Argentina programa 4.0

Ministerio de Economía
Argentina

Secretaría de
Economía del Conocimiento

primero la gente

UNTREF
UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

CLASE 16: Testing de aplicaciones BackEnd

Diseño y planificación de pruebas

Ministerio de
Desarrollo Productivo
Argentina

Agenda de hoy

A. Testing de aplicaciones Backend

B. Planificar el testing en aplicaciones de backend

C. Herramientas en el mercado actual

   a. POSTMAN

   b. HOPPSCOTCH

   c. THUNDER CLIENT

   d. INSOMNIA

D. Elaborar un plan de testing

   a. Probar nuestra aplicación de backend

E. Proyecto Integrador: pautas del segundo proyecto

Testing de aplicaciones Backend

Al igual que la documentación aporta un valor agregado para entender el uso de aplicaciones backend, **el testing también es parte esencial del desarrollo de software**.

Éste se centra en probar y verificar la funcionalidad y rendimiento de la lógica de negocio y los componentes de una aplicación. Mediante el testing, buscaremos identificar posibles errores, asegurar la estabilidad del sistema y mejorar la calidad general del software.

Testing de aplicaciones Backend

Este proceso incluye la realización de pruebas exhaustivas a través de las cuales podremos identificar y corregir errores antes de que el sistema se implemente en Producción, y estos errores impacten en las aplicaciones que utilizan nuestro software de backend.

Esto permitirá contribuir a la mejora continua del producto garantizando la calidad y confiabilidad del software, permitiendo la detección temprana de errores y la implementación de mejoras en la funcionalidad y rendimiento del sistema.

Testing de aplicaciones Backend

El punto principal del Testing de aplicaciones backend es asegurar que los servicios y funcionalidades de la parte no visible de la aplicación funcionen correctamente.

Este punto incluye:

- la interacción con bases de datos
- los servicios web y APIs
- el sistema de archivos
- la correcta identificación de usuarios

Al poner a prueba todos estos componentes, buscaremos validar el funcionamiento individual y en conjunto de estos, asegurando que cumplan con los requisitos definidos.

Testing de aplicaciones Backend

Para lograr todo lo mencionado anteriormente, debemos crear y ejecutar diferentes tipos de pruebas.

Entre ellas se encuentran:

- las pruebas unitarias, las cuales verifican la funcionalidad de unidades de código aisladas
- las pruebas de integración, que se ocupan de evaluar la interacción entre diferentes componentes
- las pruebas de rendimiento, que miden la capacidad de respuesta y la escalabilidad del sistema bajo diferentes cargas de trabajo

Testing de aplicaciones Backend

Un enfoque común en el testing de aplicaciones backend es la automatización de pruebas.

Esto implica el uso de herramientas y frameworks para escribir scripts que ejecuten pruebas de manera programática. La automatización permite repetir las pruebas de forma rápida y consistente, ahorrando tiempo y recursos en comparación con las pruebas manuales.

Testing de aplicaciones Backend

Con todas estas premisas que conforman los fundamentos del Testing, nos ocuparemos de analizar las principales herramientas que nos guiarán por el camino del testing manual.
Para ello, diseñaremos un plan de testing ajustado a nuestras capacidades técnicas para garantizar el correcto funcionamiento de nuestro desarrollo, corrigiendo cualquier posible error que exista en este.

Planificar el testing en aplicaciones backend

Planificar el testing de aplicaciones Backend

La planificación del testing de aplicaciones web es crucial para garantizar una cobertura adecuada y eficiente de las pruebas.

Veamos a continuación, un extracto de los pasos clave a tener en cuenta para una correcta planificación de este proceso tan importante.

Planificar el testing de aplicaciones Backend

01
Definir objetivos

02
Establecer alcance

03
Diseñar tipos de pruebas

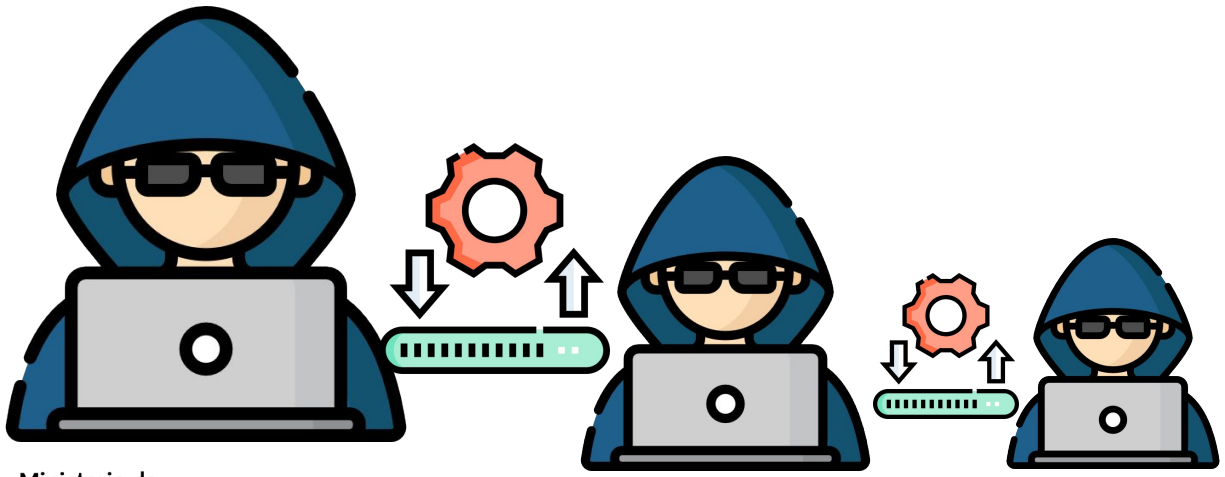
04
Crear un plan de pruebas

05
Asignar recursos y plazos

06
Configurar entornos de pruebas

07
Ejecutar las pruebas

08
Analizar pruebas e informar resultados



Planificar el testing de aplicaciones Backend

Ten presente que esto no es una regla estricta ni que en todos los ambientes laborales se sigue al pié de la letra.

