

# CIENCIAS DE LA COMPUTACIÓN PARA EL AULA

Manual para docentes

## 2º CICLO SECUNDARIA





## MANUAL 2º CICLO DE SECUNDARIA

### AUTORES (por orden alfabético)

Claudia Banchoff  
Hernán Czemerinski  
Julián Dabbah  
Franco Frizzo  
Claudia Queiruga  
Paula Venosa

### COORDINADORA DEL EQUIPO AUTORAL

Claudia Queiruga

### COORDINADORA PEDAGÓGICA

Silvina Justianovich

### REVISORES DE CONTENIDOS

Julián Dabbah  
Herman Schinca  
Daniela Villani

### EDITORES

Ignacio Miller  
Alejandro Palermo

### FOTÓGRAFO

Facundo Manini

### DISEÑADORES GRÁFICOS

Jaqueleine Schaab  
Juan Martín Serrovalle

### ILUSTRADORA

Klinko

### COLABORADORES

Vanessa Aybar Rosales  
Néstor Castro  
Leandro Ariel Farkas  
Fernando López  
Sofia Martin  
Isabel Miyuki Kimura  
Sebastián Pacheco Véliz  
Carlos Piazza Orlando

## COLECCIÓN CIENCIAS DE LA COMPUTACIÓN PARA EL AULA

### EDITORES GENERALES

Hernán Czemerinski  
Vanina Klinkovich

### SUPERVISOR DISCIPLINAR

Franco Frizzo

### FUNDACIÓN DR. MANUEL SADOSKY

### COORDINADORES DE LA INICIATIVA PROGRAM.AR

María Belén Bonello  
Fernando Schapachnik

### DIRECTOR EJECUTIVO

Esteban Feuerstein

### PRESIDENTE

Ministro de Ciencia, Tecnología e Innovación, Roberto Salvarezza

Ciencias de la computación para el aula : 2do. ciclo de secundaria / Claudia Banchoff Tzancoff ... [et al.] ; contribuciones de Vanessa Aybar Rosales ... [et al.] ; compilado por Silvina Justianovich ; coordinación general de Vanina Klinkovich ; Hernán Czemerinski ; editado por Ignacio David Miller ; Alejandro Palermo ; fotografías de Facundo Manini ; ilustrado por Jaqueline Schaab ; Juan Martín Serrovalle ; Klinko ; prólogo de María Belén Bonello ; Fernando Pablo Schapachnik. -1a edición para el profesor - Ciudad Autónoma de Buenos Aires : Fundación Sadosky, 2019. Libro digital, PDF - (Ciencias de la Computación para el aula / Klinkovich, Vanina; Czemerinski, Hernán; 4)

Archivo Digital: descarga  
ISBN 978-987-27416-8-6

1. Informática. 2. Programación. 3. Educación Secundaria. I. Banchoff Tzancoff, Claudia. II. Aybar Rosales, Vanessa, colab. III. Justianovich, Silvina, comp. IV. Klinkovich, Vanina, coord. V. Czemerinski, Hernán, coord. VI. Miller, Ignacio David, ed. VII. Palermo, Alejandro, ed. VIII. Manini, Facundo, fot. IX. Schaab, Jaqueleine, ilus. X. Serrovalle, Juan Martín, ilus. XI. Klinko, ilus. XII. Bonello, María Belén, prolog. XIII. Schapachnik, Fernando Pablo, prolog.  
CDD 004.0712

### DISTRIBUCIÓN LIBRE Y GRATUITA



**CIENCIAS DE LA  
COMPUTACIÓN  
PARA EL AULA**

Manual para docentes

**2º CICLO SECUNDARIA**



# ÍNDICE

7	PRÓLOGO
11	INTRODUCCIÓN
21	CAPÍTULO 1: CIUDADANÍA DIGITAL Y SEGURIDAD
69	CAPÍTULO 2: PROGRAMAS Y EVENTOS
121	CAPÍTULO 3: PROCEDIMIENTOS
167	CAPÍTULO 4: ALTERNATIVAS, REPETICIONES Y VARIABLES
263	CAPÍTULO 5: REPRESENTACIÓN DE LA INFORMACIÓN
301	CAPÍTULO 6: LA COMPUTADORA
375	CAPÍTULO 7: SISTEMAS OPERATIVOS
405	GLOSARIO



# PRÓLOGO

La tarea de prologar estos manuales se asemeja, para nosotros, a la de colocar el cartel que dice “bienvenidos” en la puerta de un edificio de varios pisos. Este gesto, que es final e inaugural a la vez, corona años (literalmente hablando) de duro trabajo y, a la vez, anticipa la espera ansiosa de la etapa siguiente: su uso en las aulas.

Se trata de los **primeros manuales escolares sobre Ciencias de la Computación en el escenario editorial argentino, que se ponen a disposición del público de manera libre y gratuita**, y que se suman a un pequeño grupo de ejemplos pioneros a nivel mundial en esta temática. Somos conscientes de que, al hablar de “manuales escolares sobre Ciencias de la Computación”, queda mucho por aclarar. Comencemos por el principio: ¿por qué Ciencias de la Computación?

**Ciencias de la Computación es el nombre que recibe el área del conocimiento que aporta una serie de saberes** (programación, funcionamiento de las computadoras e Internet, inteligencia artificial, etc.) **que resultan fundamentales para comprender el mundo cada vez más tecnológico en el que viven y se desarrollan los alumnos que transitan su escolaridad hoy en día**. Sin estos conocimientos, su comprensión de la realidad se verá limitada, y no podrán participar como ciudadanos activos e informados en los debates actuales sobre las múltiples interacciones entre la tecnología informática y la sociedad. Argentina ha decidido avanzar sobre esta materia y es por eso que **el Consejo Federal de Educación declaró, mediante su resolución 263/15, que la enseñanza y el aprendizaje de programación es de importancia estratégica para fortalecer el desarrollo socioeconómico de la nación**.

Estos manuales **se concibieron para el aula**, como una herramienta para el docente, al que le brindan secuencias didácticas detalladas y fichas de trabajo para entregar a sus estudiantes. ¿Para qué aula, para qué docentes, para qué estudiantes? En principio, estas actividades están pensadas para el aula argentina. Este material fue **escrito en su totalidad por y para argentinos y argentinas**, tomando como referencia la realidad de la escuela argentina. Esto se refleja en el lenguaje, en las referencias y en los marcos culturales que se utilizan, características que no impiden que incentivemos su uso en otros países de la región y del mundo.

Los cuatro manuales que componen esta colección **cubren, respectivamente, el primer ciclo de la educación primaria, el segundo ciclo de la educación primaria, el primer ciclo de la educación secundaria y el segundo ciclo de la educación secundaria**. El rango etario determina, en cada caso, el recorte de temas, la profundidad con que se abordan, la complejidad del texto y la línea estética. En su mayoría **son manuales iniciales** (es decir, tres de ellos están concebidos para alumnos y alumnas que dan sus primeros pasos en Informática, a distintas edades). El manual destinado al segundo ciclo de la escolaridad primaria tiene como antecedente nuestro primer manual para programar en el aula.<sup>1</sup>

En cuanto a sus destinatarios principales, los docentes, estos manuales buscan interpelar a un conjunto de profesionales cuyas formaciones en el área son heterogéneas. Es por esto que, sin pretender presentar explicaciones teóricas exhaustivas, a lo largo de los distintos capítulos hay desarrollos conceptuales que contribuyen a que aquellos y aquellas docentes que no poseen un dominio fluido de ciertos temas puedan contar con las nociones fundamentales. Al respecto, vale destacar que la Fundación Sadosky ha capacitado hasta abril de 2018 (a través de convenios con universidades públicas de todo el país) a más de 1500 docentes, que se suman a los que han formado los ministerios de educación nacional y provinciales. Estos docentes encontrarán particular provecho en nuestro material.

Aunque gran parte del material fue testeado y consultado con los y las docentes de primaria y secundaria que tomaron nuestros cursos, somos conscientes de que solo su uso de manera sistemática en la escuela permitirá mejorarlo.

Asimismo, resulta pertinente aclarar que estos manuales **están pensados para abordar contenidos de Ciencias de la Computación en espacios disciplinares específicos**. Distintas jurisdicciones del país (Neuquén, CABA, Tucumán, entre otras) cuentan con estos espacios curriculares, mientras que otras están discutiendo su incorporación. Desde Program.AR entendemos que este material constituye un aporte a ese camino, que se suma a las Planificaciones anuales para Tecnología de la Información de 3<sup>er</sup> y 4<sup>to</sup> año de CABA<sup>2</sup> publicadas anteriormente.

En cuanto al enfoque didáctico, las secuencias propuestas están pensadas, en buena parte, desde la **perspectiva del aprendizaje por indagación**. Imaginamos que los manuales serán usados por docentes y estudiantes que transitan un camino de descubrimiento, asociado a la tecnología informática que media en buena parte de nuestras interacciones con el mundo.

<sup>1</sup> Manual de actividades para Program.AR, disponible en <http://program.ar/manual-docentes-primaria/>

<sup>2</sup> Disponibles en <http://program.ar/planificacion-anual-ti3/> y <http://program.ar/planificacion-anual-ti4/>.

Vale la pena ahondar en el proceso que hoy encuentra un hito en la publicación de estos manuales. Al pensar de qué manera debía construirse este material y quiénes debían ser los encargados de hacerlo, recurrimos a la herramienta que más garantías ofrece en términos de calidad, apertura y transparencia: **se realizó una convocatoria pública a las universidades que componen el sistema de generación de conocimiento de nuestro país.** Como resultado de esa convocatoria, y mediante la evaluación de un jurado internacional, resultaron elegidas cuatro universidades nacionales que formaron equipos autorales compuestos por profesionales de las Ciencias de la Computación y de Educación.

Los manuales fueron escritos por colegas de la **Universidad Nacional del Centro de la Provincia de Buenos Aires** (primer ciclo de primaria), la **Universidad Nacional de Córdoba** (segundo ciclo de primaria), la **Universidad Nacional de Quilmes** (primer ciclo de secundaria) y la **Universidad Nacional de La Plata** (segundo ciclo de secundaria), con quienes estamos profundamente agradecidos por su compromiso y profesionalismo en un proceso que fue novedoso para todos los involucrados. Cada manual tiene una impronta propia, a través de la cual se traducen las diferentes miradas, prioridades y valoraciones que los distintos colegas otorgan a diversos aspectos de las Ciencias de la Computación.

Deseamos destacar la labor de todo el **equipo de la Fundación Sadosky** que participó en el desarrollo de estos manuales. Estas palabras apenas resumen un proceso que implicó años de trabajo y, por sobre todas las cosas, el compromiso de todos los que lo hicieron posible. En primer lugar, agradecemos a Hernán Czemerinski y Vanina Klinkovich, editores generales de la colección, quienes realizaron un gran trabajo dirigiendo este proyecto desde su génesis en 2016 y se ocuparon, además, de revisar, corregir, unificar y armonizar todas las visiones.

