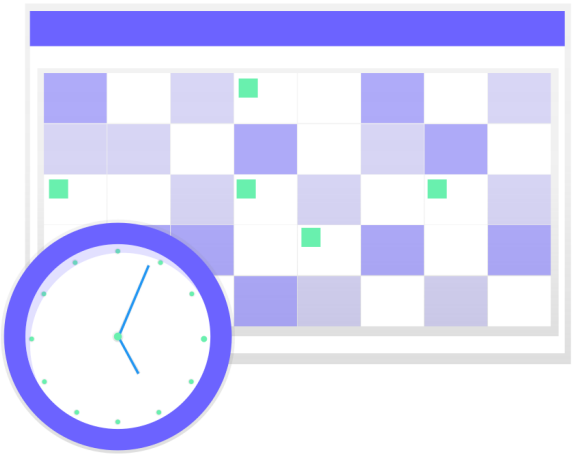
¡Hola, mundo!

> *"Juntarse es un comienzo. Seguir juntos es un progreso. Trabajar juntos es un éxito."*
>
> Henry Ford - Ingeniero y empresario.

¡Te damos la bienvenida a tu primer Sprint de Aprendizaje para el programa de Desarrollo Web Back End! ¿Qué tal te ves en este desafío que estás por emprender?

Esto que estás leyendo es tu primera toolbox: un material multimedia que ordenará tu entrenamiento individual. Aquí encontrarás lo que necesitas saber para prepararte antes de asistir a la primera meeting con tu equipo donde se retomarán los temas presentados y se pondrán en práctica.

Semana a semana, las toolboxes anticipan los temas centrales que deberás incorporar respondiendo a las siguientes preguntas: ¿de qué se trata este asunto? ¿qué relación tiene con lo que ya conoces? ¿por qué es importante? ¿para qué sirve? ¿cómo se hace?



Al final de cada toolbox, encontrarás un Challenge: es fundamental que lo resuelvas, ¡será el punto de partida para trabajar con tu equipo! A continuación, en la sección Herramientas, podrás acceder a tutoriales para descargarte los programas que necesitas para preparar tu entorno de trabajo.

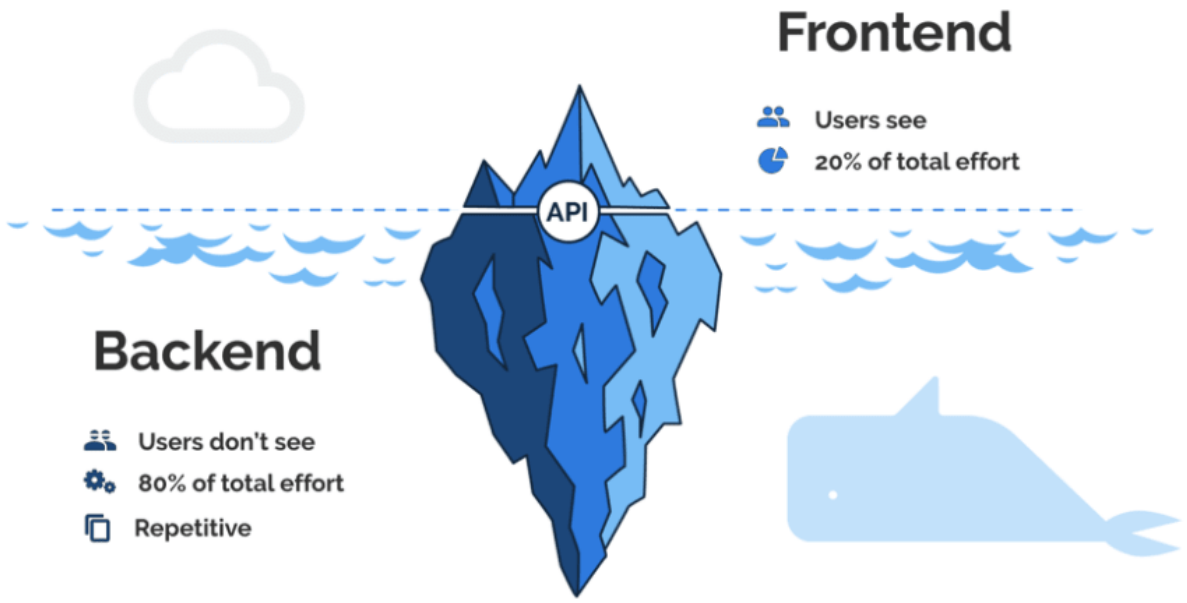
Además, en muchas ocasiones contarás con otras secciones optativas disponibles para seguir explorando. En Profundiza podrás conocer más fuentes que están hablando de cada tema; en Comunidad te mostraremos espacios de interacción en los que developers del mundo nos consultamos, recomendamos y damos a conocer herramientas y soluciones valiosas. En Potencia tu talento hallarás información relevante para tu desarrollo profesional.

