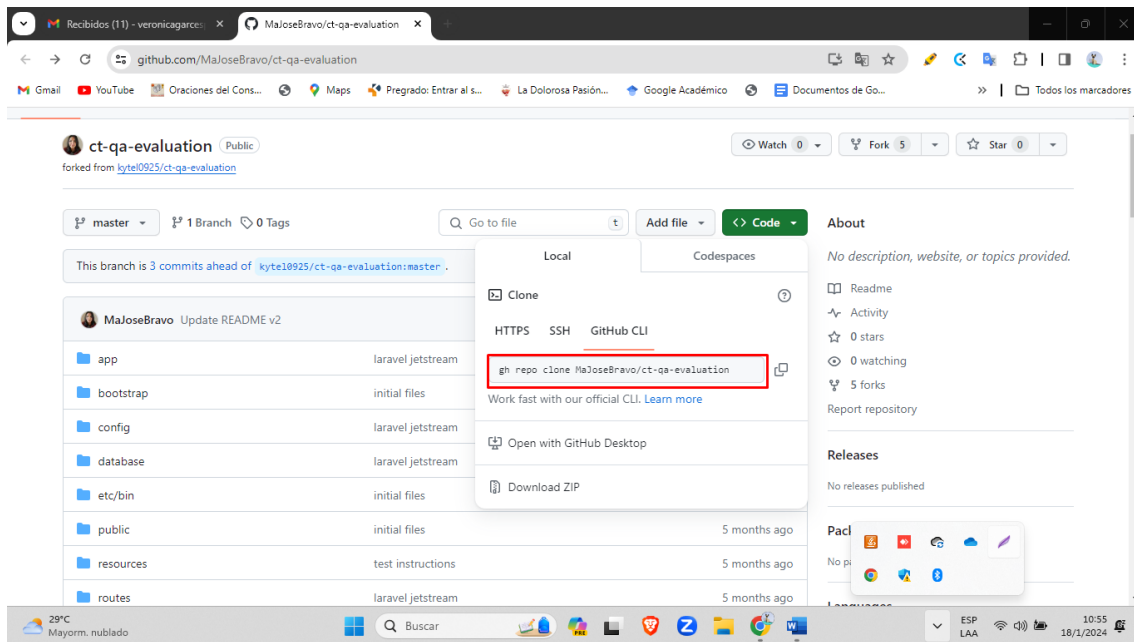


Ilustración 1: Clonando el repositorio

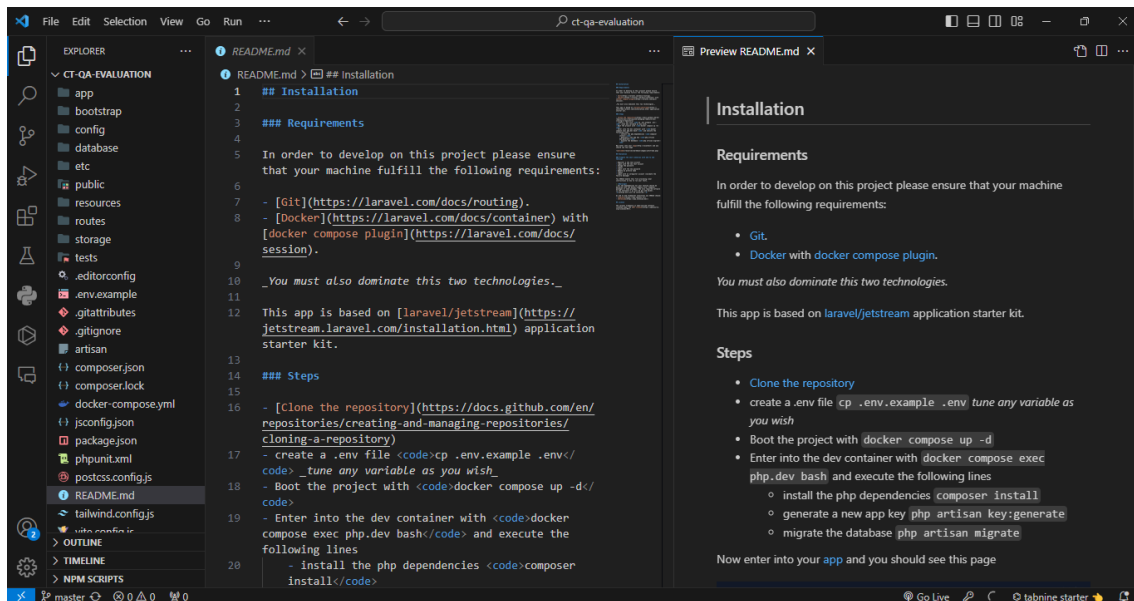


Ilustración 2: Repositorio Clonado