Es un modelo ideal de cómo llevar una planificación asertiva de las mejores prácticas en ambientes de desarrollo de software.

Planificar el testing de aplicaciones Backend

Definir los objetivos del testing

Debemos identificar los objetivos específicos del testing para nuestra aplicación. Estos objetivos pueden incluir:

- validación de funcionalidades clave
- detección de errores
- evaluación del rendimiento y la usabilidad

entre otros.

Siempre debemos tener claridad sobre los objetivos. Esto nos ayudará a enfocar esfuerzos de testing de manera efectiva.

Planificar el testing de aplicaciones Backend

Establecer el alcance

Debemos determinar qué áreas serán cubiertas por el testing. Esto implica identificar las funcionalidades principales, casos de uso críticos y flujos de trabajo clave.

Además, debemos considerar también si el testing se enfocará en diferentes navegadores, dispositivos o plataformas, según las necesidades de tu público objetivo. En el caso de aplicaciones backend, este último punto no aplica, a menos que trabajemos con algún proceso de Server Side Rendering.

Planificar el testing de aplicaciones Backend

Diseñar los tipos de pruebas

Identifica los diferentes tipos de pruebas que se realizarán en tu aplicación web. Esto puede incluir:

- pruebas unitarias
- pruebas de integración
- pruebas de aceptación
- pruebas de rendimiento
- pruebas de seguridad

entre otras.

Debemos determinar cuáles son los más relevantes para la aplicación y cómo pueden combinarse para obtener la mejor cobertura de pruebas posible.

Planificar el testing de aplicaciones Backend

Plan de pruebas

Debemos elaborar un plan detallado que incluya los escenarios de prueba, casos de prueba específicos, datos de prueba necesarios además de los resultados esperados.

También durante la organización del plan de pruebas, debemos tener en cuenta este último proceso en función de los diferentes tipos de pruebas y las áreas funcionales de la aplicación web. Por último, establecer una secuencia lógica de ejecución de las pruebas.

Planificar el testing de aplicaciones Backend

Asignar recursos y plazos

Debemos determinar los recursos necesarios para llevar a cabo las pruebas, como ser el personal de pruebas, los entornos de prueba y las herramientas de testing.

Debemos asignar responsabilidades claras a los miembros del equipo y establecer plazos realistas para la ejecución de las pruebas. También debemos considerar la posibilidad de iteraciones y ajustes, en función de los resultados de las pruebas iniciales.

Planificar el testing de aplicaciones Backend

Configurar entornos de prueba

Debemos preparar entornos de prueba que sean similares o lo más cercano posibles a los entornos de producción.

Esto puede incluir:

- configurar servidores
- bases de datos
- entornos de red
- otros componentes necesarios que repliquen las condiciones de producción

Planificar el testing de aplicaciones Backend

Ejecutar las pruebas

Con todos o la mayoría de los puntos anteriores validados, es momento de llevar a cabo las pruebas, de acuerdo al plan establecido.

Toda prueba debe contar con un registro de las mismas para que, una vez ejecutadas, se puedan analizar los resultados de estas.

Planificar el testing de aplicaciones Backend

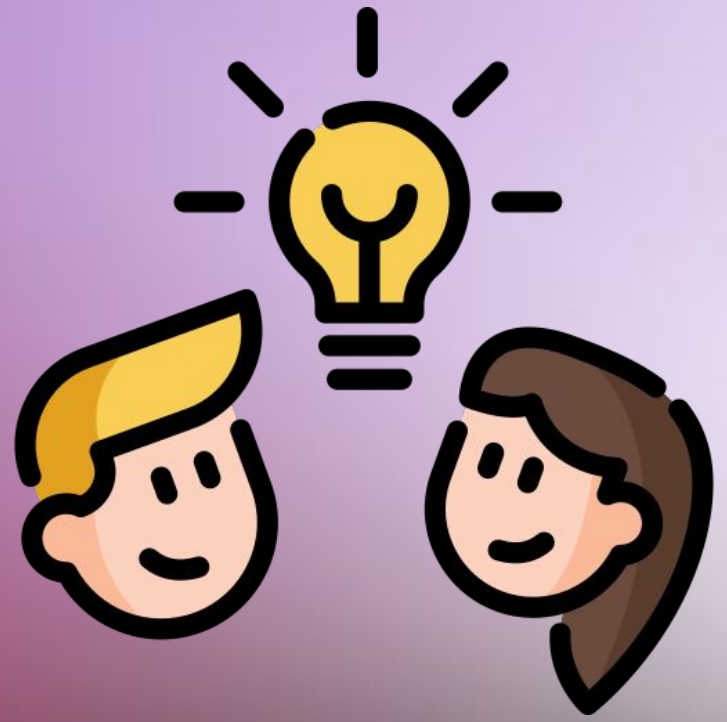
Informar y comunicar los resultados

El testing requiere también preparar informes de pruebas detallados que resuman los hallazgos y los resultados obtenidos. Concluidos los informes, debemos comunicar estos resultados de manera clara y concisa a todos los involucrados en el equipo de desarrollo y a otras partes interesadas.

Planificar el testing de aplicaciones Backend

Debemos recordar siempre que la planificación del testing es un proceso iterativo y adaptable.