Aquí, en tu primera toolbox, encontrarás una aproximación a la programación, ¿qué hacemos y en qué nos especializamos los/as developers? ¿en qué entornos trabajamos? ¿dónde alojamos cada cosa que hacemos? ¡Empecemos!

Ser o no ser developer

Un developer es el nexo entre 2 mundos: por un lado, están las personas que tienen algún tipo de necesidad con la tecnología, puede suceder a la hora de usar una aplicación en un smartphone o navegar un sitio web en una computadora, cuando utilizas el GPS de un vehículo, ¡hasta una aspiradora robot! Por el otro, está el software que provee la solución.

Allí en el medio estamos nosotras/os, las/os developers (también nos conocen como *devs*). Nos encargamos de ver, escuchar, preguntar y transformar todo eso en un software. Puedes pensarnos como traductores/as ya que ¡transformamos el lenguaje humano a un lenguaje digital!



Hay muchos tipos de developers, porque nos especializamos en distintas partes del software, o tecnología, ya que hay diversas problemáticas a resolver. En términos generales nos organizamos en 2 grandes grupos:

Developers de Front End: trabajan en la interfaz de una aplicación. Es decir, hacen que las/os usuarios tengan disponible todo lo que necesitan de la mejor manera; realizan interfaces sumamente intuitivas y con diseños alucinantes.

Developers de Back End: trabajamos en la parte que los/as usuarios no perciben. Hacemos la sigilosa tarea de almacenar su información, de generar transacciones, de proveerles seguridad, de optimizar los recursos al máximo. Aunque nadie lo note, estamos allí, ocupándonos de lo fundamental. ¡Somos el motor de una aplicación! Imagina un vehículo sin motor, podrá ser muy atractivo y contar con miles de prestaciones pero, ¿alguna de ellas puede funcionar sin un motor?

A lo largo del programa verás cómo construir ese motor y cada uno de los engranajes necesarios para que las aplicaciones puedan resolver cada una de las interacciones que presentan. Para eso, necesitarás desarrollar ciertas competencias fundamentales. Ninguna se adquiere de un día para el otro, deberás ejercitarlas en el tiempo. En Acámica te acompañaremos en el desafío. Verás que pronto empezarás a construir tus propios *done and goals*,

- **Pensamiento lógico**. Deberás ser capaz de tomar un problema y desmenuzarlo en pequeñas partes para encontrar micro-soluciones.
- **Manejo de la frustración**. Necesitarás mantener una consistencia a lo largo de cualquier desarrollo, y velar por la prolijidad a la hora de escribir tu código fuente. Pero también será fundamental que sepas que las cosas pueden fallar, de hecho sucede constantemente. Como developer tendrás que tener el compromiso de ser paciente y aprender de la frustración. Eso te permitirá construir más y mejores soluciones.

- **Creatividad e iteración**. Te encontrarás constantemente con obstáculos que deberás sortear. Muchas veces las respuestas no vienen del lado técnico y necesitarás poner en juego tu lado creativo. Deberás animarte a probar soluciones, testear y ajustar ágilmente.