Sumamos a este agradecimiento a Franco Frizzo, que ha supervisado el contenido de toda la colección, y a Jacqueline Schaab, a cargo del diseño gráfico. Queremos también destacar la participación del conjunto de revisores, desarrolladores y gestores de la Fundación Sadosky que participaron en este proyecto (por orden alfabético): Julián Dabbah, Pablo Factorovich, Mariana Labhart, Alfredo Sanzo, Herman Schinca, Daniela Villani. Sin su colaboración, estos manuales no serían una realidad.

Dedicamos un párrafo especial al querido Alfredo Olivero, a la vez mentor y compañero de ruta, de un humor tan incisivo como su inteligencia, que fue parte de este equipo y a quien extrañamos mucho.

Asimismo, no queremos dejar de agradecer al equipo de Legales, Administración y Gestión de la Fundación Sadosky, compuesto por Roxana Ríos, Andrea Córdoba, Rosa Córdoba, Mariano Tiseyra, Melina Rodríguez y a su Director Ejecutivo, Esteban Feuerstein. Queremos también expresar nuestro agradecimiento a Santiago Ceria, quien ocupaba la Dirección Ejecutiva al momento de comenzar este proyecto.

Al mismo tiempo, fue necesario contar con los servicios de profesionales encargados de la edición y corrección de los textos, el diseño y la ilustración. Es por esto que también agradecemos al equipo cuyo talento permitió tener este material en su forma actual: Ediciones Colihue, Ignacio Miller, Alejandro Palermo, Luciano Andújar, Celeste Maratea y Luz Rodríguez.

En cuanto al **uso del género gramatical en esta colección**, hemos decidido respetar la norma vigente del uso del masculino genérico para grupos mixtos. Conscientes de que esta decisión deja pasar una oportunidad para contribuir a la construcción de una norma más inclusiva, optamos por un texto que resultara menos disruptivo.

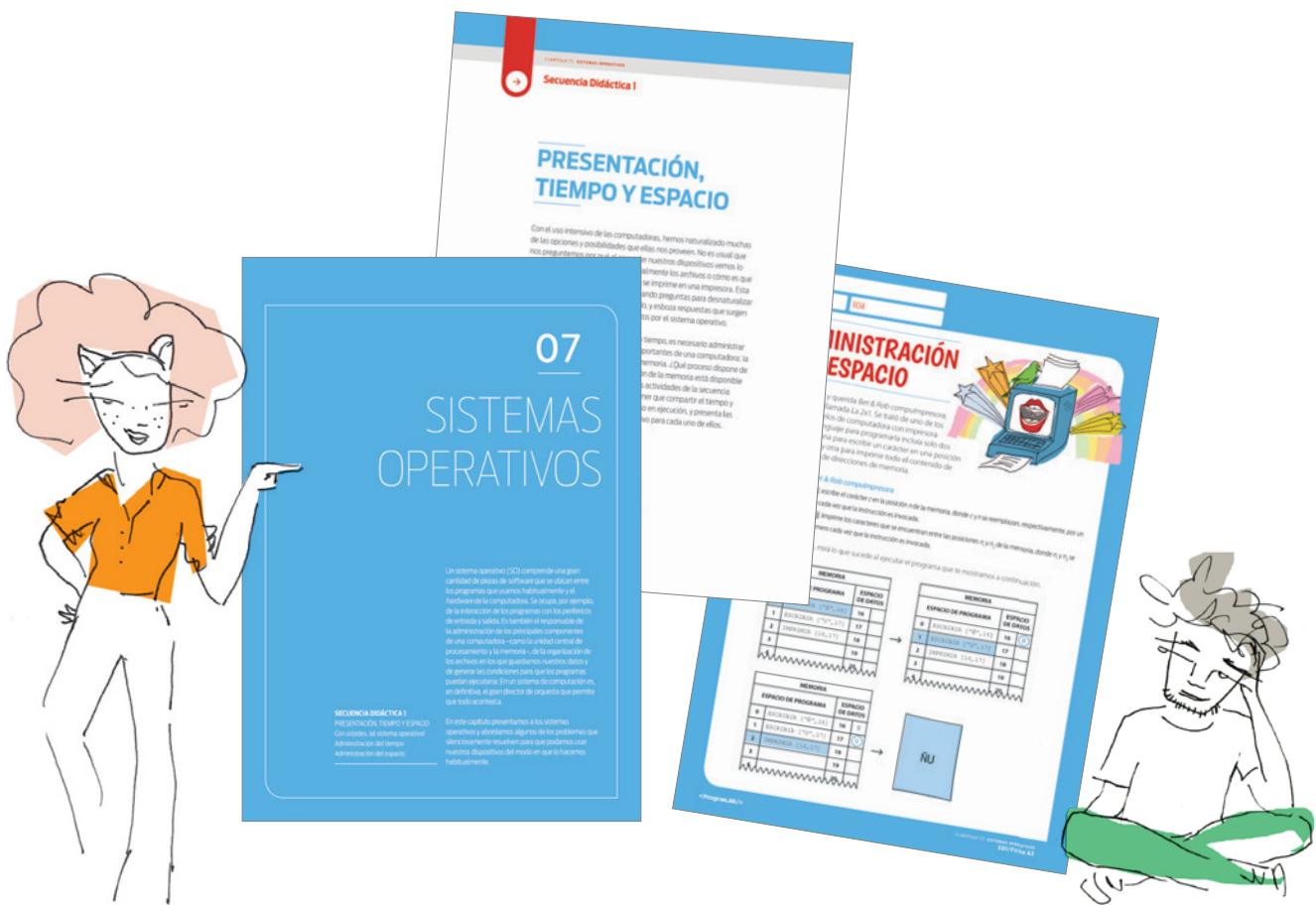
Esta colección se pone a disposición del público con **licencia Creative Commons**,<sup>1</sup> como una forma de incentivar la creación de obras derivadas. Dicho de otra forma, fomentamos activamente que las y los colegas generen sus propias versiones de este material y las compartan con la comunidad.

Estos manuales son para nosotros una versión 1.0 que, con un fuerte anclaje en el aula, **no deja de tener carácter experimental**. Muchos de los temas abordados cuentan con pocos antecedentes en la bibliografía internacional: existe mucho material para enseñar programación inicial a niños y adolescentes, mucho menos sobre cómo funciona una computadora, y muy poco sobre otras cuestiones tales como el funcionamiento de Internet, la representación de la información o la inteligencia artificial. Sabemos también que algunos temas que merecerían tener un lugar han quedado afuera, algo que pensamos subsanar en próximas ediciones.

Nos encantaría enriquecer este material con los aportes de la comunidad: docentes, académicos, investigadores e interesados en la temática en general están invitados a acercarnos sus comentarios, críticas y sugerencias escribiendo a [info@program.ar](mailto:info@program.ar). A su vez, queda abierta la invitación a revisar periódicamente nuestro sitio web o seguirnos en las redes sociales, para mantenerse al tanto de futuras versiones.

María Belén Bonello y Fernando Schapachnik  
Coordinadores de la Iniciativa Program.AR  
Fundación Dr. Manuel Sadosky

<sup>1</sup> Específicamente, una licencia Creative Commons Atribución-No Comercial-CompartirIgual 4.0 Internacional, cuyos detalles pueden consultarse en <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>.



# INTRODUCCIÓN

Ciencias de la Computación es el nombre de un área del conocimiento que engloba un conjunto de saberes, tales como la programación, el funcionamiento de los sistemas operativos, la representación de la información o la inteligencia artificial, entre otros. Se trata de una disciplina que, si bien es relativamente joven, resulta fundamental tanto para comprender un mundo cada vez más atravesado por la tecnología, como para estar en condiciones de participar activamente en debates actuales sobre la interacción entre la tecnología, la informática y la sociedad.

Al contrario de lo que suele creerse, la disciplina no se centra en el estudio de la tecnología, sino que estudia las bases y los fundamentos sobre los que esta se monta. Muchos de los conceptos e ideas que subyacen a todos los dispositivos computacionales se concibieron incluso antes de que existieran las computadoras. En consecuencia, el aprendizaje de esta materia no está amenazado por los vertiginosos avances tecnológicos, sino que, por el contrario, su valor consiste en que brinda las herramientas para poder adaptarse a ellos sin mayores inconvenientes.

## OBJETIVO

Aun cuando existe cierto consenso sobre la importancia de enseñar Ciencias de la Computación a los niños y los jóvenes, todavía faltan algunas condiciones objetivas para que la enseñanza se produzca en forma sistemática. Por ejemplo, no existe un conjunto suficiente de docentes formados en la disciplina ni contenidos organizados para su puesta en práctica en las aulas. Esperamos que este manual contribuya a dar una solución a estos dos problemas.

El objetivo de este manual es brindar apoyo a los docentes del segundo ciclo del nivel secundario para enseñar algunos conceptos de las Ciencias de la Computación. Cada capítulo aborda un área temática de la disciplina que consideramos importante para la comprensión del mundo intensamente tecnológico en el que vivimos.



Sistemas de computación



Redes e Internet

Representación  
y manipulación de datos

Algoritmos y programación



Privacidad y seguridad informática

Se proponen actividades, organizadas en secuencias didácticas, con una descripción detallada de cómo ponerlas en práctica en el aula. Muchas de ellas están acompañadas de una ficha para entregar a los estudiantes, que incluye consignas e información relativa al tema abordado en cada actividad.

Además, no es necesario que los docentes cuenten con conocimientos disciplinarios previos, ya que el manual acompaña al lector en el aprendizaje de la asignatura y brinda marcos conceptuales sobre los que apoyarse.

## ACTIVIDADES

Las actividades del manual pueden dividirse en dos grandes grupos: las que requieren una computadora y las que no. Estas últimas, a las que llamamos “actividades desenchufadas”, exploran conceptos y proponen ejercitar técnicas cuya puesta en práctica no depende de un dispositivo tecnológico: los estudiantes juegan con cartas para aprender números binarios, practican el clásico juego Piedra, papel o tijera para descubrir las alternativas condicionales y cantan canciones para comprender qué son los

procedimientos y los parámetros. Estas actividades se combinan en el manual con ejercicios de programación para proveer una rica experiencia a los estudiantes.

Para las actividades de programación que requieren una computadora, se propone el uso de MIT App Inventor 2, un entorno de programación visual de código abierto y uso libre y gratuito que permite crear aplicaciones para teléfonos inteligentes y tabletas que tengan instalado el sistema operativo Android. Se trata de una herramienta basada en bloques, lo que facilita la creación de aplicaciones para aquellos que no están familiarizados con el mundo de la programación.

La elección de App Inventor se debe a que los estudiantes del segundo ciclo de secundaria hacen un uso intensivo de teléfonos inteligentes; por lo tanto, les resulta muy atractivo desarrollar aplicaciones que finalmente puedan ejecutar en sus dispositivos.

Es importante tener presente que este libro no es un manual de App Inventor. Esta es la herramienta elegida para ilustrar algunos conceptos fundamentales de la programación, pero estos son comunes a casi cualquier otro lenguaje. Además, no es nuestro objetivo introducir todas las características de App Inventor, sino solo aquellas partes que tiene en común con la mayoría de los lenguajes de programación.