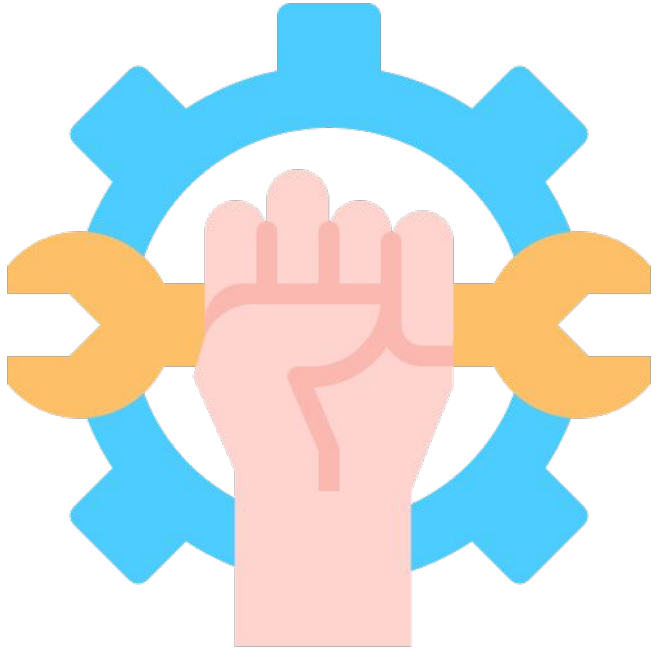
A medida que se descubren errores y se realizan mejoras en la aplicación web, es posible que sea necesario ajustar y ampliar el plan de pruebas para garantizar una cobertura adecuada.



Herramientas en el mercado actual

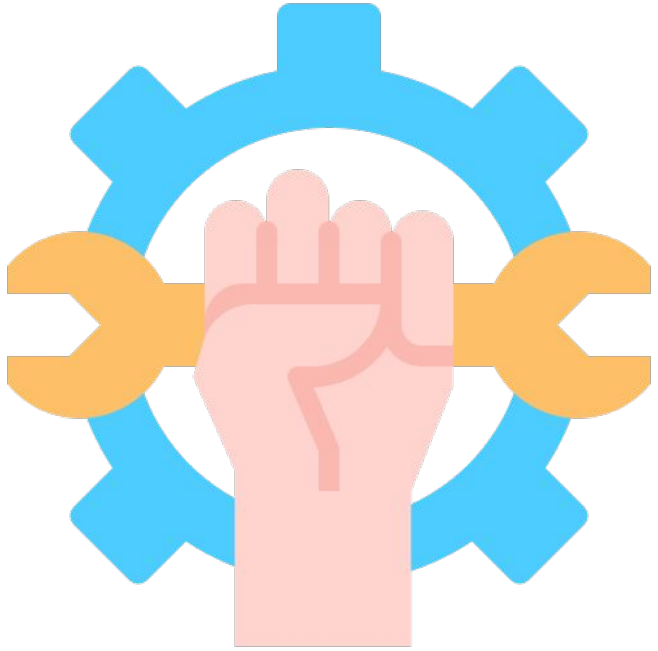
Herramientas en el mercado actual

Si bien el testing no es una competencia clave para quienes desarrollan software, consideramos que sí es importante saber de qué se trata para poder hacer las pruebas mínimas necesarias de nuestra parte, antes de que nuestro trabajo pase a una siguiente instancia.



Testing de aplicaciones Backend

La automatización de testing (Testing Automation) es una competencia muy solicitada actualmente.
Si bien no se suele aplicar en todos los casos, es bueno tener presente su concepto, entender de qué se trata, e investigar este nicho por nuestra cuenta.



Además es requisito que, quienes se dediquen a Testing Automation, tengan conocimiento y experiencia en programación. Algo que no es obligatorio para quienes se dedican al Testing Manual.

Testing de aplicaciones Backend

Y en el nicho de desarrollo de software backend, donde se involucra a Node.js y JavaScript, existen algunas herramientas para Testing Automation muy populares. Entre ellas encontramos a:

- Mocha
- Chai
- Sinon.js
- Supertest
- Istanbul
- Jest

Testing de aplicaciones Backendv

Mocha

Es un framework de pruebas ampliamente utilizado en Node.js. Proporciona una sintaxis fácil de usar y flexible para escribir pruebas unitarias y de integración. Mocha también ofrece capacidades de aserción integradas y soporte para la ejecución de pruebas tanto de forma sincrónica como asíncrona.

Testing de aplicaciones Backend

Chai

Es una biblioteca de aserciones que se utiliza comúnmente junto con Mocha. Proporciona una sintaxis expresiva y legible para realizar afirmaciones en las pruebas. Chai ofrece diferentes estilos de aserciones, como should, expect y assert, adaptándose a las preferencias de los desarrolladores.

Testing de aplicaciones Backend

Sinon.js

Es una biblioteca de pruebas que permite crear mocks, stubs y spies en JavaScript. Sinon.js es útil para simular el comportamiento de dependencias externas o para verificar las interacciones entre diferentes componentes durante las pruebas. Se integra bien con Mocha y otras herramientas de testing.

Testing de aplicaciones Backend

Supertest

Es una biblioteca que permite realizar solicitudes HTTP en las pruebas de Node.js. Supertest simplifica la realización de solicitudes a la API de tu aplicación y la comprobación de las respuestas. Puedes utilizarlo para probar fácilmente endpoints, métodos HTTP y la estructura de las respuestas.

Testing de aplicaciones Backend

Istanbul

Es una herramienta de cobertura de código para Node.js. Istanbul proporciona información detallada sobre la cobertura de pruebas, mostrando qué partes del código se ejecutaron y cuáles no durante las pruebas. Esto permite identificar áreas que no están siendo probadas adecuadamente y mejorar la calidad de las pruebas.