- **Trabajo en equipo y comunidad.** Una persona que te ayude en el aprendizaje es sensacional, pero una comunidad es sencillamente invaluable. Verás que un mismo tema tiene muchas maneras de ser contado por diferentes developers. Lee, escucha, observa y pregunta en diferentes lugares; apóyate en tu equipo y también ten presente lo siguiente: hay una red inmensa de personas en el ecosistema digital dispuesta a interactuar.

HTML, un lenguaje del Front para aprender Back

En el mundo de la programación web hay dos tipos de lenguajes esenciales: lenguajes de marcado y lenguaje de programación. Un lenguaje de marcado nos permite codificar un documento, mientras que un lenguaje de programación nos permite escribir instrucciones las cuales serán interpretadas para generar un resultado. Es decir, a través de un lenguaje de marcado uno puede armar la estructura de una página web, y junto con el lenguaje de programación uno podrá alimentar de datos esa página web y asegurarse que la lógica detrás funcione correctamente.

HTML (*Hypertext Markup Language*) es uno de los lenguajes de marcado más populares dentro del rubro, y se utiliza, como mencionamos, para construir la estructura de todas las páginas en la web. Este es un lenguaje **descriptivo** que nos permitirá delinear la forma en que se organizará y mostrará el contenido en la página o aplicación web. Piensa en un documento HTML como si fuera un archivo Word donde solamente ordenas el contenido que quieres mostrar, como por ejemplo dónde irá ubicado el texto, dónde irán las imágenes, y así.

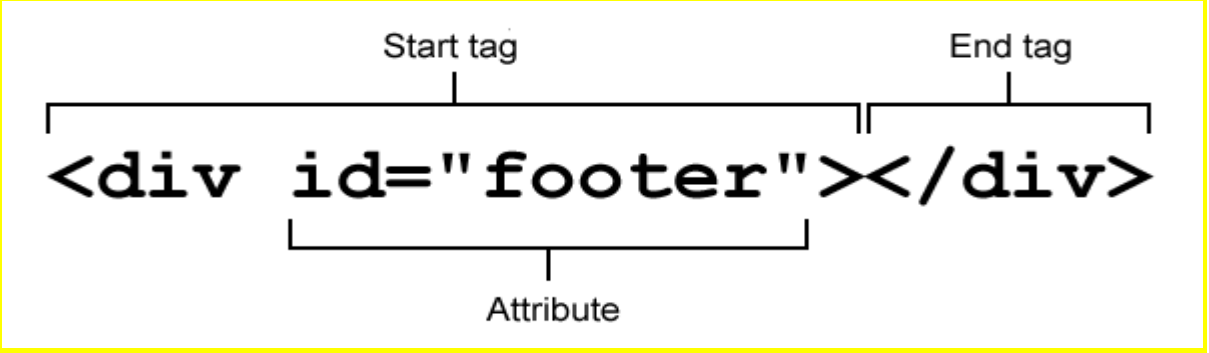


Photo on w3.org

Si bien HTML es un lenguaje utilizado mayormente por los front end developers, quienes se encargan de diseñar la interfaz gráfica de las aplicaciones web, en este programa nos servirá para visualizar aquello que 'codeamos' como parte del back end. Durante este primer sprint veremos HTML como soporte para realizar distintas pruebas y comprender la relación entre el front y el back. Usando HTML crearemos páginas sencillas para así poder enviar información al back end para que la procese y nos retorne un resultado, permitiendo que todo lo que vemos cuando interactuamos con una aplicación o sitio web funcione. ¡Veamos brevemente cómo esto se pone en acción!

Estructura básica de una página web

¿Has oído hablar de la frase "Hello, world" / "Hola, mundo"? Este es un estándar famoso, mundialmente reconocido dentro del mundo de la programación, y se utiliza para testear un entorno de desarrollo. Esta frase fue mostrada por primera vez en una aplicación de prueba por Brian Kernighan. Veamos cómo funciona.