MIT App Inventor 2 es una aplicación web, por lo que para usarla es necesario contar con una conexión a Internet y un navegador. Sin embargo, distintos grupos de desarrolladores de software han construido entornos que permiten el uso *offline* de App Inventor.<sup>1</sup> Por lo tanto, las instituciones educativas sin conectividad podrán poner en práctica las actividades propuestas en este manual. Además, en los anexos que complementan este libro, se ofrecen actividades basadas en los lenguajes Alice, Python y RITA, que exploran los mismos conceptos y no requieren de conectividad.

## EQUIPO INTERDISCIPLINARIO

El desarrollo del manual fue íntegramente realizado por un equipo interdisciplinario de la Universidad Nacional de La Plata (UNLP), compuesto por profesionales del área de Ciencias de la Computación y de la de Ciencias de la Educación, en conjunto con el equipo de la iniciativa Program.AR de la Fundación Sadosky.

## PROPIUESTA PEDAGÓGICA

La propuesta pedagógica del manual se enmarca dentro de lo que se conoce como *aprendizaje por indagación*. A partir de la presentación de preguntas y problemas que les resulten significativos, los estudiantes tendrán que encontrar respuestas y soluciones sin que previamente hayan recibido una explicación de índole teórica de todos los elementos necesarios para poder resolver la consigna. Serán los estudiantes quienes de forma activa construyan el conocimiento, en contraposición a lo que sucede cuando, en una clase, escuchan pasivamente lo que explica un docente. De este modo, el proceso de conceptualización se produce más como una conclusión del aprendizaje que como un punto de partida.

<sup>1</sup> Algunas opciones disponibles para el uso del entorno fuera de línea se encuentran documentadas en <http://bit.ly/2ZwuznN>.

Las propuestas plantean problemas que, una vez abordados, permiten alcanzar una nueva comprensión de las ideas que se ponen en juego y sus implicaciones en la realidad. El modo de presentación de las actividades es variado: un juego o un desafío, el planteo de situaciones hipotéticas o un simple enunciado, por ejemplo. En este contexto, el rol del docente consistirá en guiar a los estudiantes con preguntas, analogías y referencias a otras experiencias previas que hayan hecho, para poder pensar las soluciones.

La evidencia indica que esta metodología facilita en gran medida la apropiación del conocimiento por parte de los estudiantes porque implica la construcción de saberes mediante un proceso, sin imposiciones externas que deben fijarse de memoria y que no están relacionadas con ninguna experiencia significativa para ellos. Aunque se les da parte de la información, esta solo puede completarse a partir de la búsqueda propia, la elaboración, el análisis y la argumentación.

### ¿QUÉ CONTENIDOS ABORDA CADA CAPÍTULO?

Cada capítulo de este manual está integrado por una serie de secuencias didácticas que tienen una coherencia conceptual. Cada secuencia didáctica está formada por actividades que pueden o no requerir el uso de una computadora. A su vez, cada actividad propone formas de abordar la clase, un desarrollo y un cierre. Además, muchas están acompañadas de una ficha para entregar a los estudiantes.



### CAPÍTULO 1: CIUDADANÍA DIGITAL Y SEGURIDAD

El mundo tecnológico atraviesa cada vez más aspectos de la vida cotidiana. La conectividad establece novedosas formas de vincularnos con el mundo que, hasta hace poco, solo existían dentro de los límites de la ciencia ficción: realizamos videoconferencias con personas en distantes lugares del mundo mientras paseamos a un perro, disponemos de la discografía completa de nuestros músicos favoritos con solo presionar un botón que, en realidad, solo existe en una pantalla, etc.

Esta nueva realidad impone nuevas reglas de juego, nos enfrenta a nuevos desafíos y nos expone a ciertos riesgos sobre los cuales debemos ser conscientes. A lo largo de este capítulo se exploran diversos aspectos de ciudadanía digital, la privacidad de la información y los métodos seguros para el resguardo y envío de datos en Internet.

---



## CAPÍTULO 2: PROGRAMAS Y EVENTOS

Un programa es, ante todo, una descripción muy precisa de cómo esperamos que se comporte una máquina. En general, se trata de una colección de instrucciones escrita en un lenguaje de programación que, a diferencia de un lenguaje coloquial, posee una sintaxis muy estricta y una semántica que no admite múltiples interpretaciones.

En este capítulo se introduce la noción de programa, y se hace especial énfasis en aquellos en los que existe una interacción con el usuario. Las secuencias didácticas llevarán a los estudiantes a construir sus primeras aplicaciones, comprender qué son los eventos, cómo se programa una respuesta frente a ellos y la importancia de las interfaces gráficas.



## CAPÍTULO 3: PROCEDIMIENTOS

Un procedimiento es una porción de un programa más grande que encapsula una tarea específica. Los procedimientos suelen usarse para descomponer problemas complejos en piezas más simples que, al combinarlas, resuelven el problema original. Bien utilizados, mejoran notablemente la legibilidad de los programas.

Las secuencias didácticas de este capítulo introducen cómo definir y usar procedimientos. Las actividades proponen la reflexión sobre la redundancia en los programas y muestran de qué manera puede evitarse mediante el uso de procedimientos. Además, presenta el uso de parámetros para obtener soluciones generales que resuelven distintos problemas particulares.



## CAPÍTULO 4: ALTERNATIVAS, REPETICIONES Y VARIABLES

Las condiciones en las cuales se ejecutan los programas suelen ser desconocidas a la hora de programar. Por lo tanto, debemos considerar distintos escenarios y las acciones adecuadas a realizar en cada caso. Los lenguajes de programación permiten expresar alternativas condicionales, que indican que algunas instrucciones solo tienen que ejecutarse en algunos casos, según se cumpla o no cierta condición.

Por otro lado, hay problemas cuya resolución requiere repetir, en forma consecutiva, una serie de acciones. En este capítulo mostraremos cómo construir programas usando repeticiones, que son comandos que permiten expresar la reiteración de acciones sin necesidad de escribir las varias veces.

Por último, es frecuente que los programas requieran guardar información, recuperarla y modificarla. Para ello existen las variables, que abstraen un espacio de la memoria de la computadora para almacenar información.



## CAPÍTULO 5: REPRESENTACIÓN DE LA INFORMACIÓN

Las computadoras son máquinas que manipulan información. Nos permiten ver, escuchar, crear y editar información: escribir un documento, editar imágenes, ver y grabar videos o escuchar música, entre otras actividades. Por ello resulta fundamental poder representar la información de alguna manera dentro de la memoria RAM, en el disco o para enviarla a través de una red.

En este capítulo, se presentan actividades que trabajan sobre las ideas subyacentes usadas por los sistemas digitales para la representación de números, texto e imágenes.

---

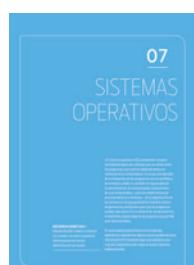


## CAPÍTULO 6: LA COMPUTADORA

Los programas son descripciones sin ambigüedades del comportamiento que se espera que pueda tener una máquina. En rigor, se trata de piezas intangibles cuya existencia adquiere sentido solo porque hay una contraparte capaz de ejecutarlas: el *hardware*. Con este término hacemos referencia al conjunto de componentes físicos que forma parte de una computadora.

El universo del *hardware* abre la posibilidad para que nos hagamos muchas preguntas: ¿Qué es una computadora? ¿Cómo son sus componentes, qué funciones cumplen y cómo interactúan? ¿Qué cualidades definen el desempeño de un sistema de computación? ¿Cómo es que el *hardware* ejecuta efectivamente los programas que escribimos? En este capítulo se proponen actividades que buscan contribuir a que los estudiantes esbozen respuestas a estas y otras preguntas sobre la computadora.

---



## CAPÍTULO 7: SISTEMAS OPERATIVOS

Un sistema operativo (SO) comprende una gran cantidad de piezas de *software* que se ubican entre los programas que usamos habitualmente y el *hardware* de la computadora. Se ocupa, por ejemplo, de la interacción de los programas con todos los periféricos de entrada y salida. Es también el responsable de la administración de los principales componentes de una computadora –como la unidad central de procesamiento y la memoria–, de la organización de los archivos en los que guardamos nuestros datos y de generar las condiciones para que los programas puedan ejecutarse. En un sistema de computación es, en definitiva, el gran director de orquesta que permite que todo acontezca.

En este capítulo abordamos algunos de los problemas que silenciosamente resuelve el sistema operativo para que podamos usar nuestros dispositivos del modo en que lo hacemos habitualmente.

---

## Fichas para estudiantes

A fin de facilitar en un solo lugar todas las fichas para estudiantes, armamos un archivo en que se las compila. El PDF es libre y gratuito, se encuentra en versión a color y en blanco y negro, y puede ser descargado desde el sitio web de Program.AR.

Además de los temas abordados en este libro, también desarrollamos actividades sobre creación de animaciones, programación textual, redes de computadoras y proyectos educativos usando el lenguaje de programación RITA. Este contenido también es libre y gratuito y se encuentra disponible en el sitio web de Program.AR. Se ofrece a continuación un breve resumen de este material complementario.

### ANEXO 1: ANIMACIONES

En este anexo se trabaja con Alice, una herramienta gráfica que permite construir animaciones a partir personajes que se colocan en un escenario y a los que se les dan indicaciones.

Las primeras actividades funcionan como una introducción al entorno y la metodología de trabajo y, gradualmente, se introducen conceptos clave de programación motivados por consignas para que los estudiantes, en grupos, puedan incorporarlos a sus proyectos.

### ANEXO 2: PROGRAMACIÓN TEXTUAL

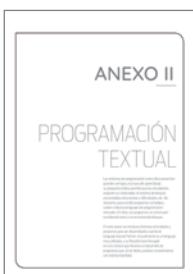
Los entornos de programación como Alice presentan grandes ventajas a la hora del aprendizaje: su propuesta lúdica permite que los estudiantes exploren su creatividad, el sistema de bloques encastrables evita errores y dificultades, etc. No obstante, para escribir programas complejos, suelen utilizarse lenguajes de programación textuales. En ellos, los programas se construyen escribiendo texto y no encastrando bloques.

En este anexo se introduce distintas actividades y proyectos para ser desarrollados usando el lenguaje textual Python. Actualmente es un lenguaje muy utilizado, y su filosofía hace hincapié en una sintaxis que favorece el desarrollo de programas que, al ser leídos, puedan comprenderse con relativa facilidad.

### ANEXO 3: REDES DE DATOS E INTERNET

Una red de computadoras es un conjunto de dispositivos interconectados con el objetivo de compartir información, recursos y servicios. El conjunto de computadoras, el software de red, los medios y los dispositivos de interconexión forman un sistema de comunicación donde existen emisores, receptores y medios de transmisión.

En este anexo se trabaja en torno a los conceptos básicos de redes, a través de actividades que permiten experimentar y comprender el funcionamiento de una red en general y de Internet en



particular. Asimismo, se incluyen algunas actividades que permiten reflexionar sobre los buscadores en Internet.

---

#### **ANEXO 4: PROYECTOS EDUCATIVOS CON RITA**

La búsqueda y adquisición de recursos digitales en Internet, así como la producción de los propios, ofrece un gran potencial cuando de aprender en la escuela se trata. Por lo tanto, los proyectos educativos se presentan como una posibilidad didáctica para que docentes y estudiantes de diferentes años y asignaturas puedan trabajar en torno a la creación, uso y apropiación de diversos recursos con distintos objetivos y formatos.