Testing de aplicaciones Backend

Jest

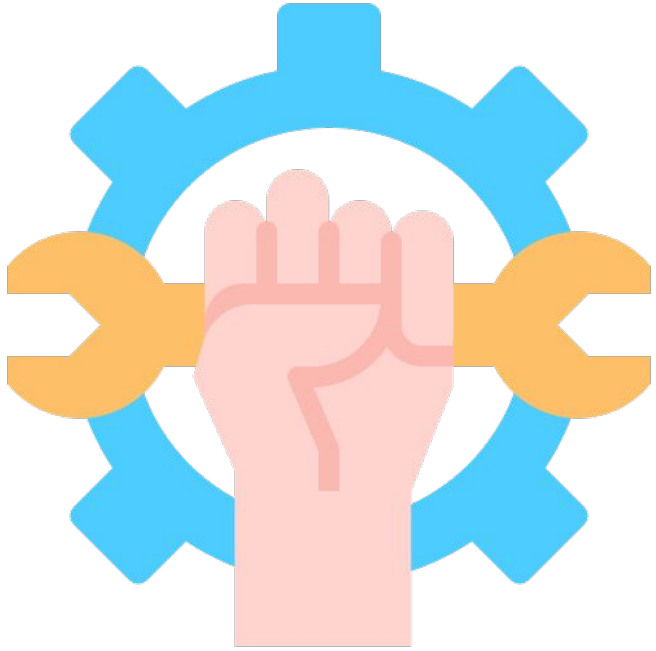
Aunque Jest se popularizó como un framework de pruebas para aplicaciones frontend en React, también es compatible con Node.js para realizar pruebas en el backend.

Jest nos brinda un modelo de configuración fácil y una amplia gama de características, incluyendo aserciones integradas, cobertura de código y ejecución paralela de pruebas.

Testing de aplicaciones Backend

Estas son solo algunas de las herramientas de testing disponibles para aplicaciones backend construidas para y con Node.js.

Más allá de que la elección de herramientas de testing, depende de cuestiones específicas del proyecto, es bueno que profundices y pruebes una o dos de estas herramientas para así aumentar tus competencias como desarrolladora de software.



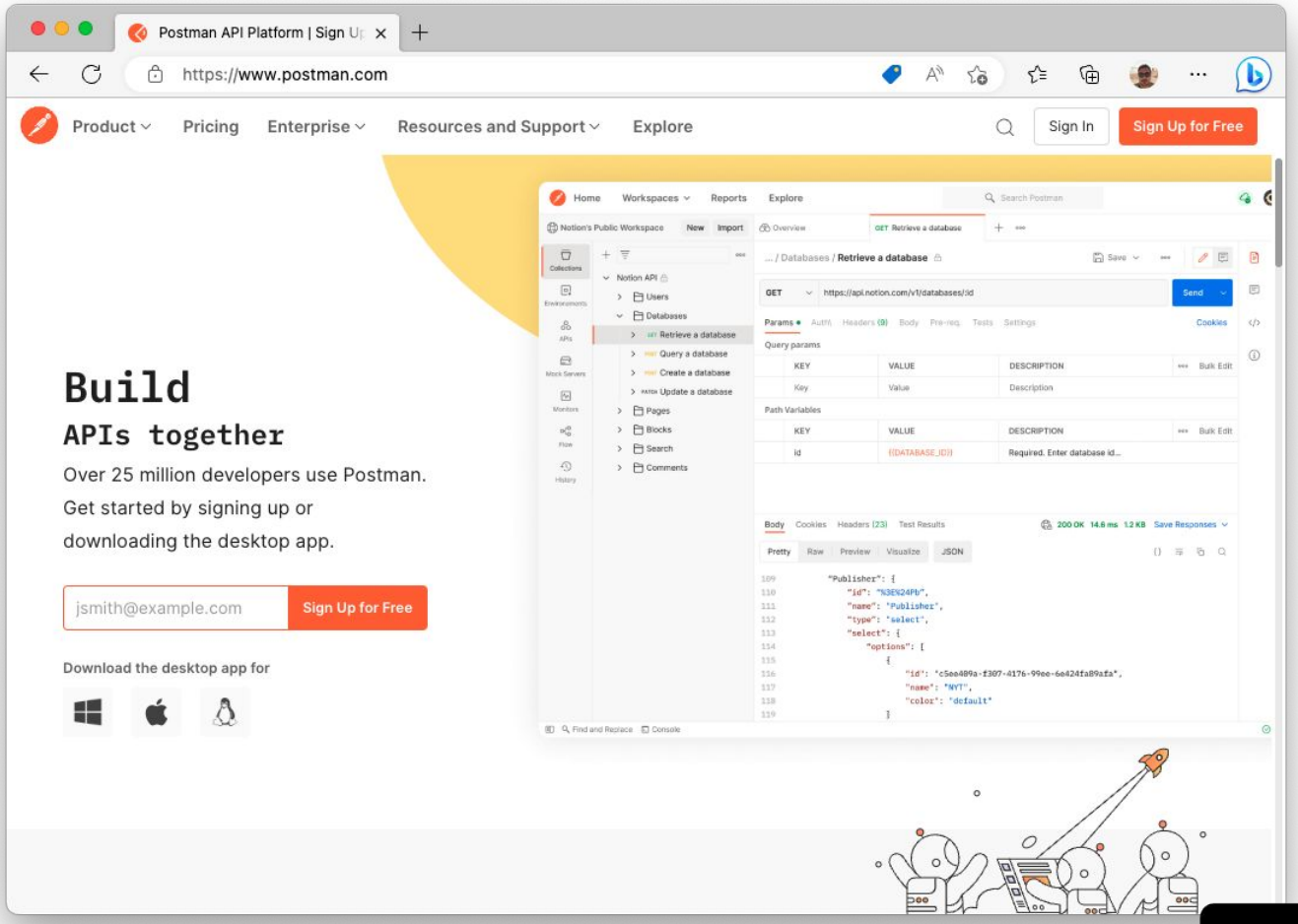
Herramientas en el mercado actual

POSTMAN

POSTMAN

Esta herramienta de testing se consolida como una de las más antiguas herramientas que nos permiten probar y documentar APIs.