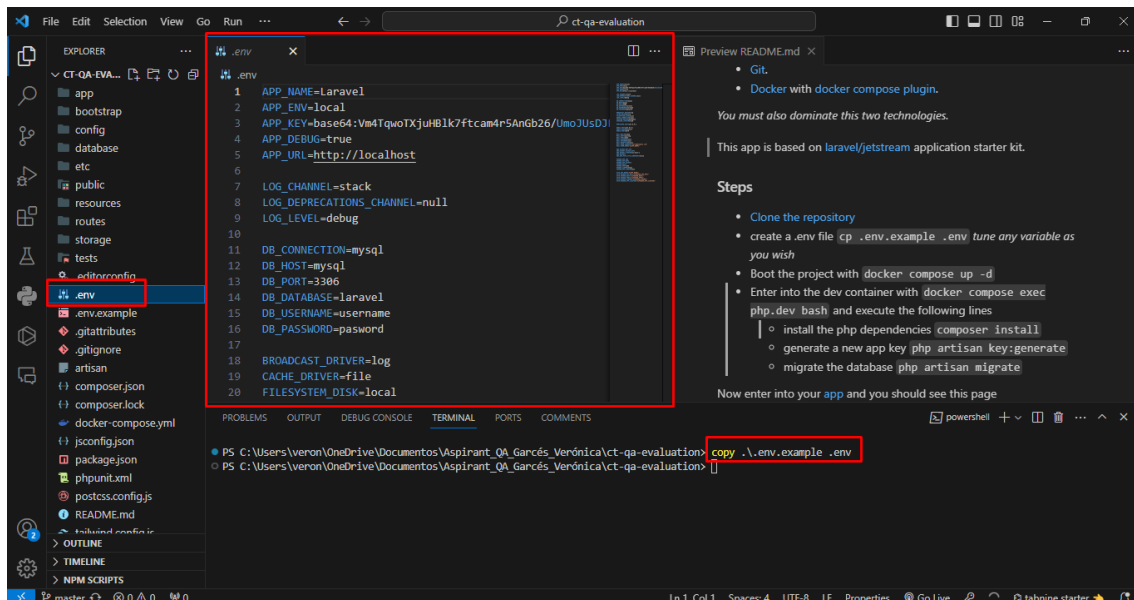


Ilustración 3: creación de archivo .env

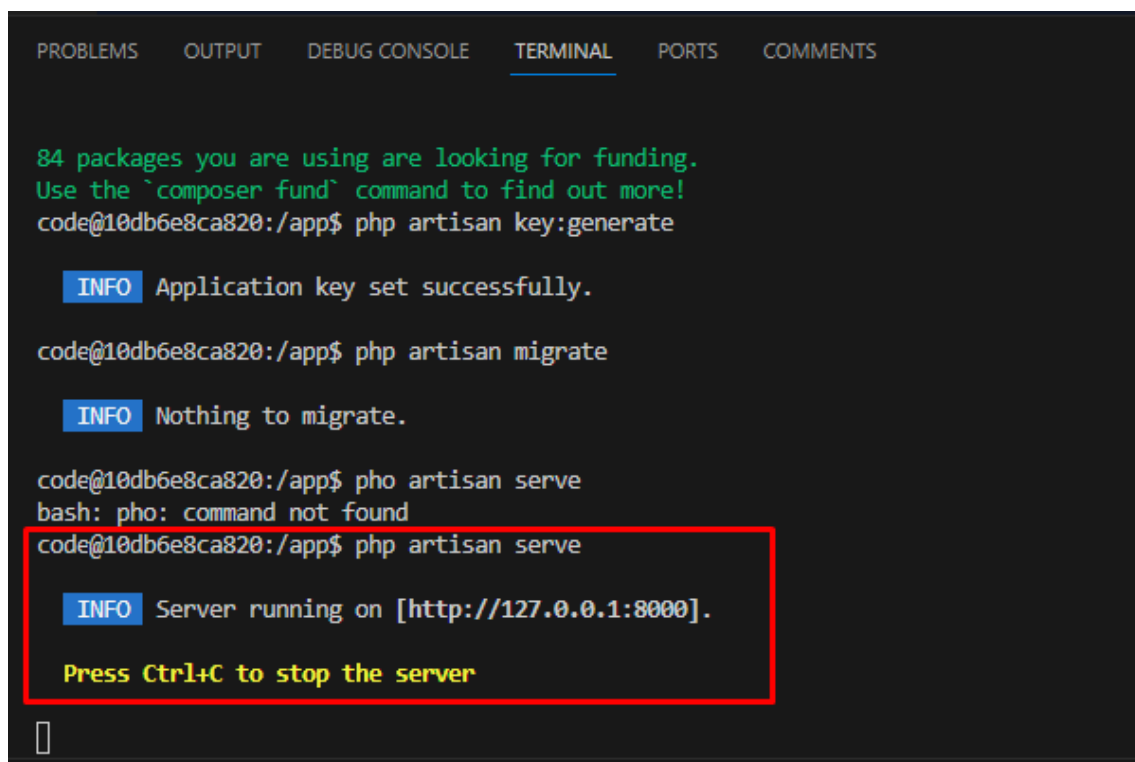
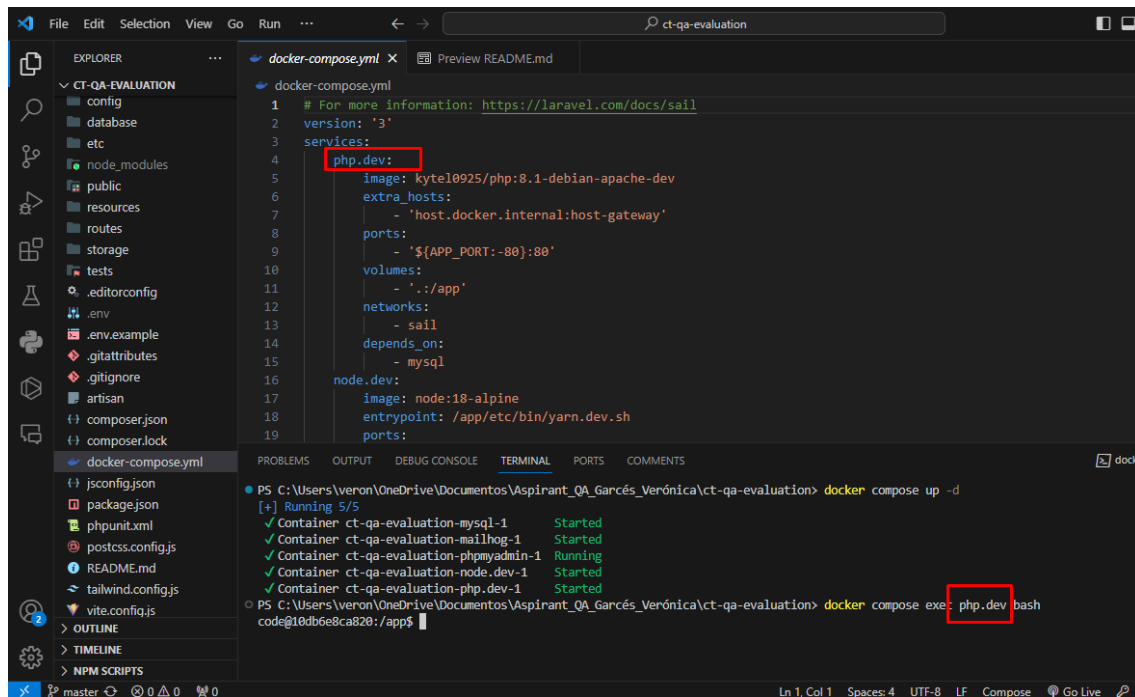