En este anexo se trabaja en el desarrollo de proyectos educativos orientados a la programación usando RITA (*Robot Inventor to Teach Algorithms*): una herramienta libre, de código fuente abierto y desarrollada en el LINTI-UNLP, especialmente diseñada para programar robots virtuales que compiten en un campo de batalla.

---

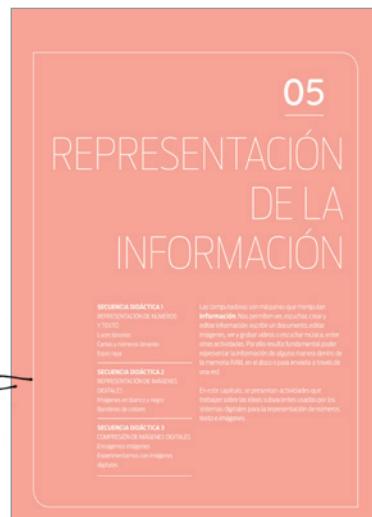


## ¿CÓMO ORGANIZAMOS ESTE MANUAL?

Cada uno de los capítulos del manual tiene las siguientes características:

### APERTURA

Consta de un breve texto introductorio sobre los conceptos abordados en el capítulo y un índice.



## SECUENCIAS DIDÁCTICAS

Cada capítulo está integrado por secuencias didácticas que tienen una coherencia conceptual. Cada secuencia didáctica está conformada por actividades que pueden o no requerir el uso de una computadora.

Datos de orientación sobre el número de secuencia didáctica (SD) y el número de la actividad a la que corresponde la página.

Presentación de la secuencia.	Título de la actividad.	Modalidad y objetivos.	
Materiales para hacer la actividad.	Cada actividad propone formas de abordar la clase, un desarrollo, una sugerencia para resolver la propuesta y un cierre.		

Materiales para hacer la actividad.

Cada actividad propone formas de abordar la clase, un desarrollo, una sugerencia para resolver la propuesta y un cierre.

## FICHAS PARA ESTUDIANTES

Muchas actividades vienen acompañadas de una ficha para entregar a los estudiantes.

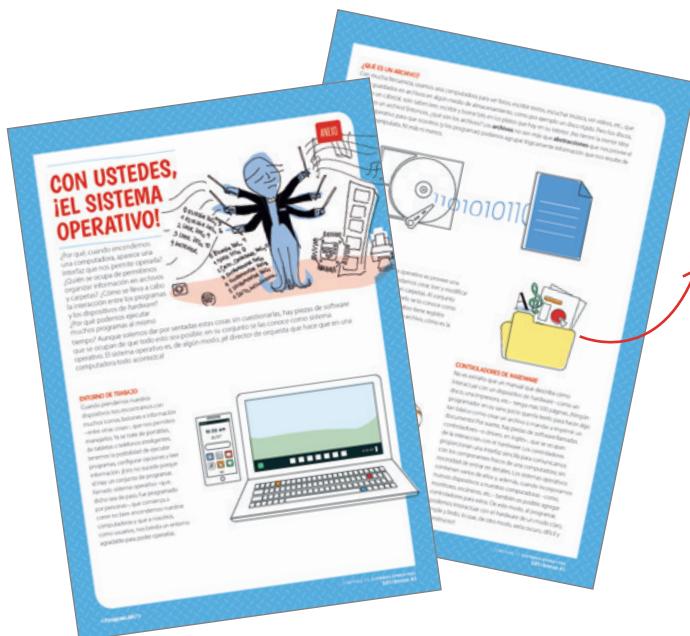
Datos que completa cada estudiante con el fin de facilitar la tarea de identificar a quién corresponde la ficha.



Curiosidades relacionadas con el tema.



Datos sobre el capítulo, secuencia didáctica y actividad a la que corresponde cada ficha.



## ANEXOS

Se trata de material que complementa las actividades.



## GLOSARIO

El manual también incluye un glosario para el docente. Las definiciones no fueron pensadas para ser dadas a los estudiantes y se sugiere no presentarlas de forma descontextualizada.

# 01

# CIUDADANÍA DIGITAL Y SEGURIDAD

## SECUENCIA DIDÁCTICA 1

INFORMACIÓN SENSIBLE  
Y AMENAZAS EN LA RED  
iCuidado: riesgo a la vista!  
Lo que publico, ¿quiénes lo ven?  
No quiero ser un pescado

---

## SECUENCIA DIDÁCTICA 2

CONTRASEÑAS Y CIFRADO  
Contrasenñas ¿seguras?  
Claves compartidas  
Claves públicas y privadas

---

El mundo tecnológico atraviesa cada vez más aspectos de la vida cotidiana. La conectividad establece novedosas formas de vincularnos con el mundo que, hasta hace poco, solo existían dentro de los límites de la ciencia ficción: realizamos videoconferencias con personas en distantes lugares del mundo mientras paseamos a un perro, disponemos de la discografía completa de nuestros músicos favoritos con solo presionar un botón que, en realidad, solo existe en una pantalla, etc.

Esta nueva realidad impone nuevas reglas de juego, nos enfrenta a nuevos desafíos y nos expone a ciertos riesgos sobre los cuales debemos ser conscientes. A lo largo de este capítulo se exploran diversos aspectos de ciudadanía digital, la privacidad de la información y los métodos seguros para el resguardo y envío de datos en Internet.



## Secuencia Didáctica 1

# INFORMACIÓN SENSIBLE Y AMENAZAS EN LA RED

Al usar los sitios, aplicaciones y recursos que nos brinda Internet, nos exponemos a diferentes situaciones de riesgo. Por un lado, es habitual que publiquemos información que, en caso de caer en manos equivocadas, puede ser usada en nuestra contra. Sobre todo, porque es poco frecuente que tengamos conciencia del alcance de lo que volcamos en la red: quiénes pueden verlo y quiénes no, dónde reside, etc.

Por otro lado, también existen mecanismos mediante los cuales individuos malintencionados pueden robarnos contraseñas, detalles de tarjetas bancarias, etc., con las que, en el universo virtual, pueden perpetrar ilícitos, como realizar publicaciones a nuestro nombre o llevar a cabo operaciones crediticias fraudulentas.

Esta secuencia didáctica propone actividades para generar conciencia de algunos de los riesgos a los que nos enfrentamos en el universo digital y brindar herramientas para saber cómo enfrentarlos.

.....  
**OBJETIVOS**

- Generar conciencia sobre riesgos que se presentan en Internet.
  - Brindar herramientas para enfrentarlos.
- .....

## Actividad 1

### ¡Cuidado: riesgo a la vista!



GRUPAL (3)

#### OBJETIVOS

- Identificar información sensible que se publica en Internet.
- Concientizar sobre los riesgos que supone la exposición de información.

#### MATERIALES



Ficha para estudiantes

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes tomen conciencia de los riesgos que implica la publicación de cierta información en Internet. Para ello, se les presentarán distintas publicaciones hipotéticas que analizarán para reconocer situaciones indeseables que puedan presentarse.

Comenzamos la actividad preguntando a los estudiantes: “¿Qué tipo de información suelen compartir en Internet, especialmente en redes sociales?”. Escuchamos con atención sus respuestas y continuamos: “¿Quiénes pueden acceder a esa información?”. Entonces, reflexionamos grupalmente sobre la importancia de resguardar la información sensible.

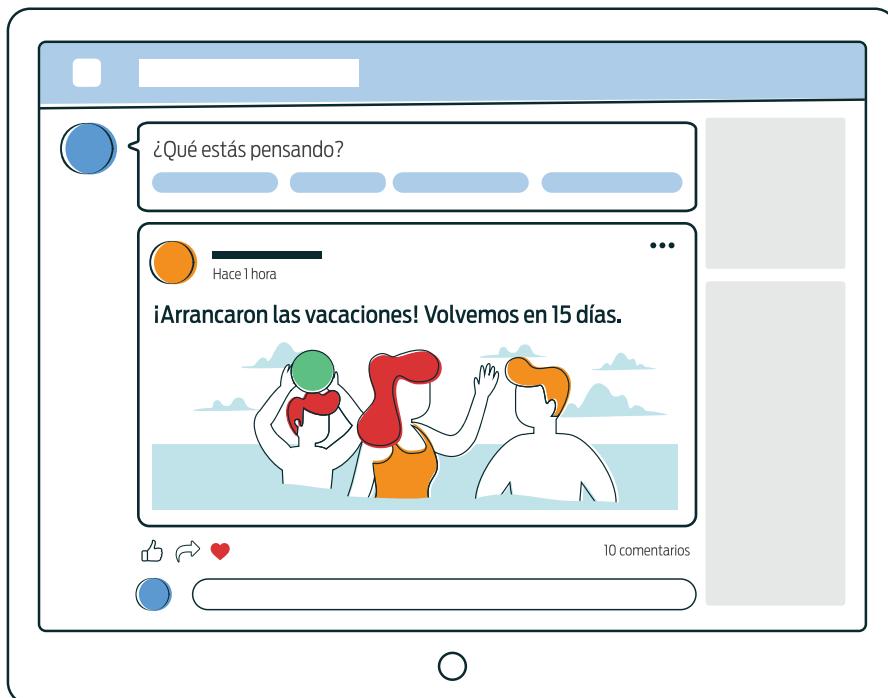
Les repartimos las fichas a los estudiantes y les pedimos que armen grupos de tres integrantes. Luego, les indicamos que observen las imágenes que aparecen en la ficha y completen la primera consigna. En ella se muestran distintos ejemplos de información que fue compartida en Internet por sus protagonistas. Entonces, tendrán que analizar con atención cada imagen, teniendo en cuenta que se puede estar proporcionando información que derive en situaciones riesgosas e indeseables.

Algunos de los datos sensibles que contienen las imágenes, y que esperamos que los estudiantes sean capaces de reconocer, se describen en los párrafos a continuación.



Una joven sola en su casa

La primera imagen muestra una publicación que revela que la protagonista se encuentra sola en su casa. Esta información puede ser muy riesgosa si cae en manos equivocadas, ya que puede derivar en situaciones que pongan en riesgo la integridad física y psicológica producto de, por ejemplo, acoso y hostigamiento.



Una familia de vacaciones

La segunda imagen da cuenta de que una familia está de vacaciones y, probablemente, su casa ha quedado deshabitada. Con esta información, personas malintencionadas podrían intentar entrar en la vivienda, ya sea para robar bienes, ocupar la casa, etc.