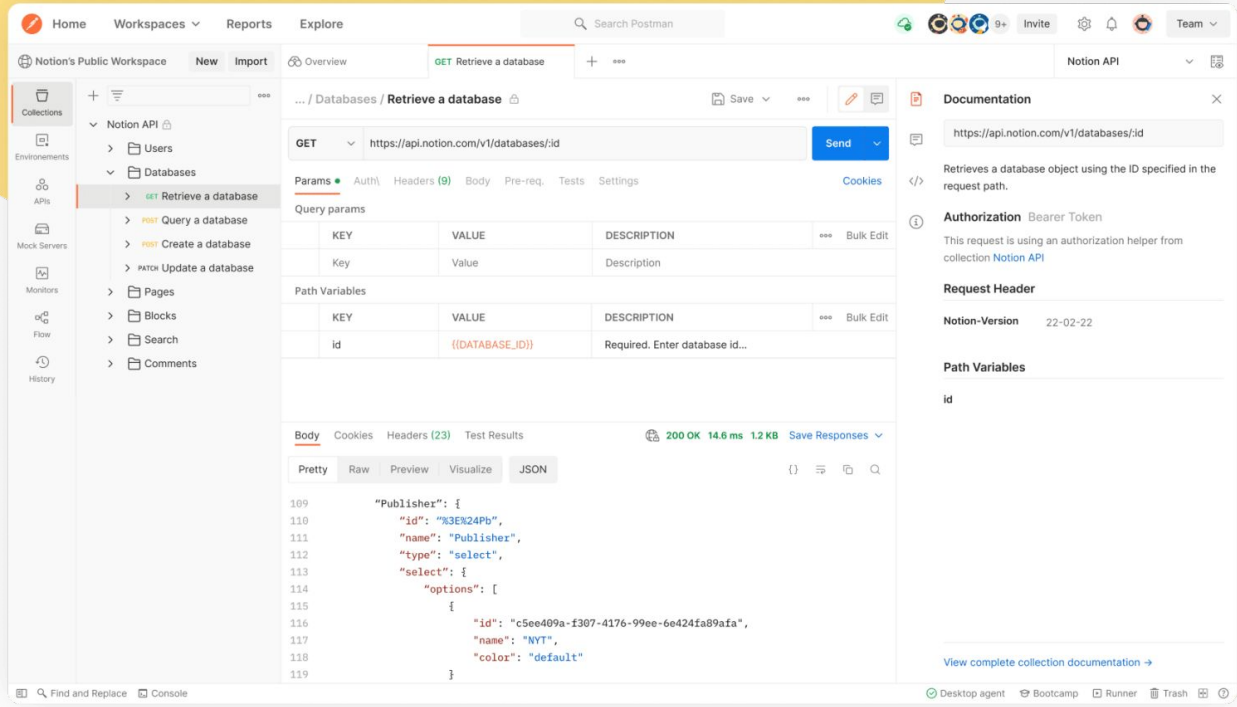
Más allá de su rol principal para testear endpoints, Posman es una plataforma de colaboración que facilita el desarrollo, prueba y documentación de APIs de manera eficiente



POSTMAN

Dentro de sus características destacables, podemos mencionar:

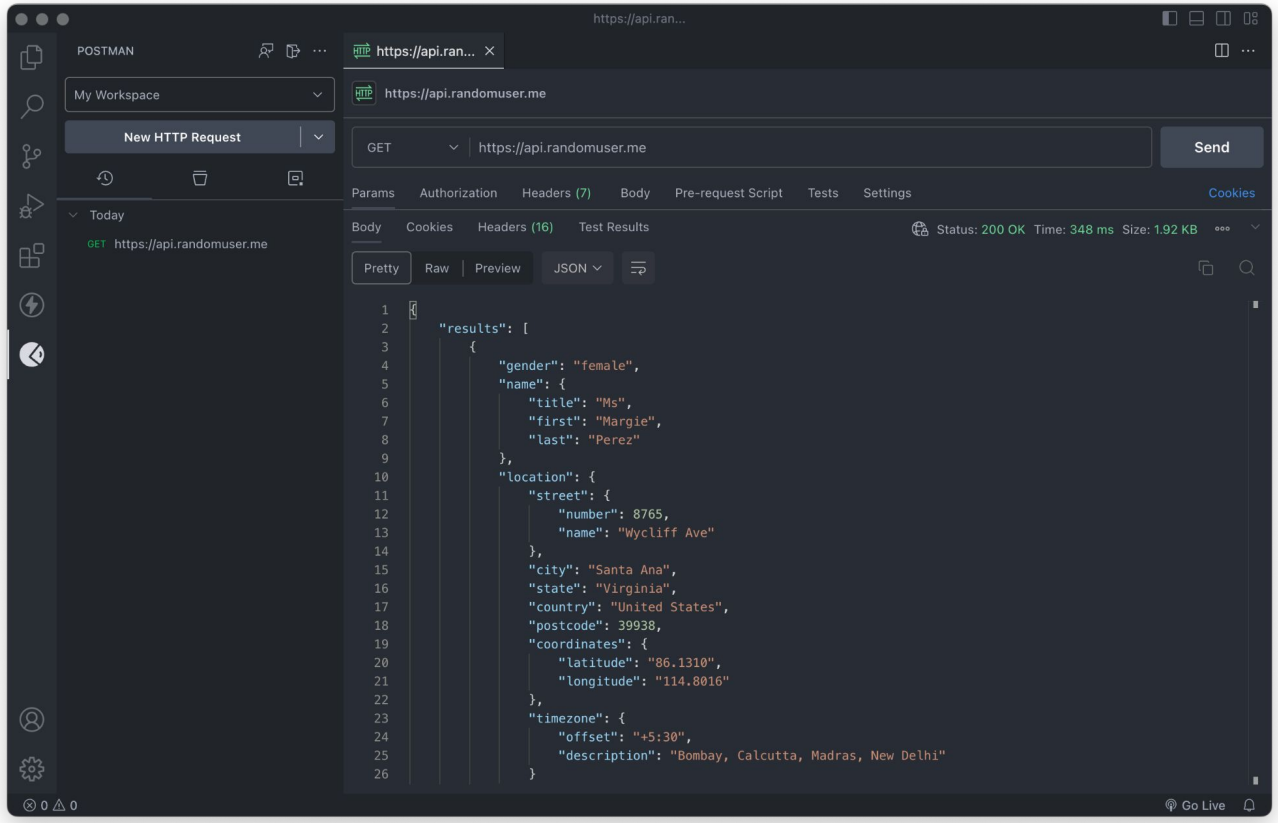
- interfaz de usuario amigable
- colecciones y entornos organizables
- automatización
- herramienta colaborativa
- generador de documentación



POSTMAN

Como novedad, hace algunos meses lanzó su propia extensión para VS CODE, por lo cual, ya podemos buscarlo e instalarlo en nuestro IDE favorito.