Photo on cloudinary.com

HTML se compone de etiquetas (también llamadas *tags*) con las cuales iremos seccionando y construyendo las diferentes partes de la página web. Observa el código del primer "hola mundo" en HTML:

```
<!DOCTYPE html>'
   <html>
       <head>
           <title> Título aquí </title>
       </head>
       <body>
           Contenido del sitio web aquí.
       </body>
   </html>
```

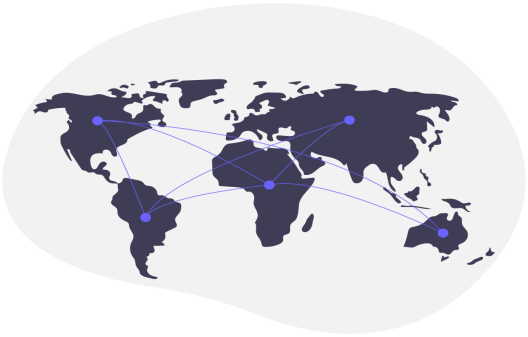
Como se ve en el código de ejemplo, un documento HTML se divide en dos partes, el *head* y el *body*. En la parte del *head* va la información de la página, elementos como el título, idioma y autor, que están dirigidas principalmente a los buscadores y navegadores. En cambio, la parte del *body* está dirigida a las personas usuarias. Como su nombre lo dice, este es el cuerpo de la página y es donde se muestra el contenido. En la próxima meeting aprenderás más en detalle para qué sirve cada uno de los tags.

Ahora, ¿te animas a armar tu propio código para ver cómo funciona? Sigue las instrucciones debajo para crear tu propio documento HTML:

1. Puedes copiarlo y pegarlo en tu bloc de notas
2. Guárdalo con la extensión .html
3. Si haces doble click en el archivo guardado, se abrirá en tu navegador predeterminado y ¡podrás ver el resultado!

¿Qué son y para qué sirven los repositorios?

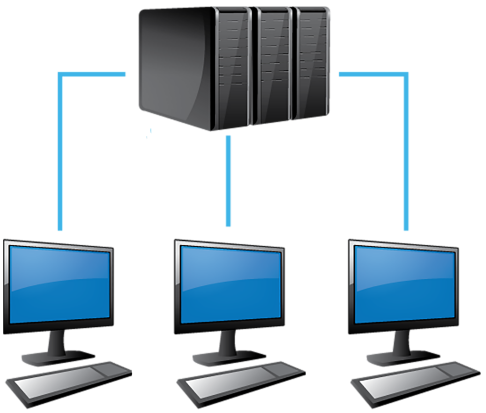
Es muy común que trabajemos junto a otras personas en un mismo proyecto. Muchas veces pueden estar junto a nosotros/as en la misma oficina, o bien de manera remota trabajando desde sus casas, ¡incluso pueden estar distribuidas por el mundo!



Es por esta razón que cuando implementamos nuestro código, es importante que dispongamos de un lugar donde se guarde de forma segura y, a su vez, que otros/as *devs* puedan acceder y colaborar en la creación del software. Para ayudarnos en esta tarea existen los repositorios.

Pero, ¿qué es un repositorio? Es un espacio centralizado donde se guarda el código del software que escribimos y en donde se puede distribuir a otros/as developers. En palabras más simples, aquí podrás subir sus proyectos y archivos, y compartirlos con tus colegas.

Usualmente los repositorios están ubicados en un servidor, lo cual permite compartir los diferentes archivos entre un amplio número de colaboradores. Un servidor es una computadora que puede almacenar gran cantidad de información (como un sitio web, fotos, emails, archivos, música o videos), y está preparado para brindar esa información a muchos clientes a la vez. No te preocupes, ¡más adelante veremos el concepto en profundidad!



Otra función muy importante de los repositorios a tener en cuenta es el control de versiones. Esto permite que muchos/as developers trabajemos en forma simultánea evitando conflictos entre los diferentes fragmentos de código que se suben, y archivando todas las diferentes versiones de los programas.

