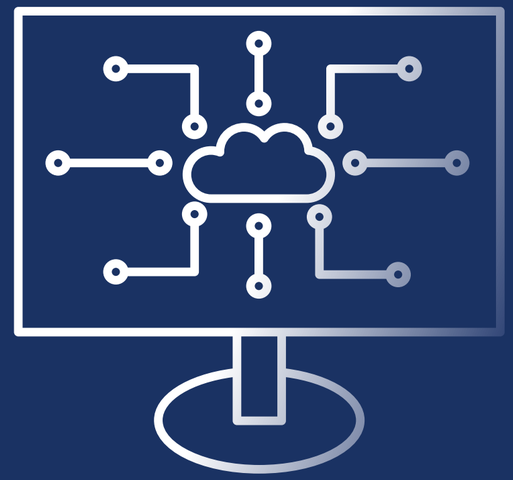
Programación I

Lic. Julia Monasterio
marmonasterio@uade.com.ar



UADE

Lic. Julia Monasterio

Unidades

- **Unidad 1:** Repaso de matrices, estructura, operaciones y uso en funciones
- **Unidad 2:** Implementación y gestión en proyectos en Git
- **Unidad 3:** Listas avanzadas, cadena de caracteres y expresiones regulares.
- **Unidad 4:** Manipulación avanzada de diccionarios, tuplas y conjuntos.
- **Unidad 5:** Excepciones y pruebas unitarias
- **Unidad 6:** Archivos
- **Unidad 7:** Recursividad

UADE

Lic. Julia Monasterio

Clase N° 12

TEMAS

- Entornos de desarrollo
- Test Unitarios

UADE

Lic. Julia Monasterio

Entornos de desarrollo

Entornos de desarrollo

- Al desarrollar una aplicación, es esencial contar con al menos dos entornos:
  - **Desarrollo**
  - **Producción**

Entorno de desarrollo

- El entorno de desarrollo suele ser el equipo o el servidor donde el **programador escribe y prueba su codigo.**

Entorno de producción

El entorno de producción es el que se utiliza para desplegar y hacer pública la aplicación.

Otros entornos

- Pueden requerirse otros entornos:
  - Entorno de pruebas
  - Entorno de preproducción

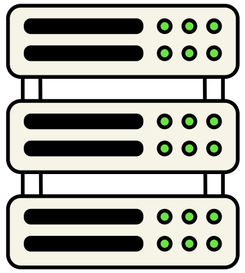
UADE

Lic. Julia Monasterio

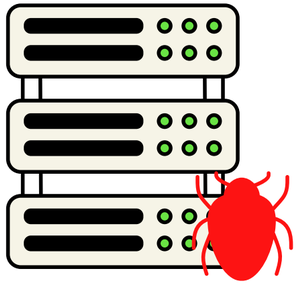
Pasajes frecuentes

Pasajes controlados

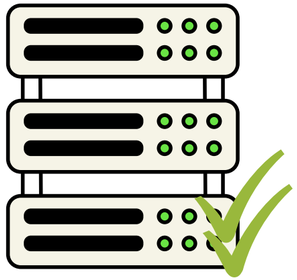
Pasajes muy controlados



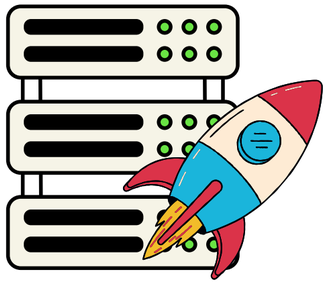
DESARROLLO



PRUEBAS



PREPRODUCCIÓN

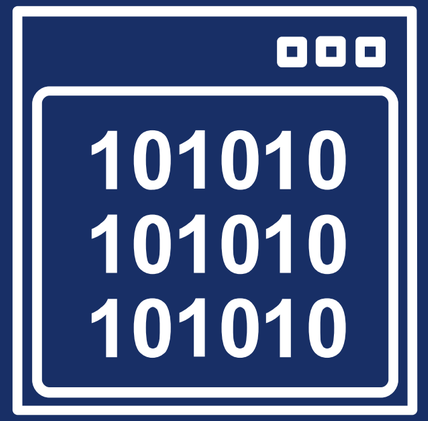


PRODUCCION

UADE

Lic. Julia Monasterio

Entorno de desarrollo



Entorno de desarrollo

- Es donde los **programadores escriben y prueban el código en las etapas tempranas del proyecto**.

- Utilizado exclusivamente por los desarrolladores, suele estar en sus equipos locales o en servidores dedicados.

- La principal característica del entorno de desarrollo es su capacidad de **soportar iteraciones rápidas**. Los desarrolladores pueden escribir código, **ejecutar pruebas unitarias** y hacer ajustes en tiempo real, acelerando el proceso de desarrollo y permitiendo una integración continua.