Ten presente que debemos registrar una cuenta para comenzar a utilizarlo.

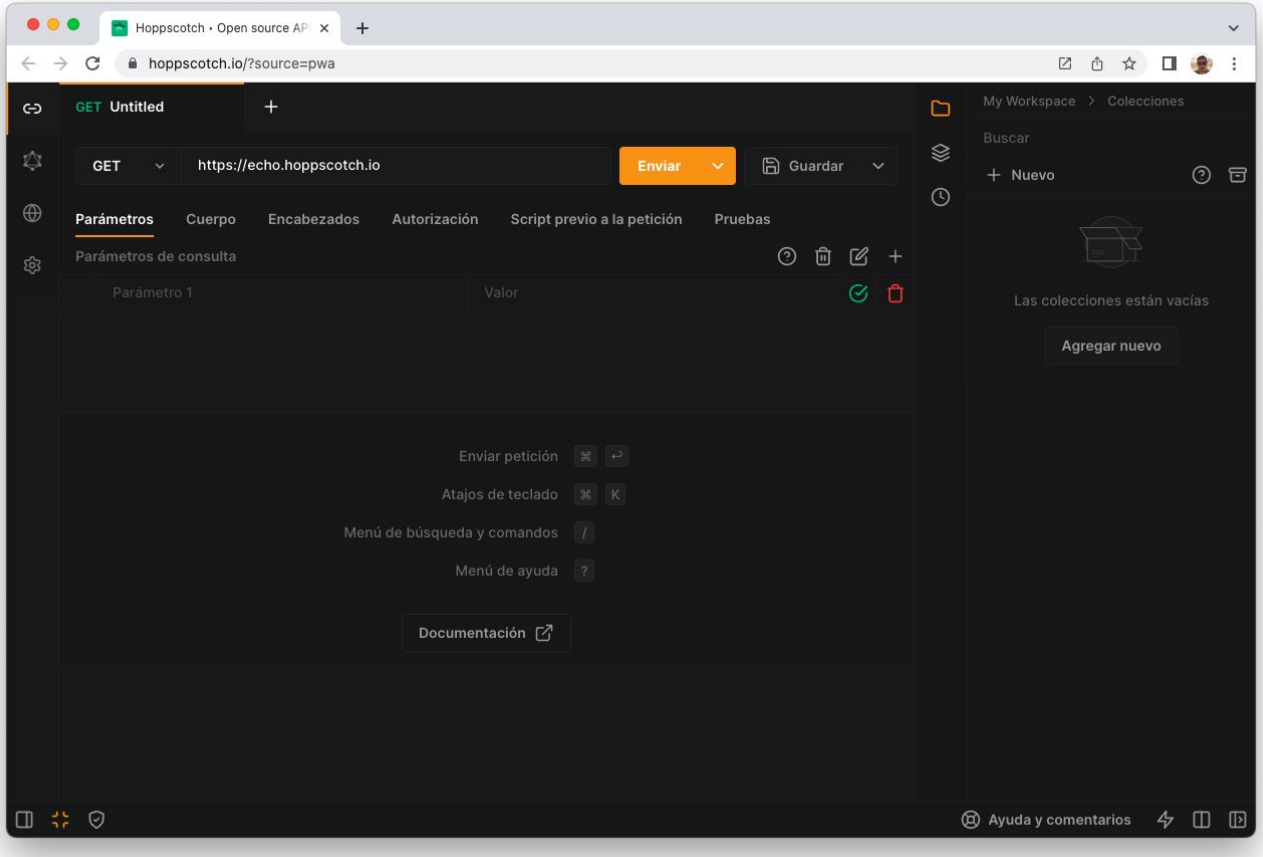


HOPPSCOTCH

HOPPSCOTCH

Hoppscotch es una herramienta de desarrollo de APIs de código abierto que permite realizar pruebas y explorar APIs de manera fácil y eficiente.

Es una alternativa popular a Postman, con una interfaz moderna y funcionalidades avanzadas.