Hay varios sistemas diferentes de control de versiones, entre ellos podemos encontrar a SVN, TFVC, Mercurial y muchos más. En este programa trabajaremos con uno de los más famosos y utilizados en todo el mundo: Git. Indaguemos un poco más sobre esto.

Git

Varios años después de crear Linux, Linus Torvals se encontraba en la búsqueda de un sistema de control de versiones (VCS) que lo ayudara a trabajar con developers distribuidos/as en todo el mundo. A su vez, tenía como visión que fuera rápido, sencillo y gratuito. Luego de buscar sin mucho éxito, decidió crear su propio VCS y así nació Git.

En principio, Git nos permitirá crear nuestro propio repositorio. Éste funcionará como si fuera una carpeta compartida en el servidor donde podremos alojar todos los archivos de nuestros proyectos. Básicamente, nos permitirá vincular una carpeta ubicada en nuestro escritorio a un servicio. El flujo de trabajo comúnmente funciona de la siguiente manera:

1. Vinculamos una carpeta de nuestra computadora con el repositorio. Ten en cuenta que una vez vinculada, solamente deberás cargar y descargar los archivos que quieras modificar.
2. Agregamos los archivos necesarios del repositorio a nuestra carpeta local (imágenes, código fuente, archivos html, etc) y los editamos.
3. Preparamos los archivos agregándolos al área de preparación, o *Staging*, un área donde se alojan los archivos que están listos para subirse al repositorio.
4. Confirmamos los cambios subiendo los archivos de staging de vuelta al repositorio.

GIT es un software de uso libre, puedes inclusive crear tu propio servidor git pero existen muchos servicios de GIT que nos proveen un servidor para no tener que configurar el nuestro, los más populares son GitHub, Bitbucket y GitLab.

Durante la cursada, utilizaremos específicamente **GitLab** que además de proveerte un gestor de repositorios, el servicio ofrece también alojamiento de wikis y un sistema de seguimiento de errores, **todo ello publicado bajo una Licencia de código abierto**.

Para cerrar

Ya conoces qué y cómo trabajamos los developers. Sabes lo necesario para preparar tu entorno de trabajo. Ahora bien, antes de comenzar recuerda tres declaraciones fundamentales para maximizar tu aprendizaje:

- **Vale decir "no sé"**. Los/as mejores developers saben reconocer cuando no saben qué significa algo, qué implica, o cómo resolver ciertas situaciones.
- **Es un desafío no un problema**. Si sabes lo que tienes que 'codear' pero no sabes cómo hacerlo, no te preocupes, ¡es normal! Una situación técnica se puede resolver preguntando a otras personas o buscando en internet.
- **A programar se aprende programando**. No hay trucos, no hay secretos, simplemente es sentarse y hacerlo. Toparse con una dificultad y buscar diferentes fuentes hasta resolverla es el camino correcto. ¿No sale? ¿Te trabaste? **No te frustres**, a todos nos ha pasado, tal vez hoy no sea el día, mañana con la cabeza fresca seguro aparece la solución de la manera menos pensada.

Ahora solo queda hacerte una pregunta: **¿tienes ganas de iniciar una aventura?** Si tu respuesta es "**¡si!**" entonces te esperamos en la gran primera meeting con tu equipo. Al finalizar habrás obtenido tu primer *goal*: contarás con tus primeros archivos en línea.

¡Prepárate para tu próxima meeting!

Challenge

📝 Crea tu cuenta en GitLab. La necesitarás para subir tus primeros archivos durante la primera meeting con tu equipo. En la sección de "Herramientas" encontrarás los instructivos que necesitas.

Herramientas

🔧 Es importante que prepares tu ambiente de trabajo antes de la clase. Por eso te dejamos un listado de herramientas para que descargues e instales.

📝 Descarga Visual Studio Code, el editor de código que usaremos. ¡Lo necesitarás para poner manos a la obra durante la meeting!

📝 Descarga Git para que puedas trabajar con este tipo de repositorios (si tiene windows, recuerda reiniciar).

Profundiza

📝 Sigue estas instrucciones y crea tu primer archivo HTML.