Ilustración 4: Corriendo en el Back End



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

PS C:\Users\veron\OneDrive\Documentos\Aspirant_QA_Garcés_Verónica\ct-qa-evaluation> npm run dev

> dev
> vite

Port 5173 is in use, trying another one...

VITE v5.0.2  ready in 1479 ms

→ Local:   http://localhost:5174/
→ Network: use --host to expose
→ press h + enter to show help

LARAVEL v10.20.0  plugin v0.8.0

→ APP_URL: http://localhost
```

Ilustración 5: Corriendo en el Front End

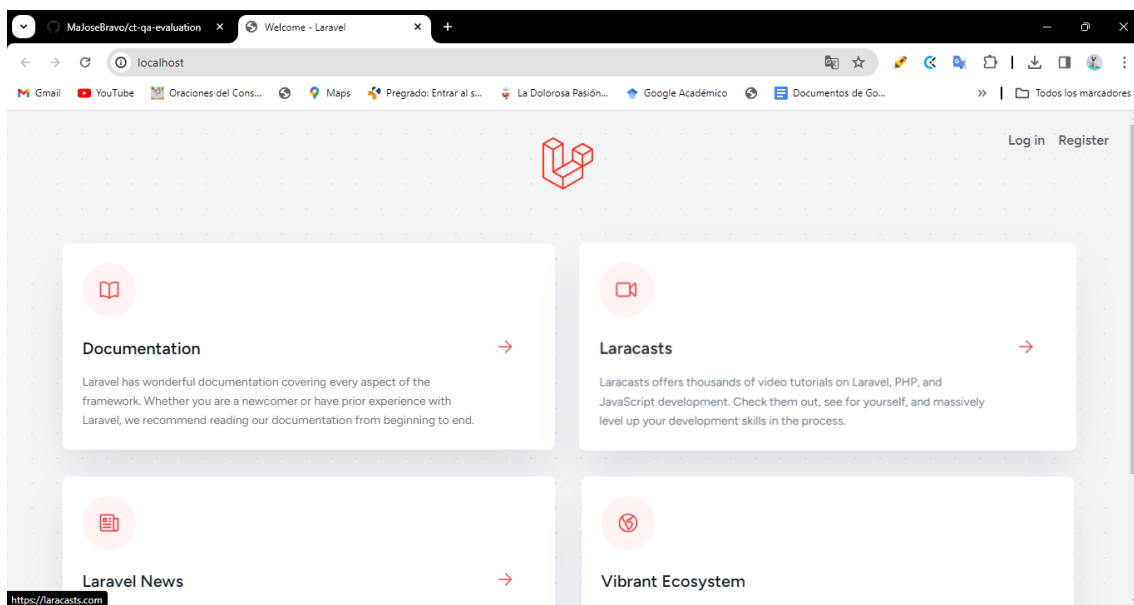


Ilustración 6: Página principal del sistema