UADE

Lic. Julia Monasterio

Entorno de pruebas

Entorno de pruebas

- Se utiliza para **verificar la funcionalidad del software** y **detectar errores antes de que el software avance a etapas más críticas**.

- Utilizado principalmente por **testers**

- Se simulan condiciones reales para realizar pruebas funcionales y no funcionales. También se suelen utilizar datos de prueba y herramientas de **automatización de pruebas** para asegurar la cobertura y eficiencia del proceso de testing.

UADE

Lic. Julia Monasterio

Entorno de preproducción

Entorno de preproducción

- Es una réplica casi idéntica del entorno de producción.
- Su propósito principal es validar la aplicación en condiciones que imitan el entorno de producción antes de su despliegue final.
- Este entorno es utilizado por el equipo de QA y también el cliente para realizar las últimas pruebas y asegurar que el software cumpla con los estándares de calidad y esté libre de errores.
- Es la última etapa de prueba antes de que la aplicación sea desplegada públicamente.

UADE

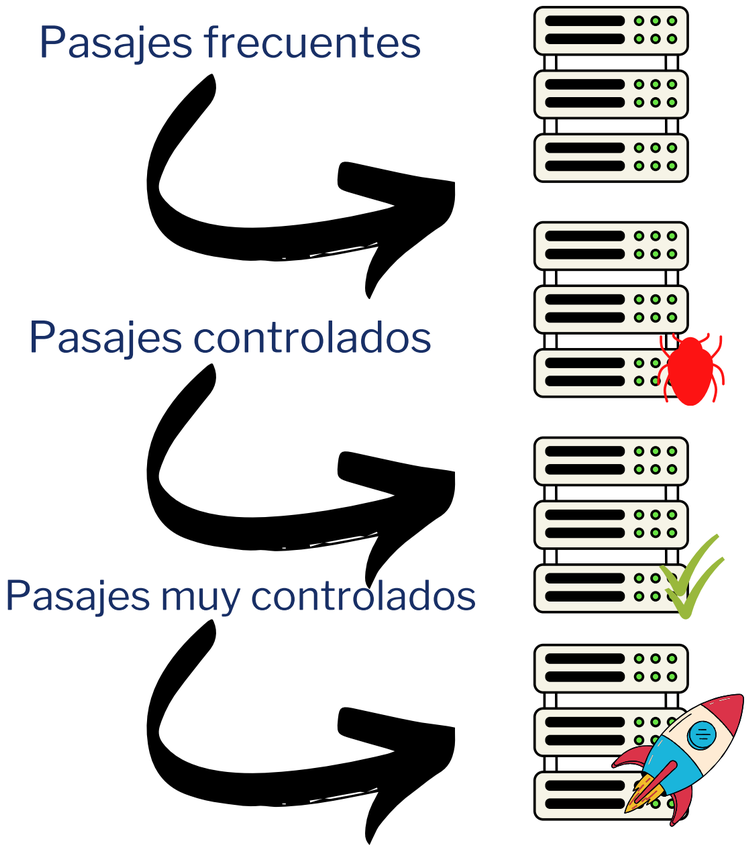
Lic. Julia Monasterio

Entorno de producción

Entorno de producción

- Donde la aplicación es finalmente desplegada y hecha pública para los usuarios finales.
- **Este entorno debe ser extremadamente estable y seguro, ya que cualquier problema puede afectar a los usuarios reales.**
- Aquí se gestionan los datos reales y se asegura que la aplicación funcione correctamente bajo carga real.
- El entorno de producción es supervisado constantemente para garantizar su disponibilidad y rendimiento óptimos.

Nuevas funcionalidades



- El ciclo de desarrollo de nuevas funcionalidades comienza en el entorno de desarrollo, donde los programadores escriben y prueban el código.
- Una vez que estas nuevas funcionalidades han sido probadas y se consideran estables, el código se mueve al entorno de pre-producción.
- En este entorno, las funcionalidades se prueban en un ambiente que simula las condiciones de producción para asegurarse de que funcionen correctamente y no interfieran con otros componentes del sistema.

Reportes QA/Testing

- En el entorno de pre-producción, los equipos de **QA (Quality Assurance)** realizan pruebas exhaustivas para garantizar la calidad del software.

- Cualquier funcionalidad nueva o cambios realizados en pre-producción son evaluados rigurosamente.

- Una vez que el equipo de QA confirma que el software cumple con los estándares de calidad, **se generan reportes que documentan los resultados de estas pruebas**.

Errores

- Durante las pruebas en el entorno de pruebas, **pueden detectarse errores o fallos en la aplicación**.
- Cuando esto ocurre, **los reportes de errores detallados son enviados de vuelta al entorno de desarrollo**.
- Los desarrolladores analizan estos reportes para identificar y corregir los errores.
- Este proceso iterativo de detectar y corregir errores se repite hasta que el software esté libre de fallos conocidos y listo para avanzar nuevamente a pre-producción y eventualmente a producción.