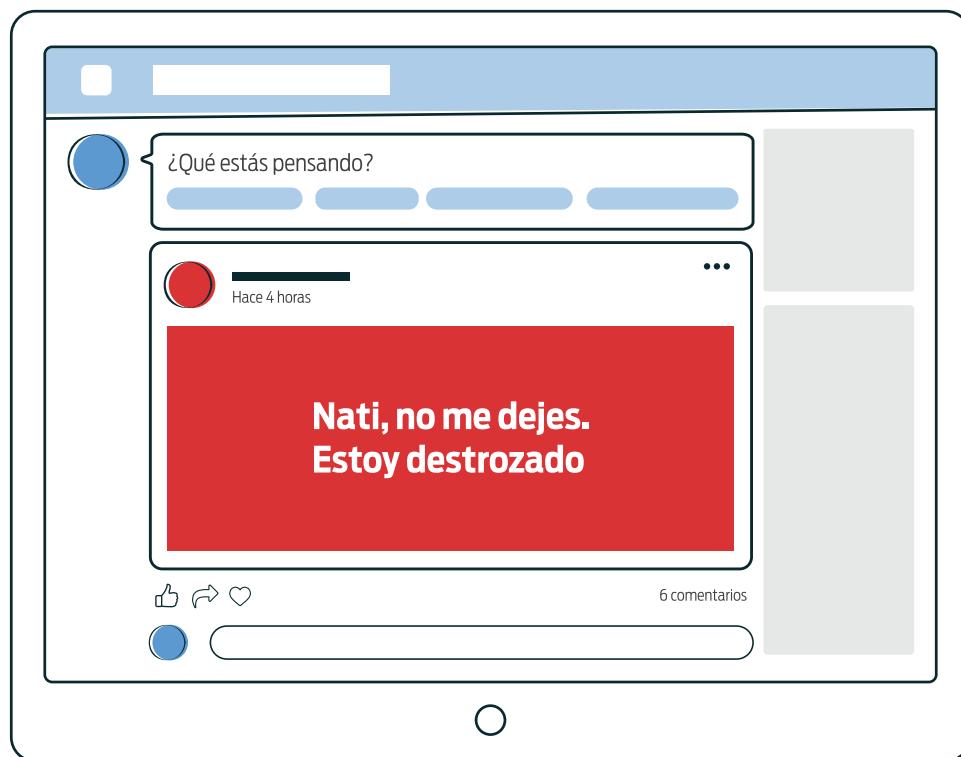
Unos chicos van a un recital

La tercera imagen da cuenta de que un grupo de jóvenes asistirá a un recital. A partir de la publicación, puede deducirse dónde estarán las personas que aparecen involucradas en el mensaje, en el día y horario del evento. Además, también se brinda información sobre sus gustos e intereses, que puede ser usada por alguien perverso para intentar ganar la confianza de estas personas.

En la cuarta imagen, el protagonista subió a Internet una imagen donde brinda involuntariamente su domicilio. Se trata de un dato sensible, pues toda persona malintencionada que acceda a ese dato sabrá dónde puede encontrarlo, cualquiera sea el propósito espurio que tenga.



Queda expuesto el domicilio



Un corazón roto

La última imagen muestra evidencia de que quien la publicó se encuentra en un momento de vulnerabilidad emocional, lo cual podría predisponer a esta persona a confiar en alguien desconocido malintencionado que simule empatizar con sus problemas.

Una vez que todos hayan completado la ficha, hacemos una puesta en común y debatimos con toda la clase acerca de la información relevada y las respuestas que escribieron.

A continuación invitamos a los estudiantes a que piensen qué harían si, luego de publicar algo en Internet, se arrepienten de haberlo hecho. Escuchamos sus respuestas y a continuación concluimos: “Suele ser prácticamente imposible hacer desaparecer de la red lo que publicamos. Además, también deben saber que, aun cuando borremos una imagen, un mensaje, etc., antes de que deje de mostrarse a otros usuarios, alguien podría haberle sacado una foto y, en consecuencia, hacer algo con ella. Conclusión: casi siempre que publicamos algo, perdemos el control sobre lo que sucede con eso”.

### **CIERRE**

Les comentamos a los estudiantes que, en general, al abrir una cuenta en una red social cedemos los derechos sobre todo aquello que publicamos. Además, en la mayoría de los casos, lo que publicamos (ya sea texto, fotos, videos, etc.) queda almacenado en los servidores de las compañías de Internet, con lo que, al borrar contenido, no es cierto que se esté borrando “del todo”.

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# ¡CUIDADO: RIESGO A LA VISTA!



El desembarco de las redes sociales en nuestras vidas ha modificado el modo de relacionarnos con el mundo. Por ejemplo, hasta hace poco, para mostrarle una foto a un amigo, primero teníamos que imprimirla, luego encontrarnos con él y, finalmente, mirarla juntos. Ahora es mucho más sencillo, ¿no? Este nuevo mundo, más virtual y menos personal, también modificó los riesgos a los que nos exponemos. Frente a nuevas reglas de juego, debemos aggiornar las precauciones para no exponernos a situaciones riesgosas.

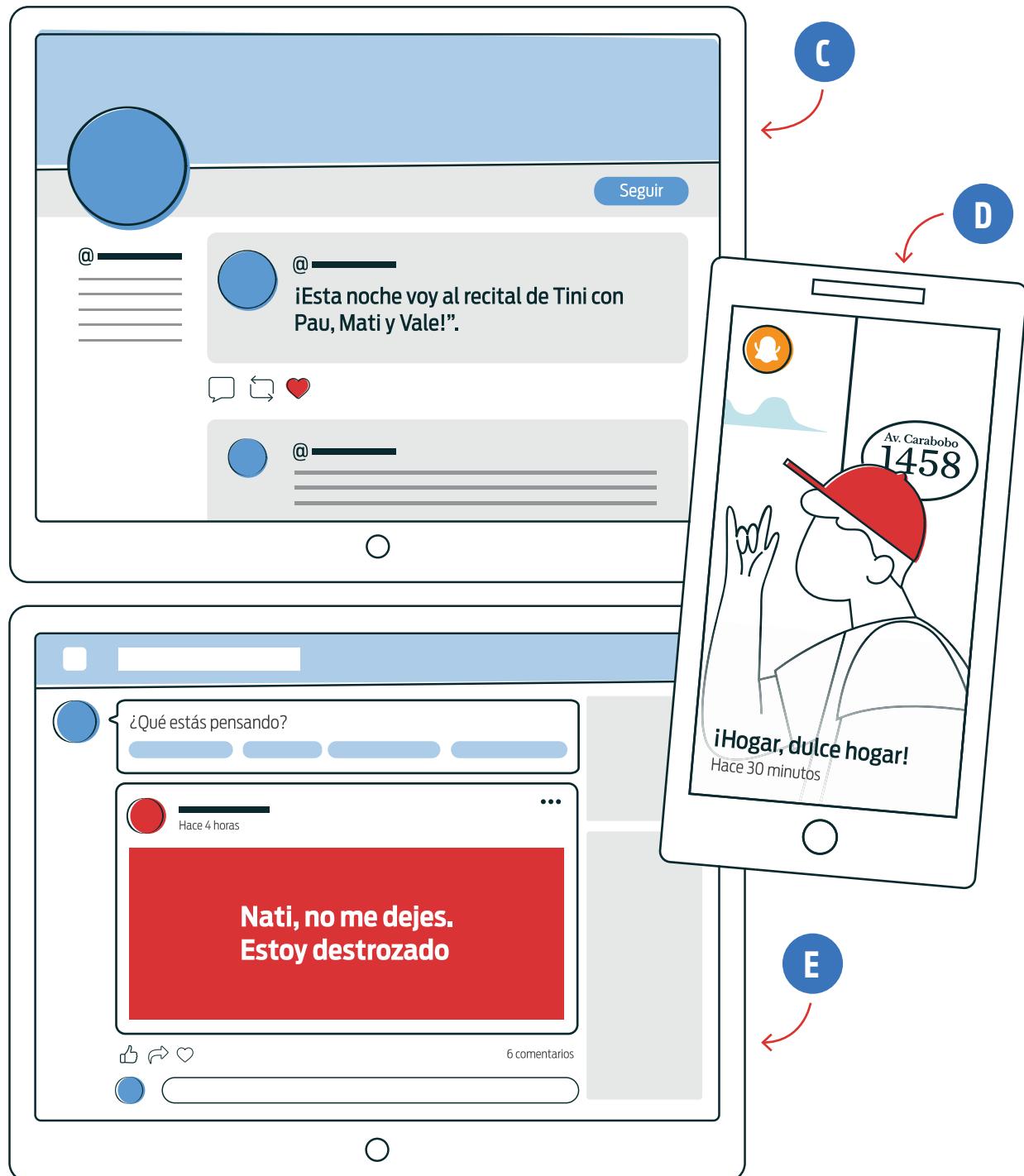
1. Observá las siguientes publicaciones en redes sociales y respondé las preguntas a continuación.



NOMBRE Y APELLIDO:

CURSO:

FECHA:



¿Qué tipo de información brinda cada imagen?

---

---

---

---

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

¿Cuáles son los riesgos que podrían originarse a partir de cada una de las situaciones de las imágenes?

---

---

---

---

En cada caso, ¿la información sensible fue compartida voluntariamente?

---

---

---

---

**2.** ¿Tenemos control sobre la información que publicamos en Internet? ¿Por qué?

---

---

---

---

## Actividad 2

### Lo que publico, ¿quiénes lo ven?



#### OBJETIVOS

- Generar conciencia sobre el alcance de las publicaciones en medios digitales.
- Configurar opciones de seguridad de redes sociales.

#### MATERIALES

- Computadoras
- Internet
- Ficha para estudiantes

#### DESARROLLO

El objetivo de la actividad es poner en evidencia lo sencillo que resulta encontrar información personal en Internet, mucha de la cual puede ser sensible y que, a menos que se tomen las precauciones adecuadas, puede permanecer disponible para desconocidos sin que se tenga conciencia. Para ello, los estudiantes analizarán la información personal de ellos mismos que se encuentra disponible en la red y el alcance de esos datos.

Para poner en práctica la actividad, en la clase previa deberemos relevar todas las redes sociales que usan los estudiantes y, para cada una de ellas, encargarnos de crear una cuenta *ad hoc*. Por ejemplo, si hay estudiantes que usan Instagram, abriremos una cuenta de Instagram; si algunos usan Facebook, crearemos una cuenta de Facebook; etc. De todas ellas, conservaremos el nombre de usuario y la clave.

Comenzamos la clase comentando a los estudiantes: “Ya hace largo rato que usamos redes sociales para diversas cosas, ¿verdad? Hay muchas redes distintas, y cada una tiene su propio propósito y su formato. Ustedes, ¿cuáles usan? ¿Para qué las usan?”. Es probable que algunos mencionen que las utilizan para compartir fotos, ideas y videos, estar en contacto con amigos, mantenerse informados, emitir opiniones, etc. Continuamos: “¿Con quiénes comparten lo que comparten?”.

A continuación, les repartimos la ficha de la actividad y los invitamos a que resuelvan la primera consigna. Allí se presentan preguntas para que los estudiantes cuenten con quiénes comparten sus datos y publicaciones, qué canales se encuentran habilitados para que extraños se comuniquen con ellos, etc. El propósito es que expongan sus creencias sobre el nivel de privacidad y el alcance que tiene la información que publican.

Hacemos una puesta en común y escuchamos sus respuestas y el intercambio que surja entre ellos. Luego, les pedimos que se organicen en parejas y se ubiquen frente a una computadora.