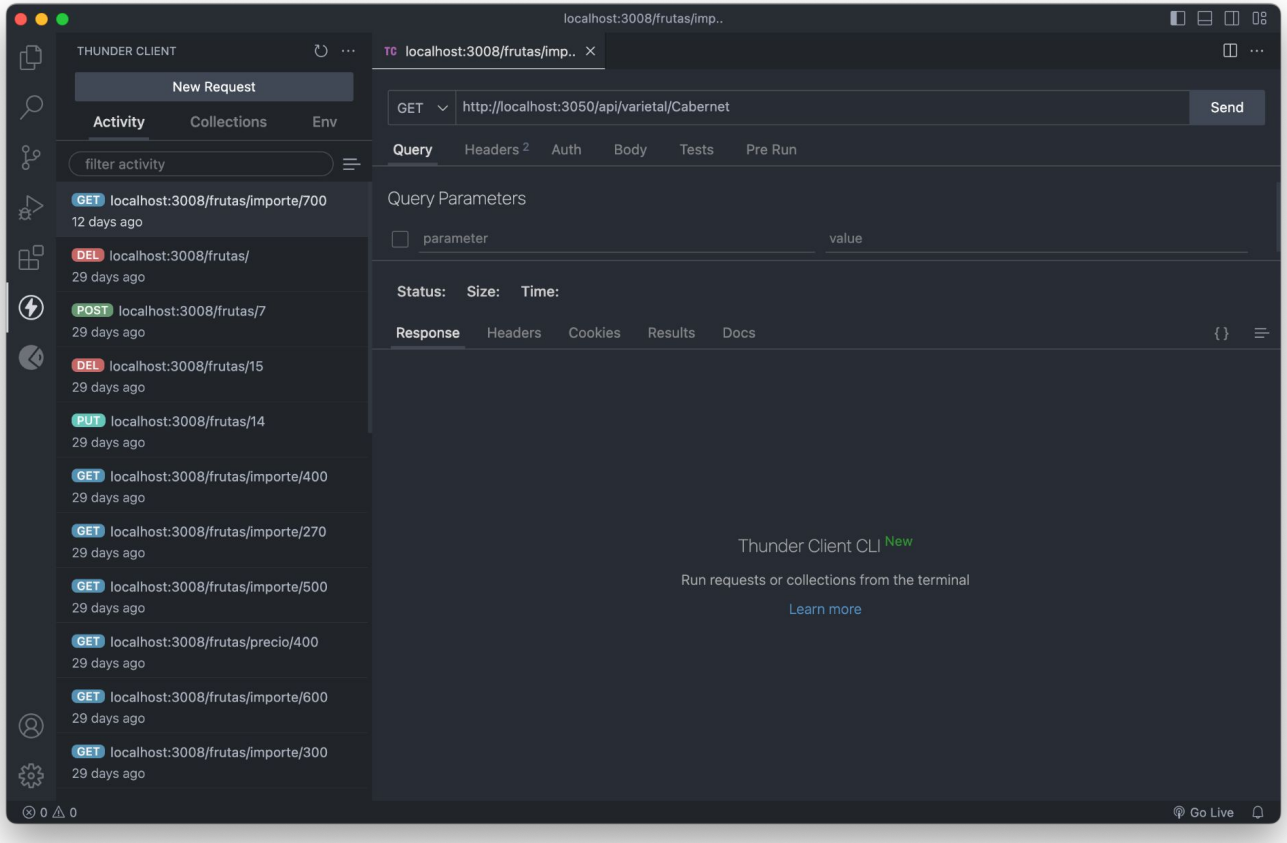


THUNDER CLIENT

THUNDER CLIENT

Ya tuvimos el placer de conocerlo y utilizarlo de forma fluída.

Thunder Client es uno de los primeros clientes de pruebas más completo, que surgió como extensión para VS CODE.