UADE

Lic. Julia Monasterio

Pruebas manuales
vs
Pruebas automatizadas

Pruebas manuales

- Son **llevadas a cabo por personas**, quienes navegan e interactúan con el software utilizando herramientas adecuadas para cada caso.

- Estas pruebas son **costosas**, ya que requieren profesionales dedicados para configurar el entorno y ejecutar las pruebas.

- Además, están expuestas a errores humanos, como errores tipográficos u omisión de pasos durante la prueba.

Pruebas automatizadas

- Son realizadas **por máquinas** que ejecutan scripts de prueba previamente escritos.
- Estos pueden variar en complejidad, según la necesidad del programa.
- Las pruebas automatizadas son más rápidas, su efectividad depende de la calidad con la que se hayan escrito los scripts de prueba

UADE

Lic. Julia Monasterio

Tipos de pruebas

Pruebas unitarias

- Validan individualmente las unidades más pequeñas del código, como las funciones, asegurando que cada una actúe correctamente de forma aislada.

Pruebas de Integración

- Verifican la interacción entre múltiples componentes o módulos de software, asegurando que funcionen bien juntos

Pruebas de Sistemas

- Evalúan el sistema completo desde el inicio hasta el final, simulando escenarios reales para asegurar que todas las partes del sistema funcionen correctamente en conjunto

Pruebas de Aceptación

- Confirmar que el software cumple con los requisitos y expectativas del usuario final, asegurando que el producto es funcional y adecuado para su uso.

UADE

Lic. Julia Monasterio

Assert

Assert

- Es una instrucción que permite **realizar comprobaciones**

- Es utilizada para verificar que una condición especifica sea verdadera en un punto determinado del programa. **Si la condición assert resulta ser falsa, el programa lanza una excepcion AssertionError,** deteniendo la ejecución y mostrando un mensaje opcional que explica el fallo.

Assert - Sintaxis

assert <condicion>, <mensaje opcional>

<condicion>: es una expresión booleana que se espera que sea True. Si es False se lanza un AssertionError

<mensaje opcional>: es un mensaje que se muestra en el error para indicar que falló, facilitando la comprensión del problema

UADE

Lic. Julia Monasterio

Libreria Pytest

Libreria pytest

- Es una librería para realizar pruebas en Python, especialmente pruebas unitarias
- **pytest** detecta automáticamente todos los archivos de prueba que comienzan con **test_** o terminan en **_test.py.** Como así también las funciones dentro de esos archivos que comienzan con **_test**

Instalación pytest

- Inicialmente se debe instalar la libreria pytest para eso ejecutar

pip install pytest

- Trabajaremos con dos archivos uno para el desarrollo y otro para las pruebas

- Para poder ejecutar las pruebas, en la ubicación donde se encuentran los archivos que deseamos probar debemos escribir por consola el comando **pytest**

- Todos los archivos dentro de la carpeta con el prefijo test_ lo va a reconocer como una prueba para pytest.
- Si ejecutamos **pytest -v** nos indicará cada uno de los llamados y si el resultado de la ejecución fue correcto

Ejemplo 1

- Creamos la funcion suma:

```python
def suma(a, b):
    return a + b
```

- Creamos el archivo test_suma.py y hacemos la prueba:

```python
from main import suma

def test_suma():
    assert suma(3,4)==7
    assert suma(-1,1)==0
    assert suma(-1,-1)==-2
```

Ejemplo 1

- Resultado **pytest**:

```
============================================== test session starts ==============================================
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\julia\OneDrive\Desktop\Organizador\UADE\Segundo cuatrimestre\Programación 1\Curso Belgrano\Clase 13\Practica
 Hecha
collected 1 item

test_suma.py .                                                                                          [100%]

=============================================== 1 passed in 0.01s ===============================================
```

- Resultado **pytest -v**:

```
============================================== test session starts ==============================================
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\julia\AppData\Local\Programs\Python\Python312\python.e
xe
cachedir: .pytest_cache
rootdir: C:\Users\julia\OneDrive\Desktop\Organizador\UADE\Segundo cuatrimestre\Programación 1\Curso Belgrano\Clase 13\Practica
 Hecha
collected 1 item

test_suma.py::test_suma PASSED                                                                          [100%]

=============================================== 1 passed in 0.01s ===============================================
PS C:\Users\julia\OneDrive\Desktop\Organizador\UADE\Segundo cuatrimestre\Programación 1\Curso Belgrano\Clase 13\Practica Hecha
```

UADE

Lic. Julia Monasterio

Resúmen de la clase

- Entornos de prueba
- Tipos de pruebas
- Pruebas unitarias
- Pytest

Muchas gracias!

Consultas?