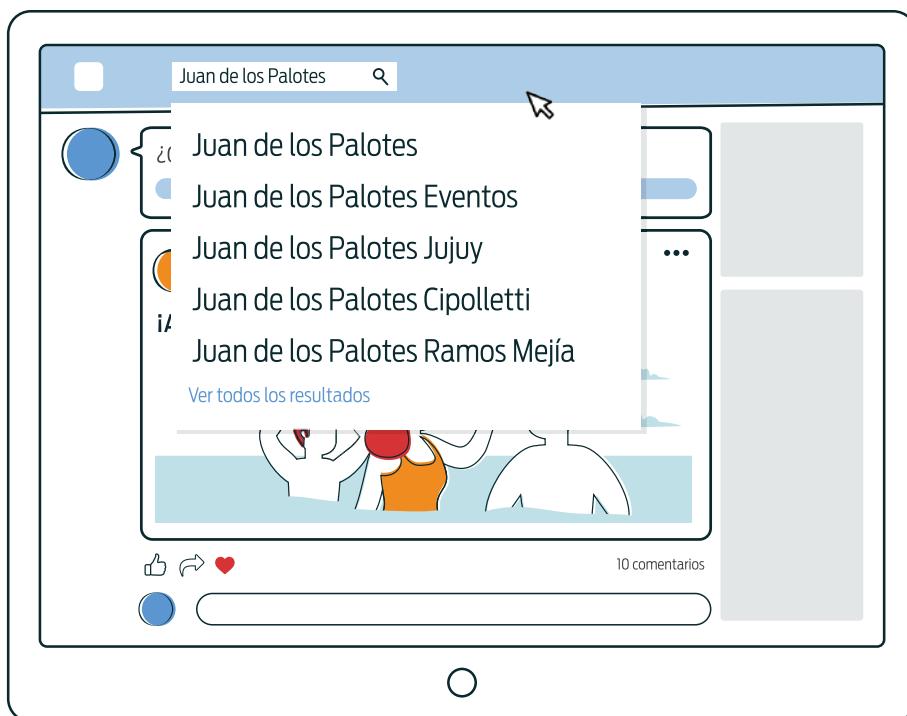
Acto seguido, copiamos en el pizarrón los nombres de usuario y las claves de las cuentas que creamos en cada una de las redes sociales.

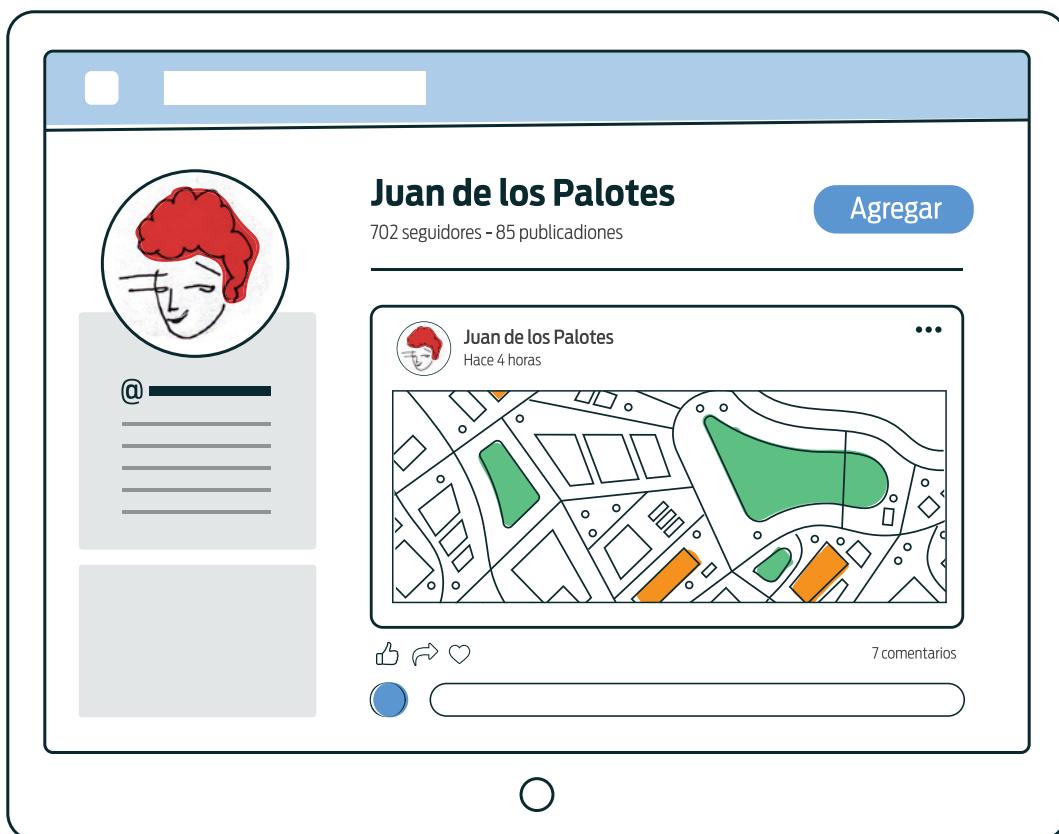
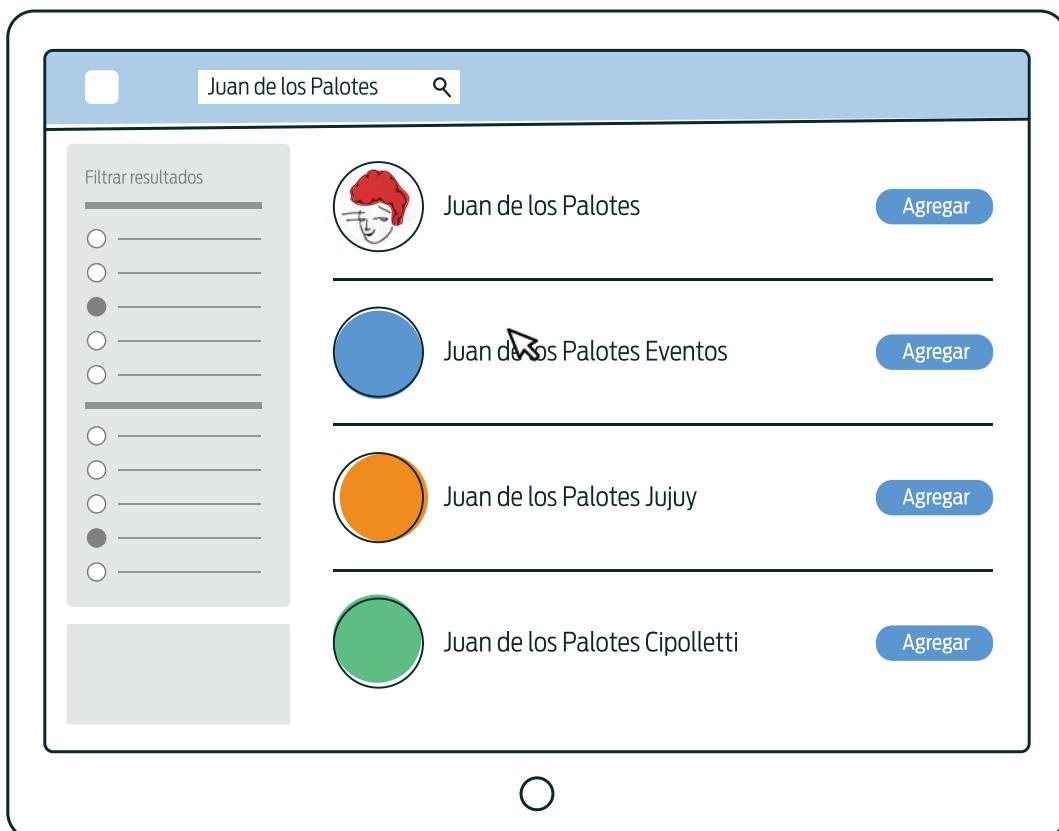
Red social	Usuario	Clave
Instagram	instagram_cuenta_1	Clave12345
Facebook	facebook_cuenta_1	Clave12345
Twitter	@twitter_cuenta_1	Clave12345

### Usuarios y claves

Les indicamos que ingresen a alguna de las redes que ellos usan habitualmente, pero sin usar sus usuarios, sino la cuenta de esa red social que copiamos en el pizarrón. Les pedimos, además, que no realicen cambios una vez que entran (como agregar contactos, modificar opciones de seguridad, etc.).

Entonces, les indicamos que resuelvan la segunda consigna. Allí se les pide que, dentro de la red, realicen una búsqueda de sus nombres y apellidos y observen, en cada caso, con qué se encuentran. La información que aparecerá dependerá de cómo tengan configurados sus opciones de seguridad y privacidad, pero puede incluir fotografías, textos, videos, contactos, dirección de correo electrónico, gustos, lugares recientemente visitados, etc.





Búsqueda por nombre y apellido de un estudiante

Subrayamos: "Tengan presente que la búsqueda la están haciendo con un usuario que no forma parte de sus contactos. Todo lo que ven se encuentra accesible para cualquier persona".

Para completar la consigna, tienen que llenar una tabla con el tipo de información personal que descubrieron que se encuentra publicada, si sabían que estaba disponible para cualquier usuario y, por último, si quieren o no compartirla con desconocidos.

RED SOCIAL	¿QUÉ ENCONTRASTE?	¿SABÍAS QUE ESA INFORMACIÓN ERA ACCESIBLE PARA CUALQUIERA?	¿QUERÉS QUE ESA INFORMACIÓN SEA ACCESIBLE PARA DESCONOCIDOS?
Facebook	Fotos, videos, lugares visitados	No	No
Instagram	Fotos, dirección de correo	Sí	Sí
Snapchat	Fotos	No	No

Possible respuesta de la consigna de la actividad

Una vez que todos hayan revisado la información pública de sus perfiles en las redes sociales, hacemos una puesta en común y promovemos un debate acerca de lo inconveniente que resulta que cualquiera pueda acceder a ciertos datos (como por ejemplo, los datos personales).

Comentamos: "En general, las redes sociales nos ofrecen la posibilidad de decidir con quién queremos compartir la información que publicamos en ellas, mediante lo que se conoce habitualmente como **configuración de privacidad**. Sin embargo, de forma predeterminada estas configuraciones no tienen los niveles más restrictivos, lo cual puede causar que parte de nuestra información sea accesible para más personas de las que nos gustaría. Por lo tanto, es recomendable dedicar un tiempo prudencial a ajustarlas y revisar de forma periódica cada una de las aplicaciones que usamos para verificar que solo sea visible aquello que queramos que sea visible". Los incitamos, entonces, a que ajusten las opciones de sus cuentas de usuario para que no quede expuesto nada que no quieran que sea público.

Configuración y herramientas de privacidad

Tu actividad

¿Quién puede ver las publicaciones que hagas a partir de ahora?

Usar registro de actividad

Revisa todas tus publicaciones y los contenidos en los que se te etiquetó

Limitar el público de publicaciones anteriores

¿Quieres limitar los destinatarios de las publicaciones que compartiste con los amigos de tus amigos o que hiciste públicas?

Cómo pueden encontrarte y ponerse en contacto contigo

¿Quién puede enviarte solicitudes de amistad?

¿Quién puede ver tu lista de amigos?

¿Quién puede buscarte con la dirección de correo electrónico que proporcionaste?