UNTREF
UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

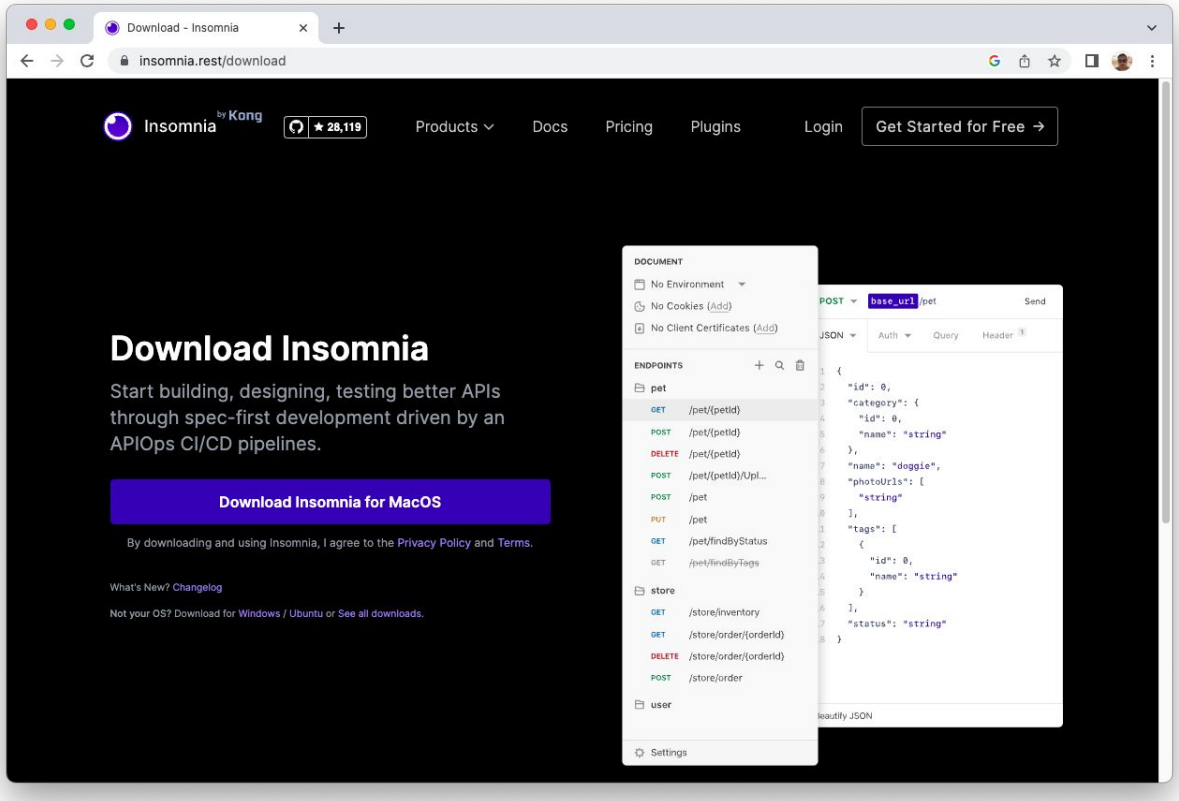
INSOMNIA

Ministerio de
Desarrollo Productivo
Argentina

INSOMNIA

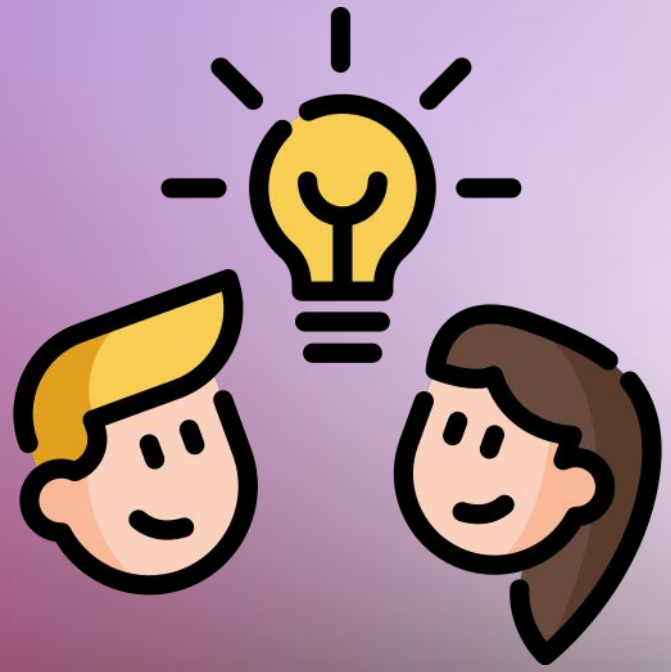
Insomnia es una herramienta de desarrollo de APIs de código abierto ampliamente utilizada por desarrolladores y equipos de desarrollo.

Proporciona una interfaz intuitiva y potente para probar y depurar APIs de manera eficiente.



Herramientas del mercado actual

Si bien nuestro plan de testing será elaborado 100% sobre Thunder Client, es importante instalar y familiarizarnos con Postman porque, al día de hoy, sigue siendo la herramienta más utilizada en el ámbito laboral.



Elaborar un plan de testing

Elaborar un plan de testing

Recuperaremos el contenido de nuestra **Clase 14 - Desarrollar un CRUD completo**, para tener un modelo ya construido y pensar en cómo elaborar un plan de testing apropiado.

Identifica el proyecto en cuestión, ábrelo con VS CODE, y veamos cómo definir un plan de testing en base a todos sus endpoints.

Testear nuestra API Restful

URL [1]	Descripción	Método
http://localhost:3008/	La URL o ruta principal	GET
http://localhost:3008/frutas	La URL general para visualizar todos los productos	GET
http://localhost:3008/frutas/:id	La URL que nos retorna un producto por su ID	GET
http://localhost:3008/frutas/nombre/:nombre	La URL que nos retorna un producto por su nombre	GET
http://localhost:3008/frutas/importe/:precio	La URL que nos retorna un producto por su precio aproximado	GET
http://localhost:3008/frutas/	La URL que nos permite dar de alta un recurso	POST
http://localhost:3008/frutas/:id	La URL que nos permite modificar un recurso existente	PUT
http://localhost:3008/frutas/:id	La URL que nos permite eliminar un recurso	DELETE

[1] En este caso, la ruta base de la API REST puede variar en nuestro entorno de pruebas, si es que el puerto definido por nosotras difiere al de los ejemplos representados en esta diapositiva.

Elaborar un plan de testing

La elaboración del plan de testing, consiste en **definir una serie de pasos a cumplir** de acuerdo a cómo se debe interactuar con cada endpoint de nuestra aplicación web.

En éste definimos:

- **qué hacer**
- **cómo proceder**
- **qué esperar como resultado**
- **qué validar de acuerdo al resultado**

Es importante también, que definamos los pasos de obtención del código fuente de nuestra aplicación backend, y de cómo configurar correctamente el ambiente de pruebas, si es que las mismas se ejecutarán en un ambiente no configurado previamente.

Veamos a continuación un documento modelo para realizar un plan de pruebas

Pre-Entrega 2

Pre-entrega 2

Para esta segunda pre-entrega deberás realizar un proyecto backend integrando la base de datos MongoDB.

Debes incluir diferentes endpoint funcionales, utilizando los métodos (GET, POST, PUT, DELETE).

Debes crear la documentación asociada que explique el funcionamiento de cada endpoint de esta aplicación de backend.

Pre-entrega 2

La temática a utilizar en la base de datos MongoDB la deberás elegir vos. Puede estar basada en un set de datos que consideres interesante, que te guste, o llame la atención.

Define la estructura del contenido de tu temática en un archivo JSON. Recuerda incluir en esta diferentes tipos de datos:

- un ID
- algún nombre, texto o descripción
- puedes sumar emojis en lugar de rutas de imágenes
- y al menos dos datos numéricos

Cuando tengas definido tu archivo JSON base y con al menos 3 o 4 documentos creados de forma manual, importa este archivo a tu base de datos MongoDB.

Pre-entrega 2

Con los datos ya importados al clúster MongoDB, comienza a elaborar al menos dos endpoint utilizando el método GET.

Ejemplo:

- Obtener todos los documentos
- Obtener un documento por su ID
- Obtener un documento por su nombre o descripción
- Define un endpoint utilizando el método POST
- Define un endpoint utilizando el método PUT
- Define un endpoint utilizando el método DELETE

Por cada endpoint que crees, recuerda controlar cualquier posible error, retornando el código de estado correspondiente a este.

Pre-entrega 2

Por último, construye un archivo con MarkDown donde se ejemplifique el uso de esta API RESTFUL, utilizando la sintaxis correspondiente de y definiendo una tabla que contenga todos los endpoint del proyecto.

Agrega al menos dos ejemplos de código en la documentación que representen la estructura de datos a consultar, de datos a agregar, modificar, o lo que consideres conveniente.

Recuerda agregar el sumario de la documentación al inicio del archivo markdown.

¡Muchas gracias!

Ministerio de Economía
Argentina

Secretaría de
Economía del Conocimiento

primero
la gente