Opciones de seguridad y privacidad

## CIERRE

Remarcamos la importancia de la **privacidad en línea**. Enfatizamos especialmente la importancia de ser conscientes de qué información estamos compartiendo y con quién. En particular, a cuál pueden acceder desconocidos, de forma tal que siempre que compartamos contenido lo hagamos de acuerdo con nuestra voluntad. Reflexionamos, también, acerca de los **peligros** de no manejar correctamente la privacidad y confiar en las opciones que vienen predeterminadas en las redes sociales.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# LO QUE PUBLICO, ¿QUIÉNES LO VEN?

A veces es difícil imaginar hasta dónde puede llegar el contenido que publicamos en Internet. Por suerte, las redes sociales poseen configuraciones de privacidad que nos permiten decidir quiénes pueden ver nuestra información y quiénes no.

- 1.** Contestá las siguientes preguntas, pensando en cada una de las redes sociales y aplicaciones que usás a diario. ¿Conocés la respuesta para todas ellas?

¿Quiénes pueden acceder a la información disponible en tu cuenta?

---

---

---

¿A quiénes aceptás como contactos, amigos, seguidores, etc.?

---

---

---

¿Resultás fácil de encontrar y reconocer? ¿Tu nombre y/o foto de perfil es visible para todos?

---

---

---

¿Quiénes pueden escribirte por chat?

---

---



NOMBRE Y APELLIDO:

CURSO:

FECHA:

¿Podés bloquear cuentas de desconocidos que intenten contactarte o comparten contenido que te resulta agresivo o desagradable?

---

---

---

---

¿Cuánta información pueden ver otras personas sobre vos? ¿Todos pueden ver lo mismo?

---

---

---

---

¿Alguna de las redes sociales que usás se vincula con otras cuentas que uses? ¿Publica automáticamente lo que subís en algún otro sitio? ¿Te pregunta cada vez?

---

---

---

---

¿Podés ser etiquetado en publicaciones ajenas? ¿Quiénes pueden compartir tus publicaciones?

---

---

---

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

2. Ingresá a las redes sociales con una cuenta que no forme parte de tus contactos y, una vez adentro, buscá tu nombre y apellido. ¿Con qué te encontraste? ¿Sabías que lo que ves está disponible para cualquier persona? Completá la siguiente tabla con tus hallazgos sobre la privacidad de tus datos en las tres redes sociales que utilices con más frecuencia.

RED SOCIAL	¿QUÉ ENCONTRASTE?	¿SABÍAS QUE ESA INFORMACIÓN ERA ACCESIBLE PARA CUALQUIERA?	¿QUERÉS QUE ESA INFORMACIÓN SEA ACCESIBLE PARA DESCONOCIDOS?

### DECIDI VOS SOBRE LO PRIVADO Y LO PÚBLICO

Es importante que seas consciente de qué información compartís y con quién. Especialmente, a cuál pueden acceder desconocidos. Si bien las redes sociales permiten configurar el nivel de privacidad de lo que publicás, los niveles por defecto no suelen ser los más restrictivos. Revisá la configuración de tus cuentas y asegurate de que el alcance de tu información no vaya más allá de lo que vos querés.



Configuración

Privacidad

Configuración y herramientas de privacidad

Tu actividad

¿Quién puede ver las publicaciones que hagas a partir de ahora?

Usar registro de actividad

Revisa todas tus publicaciones y los contenidos en los que se te etiquetó

Limitar el público de publicaciones anteriores

¿Quieres limitar los destinatarios de las publicaciones que compartiste con los amigos de tus amigos o que hiciste públicas?

Cómo pueden encontrarte y ponerse en contacto contigo

¿Quién puede enviarte solicitudes de amistad?

¿Quién puede ver tu lista de amigos?

¿Quién puede buscarte con la dirección de correo electrónico que proporcionaste?

## Actividad 3

### No quiero ser un pescado



INDIVIDUAL

#### OBJETIVOS

- Presentar el abuso de *phishing*.
- Reflexionar sobre el robo de identidad.

#### MATERIALES

- Proyector
- Computadora
- Internet
- Ficha para estudiantes

#### DESARROLLO

El objetivo de esta actividad es presentar un abuso informático conocido como *phishing*. Esta palabra es una modificación del inglés *fishing*, que significa “pesca”, pero reemplazando la letra *f* por la escritura *ph*, recurso que es común en el ambiente *hacker*. En un ataque de *phishing*, el perpetrador busca obtener información confidencial acerca de la víctima (por ejemplo, la contraseña de algún servicio o los detalles de su tarjeta de crédito) haciéndose pasar por otra persona, empresa, sitio o entidad en quien la víctima confía, a la manera de un cebo que induce al damnificado a morder el anzuelo y proporcionar la información. Posteriormente, la información robada puede usarse para fines espurios, tales como enviar mensajes a nombre de la otra persona o realizar compras por Internet.

Comenzamos preguntando a los estudiantes: “¿Alguna vez les han robado algo?”. Puede ocurrir que todos contesten que no o, alternativamente, que algunos mencionen algún episodio. En este último caso, probablemente hagan referencia a algún robo o hurto de un objeto, como por ejemplo un teléfono celular, dinero o una prenda de vestir. Indagamos: “¿Qué es lo peor que se les ocurre que les podrían robar?”. Es esperable, otra vez, que las respuestas estén orientadas a la pérdida de bienes materiales. Continuamos: “En realidad, lo más valioso que alguien puede robarnos es nuestra identidad. Pero, ¿qué significa eso? ¿Cómo podría alguien robarnos la identidad?”.

A fin de introducir las nociones asociadas a esta amenaza, utilizaremos una escena de la película *Nueve reinas*, protagonizada por Ricardo Darín y Gastón Pauls. De ser posible, la proyectamos o pedimos a la clase que se agrupe para verla en celulares o computadoras.<sup>1</sup> De no poder contar con el video de ningún modo, podemos relatársela a los estudiantes.

Los protagonistas son dos estafadores que, en pleno centro de la ciudad de Buenos Aires, se dirigen a un edificio con una gran cantidad de timbres. Uno de ellos empieza a tocar uno por uno. Cada vez que toca un timbre, una persona responde “¿Quién es?” y el personaje interpretado por Darín le repregunta “¿Tía?”, esperando una respuesta afirmativa. Si la persona del otro lado del portero responde con un “No” o “Equivocado”, o si escucha una voz joven, el estafador abandona la conversación y busca otra potencial víctima. En una oportunidad, una voz de una anciana responde “¿Quién es?”; el malhechor, al notar que esta vez puede funcionar, le responde “¿Cómo, ya no reconocés a tu sobrino favorito?” y la anciana inmediatamente responde “¡Fabián! ¿Sos vos? Querido, tanto tiempo...”. El supuesto sobrino empieza a contarle una situación falsa para obtener un rédito

<sup>1</sup> La escena se encuentra disponible en línea en <http://bit.ly/2ClknkV>, desde el minuto 0:38 hasta el minuto 3:18.

económico: le dice que tuvo un problema con el auto y que se olvidó la billetera, y le pide algo de dinero. La anciana le dice que no hay problema y que ya baja. Darín le dice que no puede dejar el auto solo, pero manda a un “amigo” para que reciba la plata. La anciana baja, se encuentra con el otro estafador, y le da dinero y un anillo.

Luego de observar la escena, los invitamos a un debate grupal: “¿Qué les parece lo que vieron? ¿En qué consistió el engaño?”. Prestamos atención a la discusión y, luego, comentamos: “Con estas conductas se busca sacar provecho de ciertos hábitos, características y decisiones de las víctimas para poder realizar algún tipo de engaño o manipulación psicológica. A las averiguaciones o presunciones respecto de la información y las mejores formas de obtenerla se las denomina **ingeniería social**. Por ejemplo, en el caso anterior, se trató de averiguar qué tipo de información sobre la vida de la anciana podría obtenerse con engaños y utilizarse para que le dé dinero a un desconocido. Cuando estos engaños se realizan a través de medios electrónicos se denominan *phishing*; la obtención de los datos puede hacerse a través de formularios, correos electrónicos, mensajes de texto, enlaces a sitios idénticos o muy parecidos a un registro original, etc.”.

Repartimos la ficha a los estudiantes y les solicitamos que resuelvan las dos primeras consignas. En la primera, se muestra una página de Internet que aparece ser la de inicio de sesión de la red social Facebook. Aun cuando la portada es igual, al mirar con atención puede observarse que la dirección, en lugar de ser `facebook.com`, es `face6ook.com` (se reemplaza la `b` por el número seis). En la ficha, bajo la imagen, se pregunta para qué sirve la página expuesta, a lo que probablemente una gran mayoría de los estudiantes conteste que se usa para iniciar una sesión en la mencionada red social. En este caso, estamos en presencia de una página que simula ser el punto de acceso a una red social para realizar robo de identidades. Un usuario desprevenido podría ingresar su nombre de usuario y contraseña para autenticarse y que, la (falsa) página, reenvíe los datos a un tercero malintencionado que busca hacerse de identidades digitales ajenas. Lo mismo podría ocurrir con páginas para iniciar sesiones en un servidor de correo electrónico, en uno de *home banking*, etc.



Falsa pantalla de inicio de sesión

La segunda consigna invita a leer un artículo periodístico que relata una estafa de la que fue víctima la municipalidad de la localidad de 25 de mayo, en la provincia de Buenos Aires. La estafa se efectuó presentando una página web apócrifa, casi idéntica a la de un banco, que era difundida con publicidades en buscadores para que fuera confundida con la original. Les pedimos que lean el artículo y que lo analicen utilizando como guía las preguntas planteadas en la ficha, que indagan sobre quiénes fueron las víctimas del ataque, qué información se robó, para qué se usó, etc.

Hacemos una puesta en común y los invitamos a que observen nuevamente la página de inicio de sesión de Facebook de la primera consigna. Es esperable que, ahora sí, reconozcan que la dirección es falsa y que, por lo tanto, se trata de una trampa. Concluimos: “Para iniciar una sesión de algún servicio, siempre es conveniente prestar atención a la dirección de la página, para estar seguros de no estar siendo víctima de *phishing*. Además, es preferible ingresar la dirección en la barra de direcciones del navegador, en vez de hacerlo mediante enlaces procedentes de fuentes desconocidas”.

### **CIERRE**

Repasamos esta modalidad de estafa y recordamos que la información confidencial no suele ser solicitada por mail, servicios de mensajería ni llamados telefónicos; en general, los organismos que la requieren no la piden por estos medios. Por lo tanto, no se debe responder a estas solicitudes sin chequear la autenticidad de los pedidos.

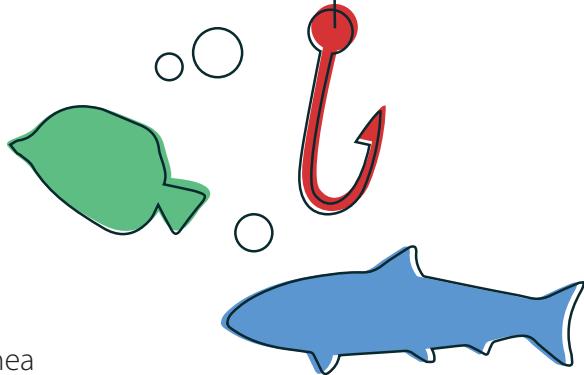
---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# NO QUIERO SER UN PESCADO



Un pescador pone la carnada en el anzuelo y lanza la línea al agua. Luego, pacientemente esperará hasta que... ¡zaz! ¡El pobre animal mordió el anzuelo! Aunque suene difícil de creer, la pesca virtual existe, y también los pobres pescados.

1. Mirá la siguiente página.



¿Qué es esta página? ¿Para qué se usa?

---

---

---

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

- 2.** Leé atentamente el siguiente artículo periodístico y respondé las preguntas que se presentan a continuación.<sup>1</sup>

7 de agosto de 2017

## 25 de mayo: el pueblo al que le robaron \$ 3,5 millones con un aviso en Google

Parecía un día más. Un día alegre, en realidad, para este municipio de 40 mil habitantes. Porque en el partido de 25 de mayo, a 220 kilómetros de la Capital, después de mucha insistencia recibían una ambulancia para Norberto de la Riestra, una de las localidades de este partido bonaerense. [...]

### La municipalidad

Aquel lunes 21 de noviembre de 2016, Roberto Testa, el tesorero del municipio, entró a trabajar a las 8 de la mañana en 25 de mayo, la ciudad que es cabecera del partido homónimo y que crece junto a la laguna Las Mulitas. A las 10, como todos los días, fue al Banco Provincia. Pidió que le imprimieran un resumen con todos los movimientos de las cuentas bancarias. [...] Ese día detectó, con su ojo prolíjo de 32 años de carrera, que algo no andaba bien. "Empiezo y veo en el resumen que teníamos una transferencia por 100 mil pesos, otra por 90 mil. Me fui corriendo a lo del contador a comprobar por qué habíamos hecho esos movimientos", explica. [...]

### El *home banking*

Paolo Salinas, el contador hace 18 años y encargado de emitir los pagos, giró su mirada hacia la puerta de la oficina para recibirla. Escuchó a Testa y de inmediato se metió en las cuentas de *home banking* con su computadora. Salinas es el único que sabe las claves. Y no las anota en ningún lado, "por seguridad", dice. "Están en mi cabeza", acota. [...]

No recuerda específicamente el proceso de ese día, pero por lo general busca "Banco Provincia BIP" (Banca Internet Provincia) en Google, hace clic en el primer resultado, ingresa las claves y empieza a revisar las cuentas. "En ese momento me decía que había un usuario más adentro al mismo tiempo que yo", recuerda. Le preguntó a Romina Mancha, la subcontadora, si era ella. "No", le contestó Mancha desde el otro lado de la oficina, mientras dejaba su silla y se acercaba a la PC de Salinas. [...]

Mancha dejó el monitoreo y fue corriendo hasta la oficina de Ticera, la secretaria de Hacienda, quien a esa hora estaba reunida. No golpeó la puerta cerrada para entrar. "Me interrumpe y me dice: 'nos están robando, nos están robando' [...]", recuerda. [...] Fueron a buscar al Intendente: "Salimos disparadas". [...]

### Tres millones y medio

El intendente, de 32 años [...], no salía de su asombro. Le pidió a Salinas que le mostrara lo que estaba sucediendo. Otra vez. Se dieron cuenta de que habían aparecido más transferencias, todas hechas a proveedores no habituales del municipio. "Cada segundo que pasaba perdíamos más plata", rememora. A esa altura, casi el mediodía de ese lunes fatídico, ya había 3 millones y medio menos de pesos en las arcas municipales [...]. Entonces, recibieron al menos una buena noticia: lograron que les bloquearan las cuentas. Eran poco más de las 12. [...]

<sup>1</sup> El artículo completo puede encontrarse en <http://bit.ly/2krwGHO>.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

## Pesca con mediomundo

Juan Ignacio Bidone es fiscal de investigaciones complejas del Departamento Judicial de Mercedes. A esta altura, tiene muy claro lo que pasó. La secuencia del robo de 3 millones y medio pesos al municipio de 25 de mayo, dinero que luego fue extraído de diferentes cuentas, se realizó mediante *phishing*: una forma de engaño informático con la que se logra que un usuario revele información personal. Los ciberdelincuentes crearon un sitio falso similar al de la Banca Internet Provincia, también conocido como BIP por sus iniciales. Era idéntico al verdadero, pero con un detalle: la dirección, que obviamente no puede ser la original. Suplantaron la *a* por la *s*, para hacer más imperceptible el cambio, y montaron un sitio en la dirección [bancsprovincia.banksinternet.com.ar](http://bancsprovincia.banksinternet.com.ar).



El resultado falso que aparecía en Google

Para lograr que alguien visitara ese sitio pensando que estaba entrando al Banco Provincia aplicaron una técnica llamada *black hat SEO*: una estratagema que desafía las reglas para lograr escalar posiciones en los listados de Google. En este caso, contrataron el servicio publicitario de AdWords; eso fue determinante, ya que lograron hacer que ante una búsqueda en Google de la frase "Banco Provincia BIP" el sitio falso que crearon apareciera como primer resultado. [...]

Cuando alguien entraba, en el sitio falso se le pedía el nombre de usuario y la clave para ingresar a las cuentas del banco. Capturaba la información y luego redirigía al verdadero sitio BIP para no despertar sospechas. [...] Así —estima la fiscalía— los delincuentes se hicieron de las contraseñas, la única credencial necesaria para realizar la operación. [...]

¿Quién o quiénes fueron las víctimas del ataque descripto en el artículo?

---

---

¿Qué información fue robada por el atacante?

---

---

---

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

¿Para qué fue usada la información robada? ¿Qué consecuencias tuvo el ataque para las víctimas?

---

---

¿Cómo se hizo para robar la información de las víctimas? ¿Qué nombre recibe este tipo de ataque?

---

---

¿Qué señales podrían haber alertado al contador de que estaba siendo engañado cuando ingresó al sitio web apócrifo?

---

---

Escribí otros ejemplos de información que podría ser robada mediante engaños similares.

---

---

¿Qué consejos pueden extraer de este artículo para evitar ser víctimas de un ataque como este?

---

---

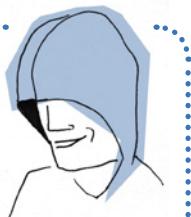
Volvé a mirar la página web de la consigna 1. ¿Te parece que es auténtica?

---

---

### ¿QUÉ ES PHISHING?

La palabra *phishing* es una modificación del inglés *fishing*, que significa "pesca", pero reemplazando la letra *f* por la escritura *ph*, recurso que es común en el ambiente *hacker*. Se trata de un abuso informático en el que el perpetrador busca obtener información confidencial acerca de la víctima (por ejemplo, la contraseña de algún servicio o los detalles de su tarjeta de crédito) haciéndose pasar por otra persona, empresa, sitio o entidad en quien la víctima confía, a la manera de un cebo que induce al damnificado a morder el anzuelo y proporcionar la información. Posteriormente, la información robada puede usarse para fines espurios, tales como enviar mensajes a nombre de la otra persona o realizar compras por Internet.





## Secuencia Didáctica 2

# CONTRASEÑAS Y CIFRADO

Cuando se desea cuidar la confidencialidad de la información que se utiliza en Internet es necesario considerar diferentes aspectos. En muchas aplicaciones se suele emplear una identidad virtual, asociada a un nombre de usuario y validada con una **contraseña**. La seguridad provista por una contraseña depende de distintos factores, como su longitud, los símbolos usados, etc.

Otro aspecto a considerar es que los datos que se envían por Internet pueden ser interceptados por terceros. Para evitar que sean comprendidos por extraños se utilizan técnicas de **cifrado**, que brindan ciertas garantías de que la información solo podrá ser interpretable para el destinatario del mensaje.

En esta secuencia se presentarán consideraciones útiles para crear contraseñas seguras y dos métodos de cifrado para el envío de información.

### ..... **OBJETIVOS**

- Establecer criterios para crear contraseñas seguras.
  - Comprender la diferencia entre texto plano y texto cifrado.
  - Distinguir los mecanismos de cifrado simétricos y asimétricos.
- .....

## Actividad 1

### Contraseñas ¿seguras?

 INDIVIDUAL

#### OBJETIVOS

- Tomar conciencia de la importancia de usar contraseñas seguras.
- Distinguir entre contraseñas seguras y vulnerables.
- Identificar características que hacen a la solidez de una contraseña.

#### MATERIALES

 Ficha para estudiantes

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes (i) distingan entre contraseñas seguras y otras fácilmente vulnerables; y (ii) reconozcan características que hacen a la seguridad de una contraseña.

Comenzamos la actividad repartiendo la ficha a los estudiantes y les pedimos que individualmente resuelvan la primera consigna. Allí encontrarán una tabla en la que, en la primera columna, hay una serie de contraseñas inventadas. Ellos tienen que completar la segunda, describiendo posibles motivos que podrían haber llevado a un sujeto a elegirlas.

Contraseña	Possible motivo para haberla elegido
Luis2006	
25062006	
Coco	
D4l3B0c4	
Kpo de Temperley	
Ceci te amo	

Contraseñas poco seguras

Luego, hacemos una puesta en común y copiamos en el pizarrón los motivos pensados por los estudiantes. En caso de que no hubiesen surgido en el intercambio con la clase, agregamos los que se muestran a continuación.

Contraseña	Possible motivo para haberla elegido
Luis2006	El usuario se llama Luis y nació en el 2006
25062006	Cumple años el 25 de junio
Coco	El nombre de una mascota
D4l3B0c4	Es hincha de Boca
Kpo de Temperley	Vive en Temperley
Ceci te amo	Su novia se llama Cecilia

Contraseñas y motivos para elegirlas

A continuación preguntamos: “¿Les parece que estas contraseñas son seguras? ¿O creen que son vulnerables?”. Orientamos la discusión para concluir que cualquiera que conozca a Luis podría llegar a descubrirlas. Les comentamos: “La **ingeniería social** permite que personas malintencionadas obtengan acceso a nuestra información para utilizarla en su beneficio. En este caso, alguien que conozca algunos datos sobre Luis podría intuir que su contraseña es alguna de las de la tabla”.

Indagamos: “¿Qué les parece usar contraseñas tales como ‘clave’, ‘123456’ o ‘qwerty’?”. Escuchamos sus respuestas y les comentamos: “Los programas que buscan descubrir contraseñas cuentan con **diccionarios** que incluyen las contraseñas más comunes, como secuencias ascendentes de números o letras, o algunas que ciertos sistemas asignan por defecto (y muchas personas no se ocupan de modificar). Por supuesto, estas alternativas también son fácilmente vulnerables”.

Continuamos: “La ingeniería social y el uso de diccionarios no son los únicos mecanismos para buscar contraseñas. Una técnica para hacerlo es la conocida como **fuerza bruta**, que consiste en emplear programas que realizan una prueba tras otras, analizando todas las posibles contraseñas, hasta llegar a descubrirla. Es por esta razón que muchos sitios bloquean el acceso cuando un usuario se equivoca en algunos intentos consecutivos al ingresar su contraseña”.

Luego, les decimos: “Aunque suene absurdo, supongamos que un sitio de Internet, para crear una cuenta, requiere definir una contraseña compuesta solo por un dígito. ¿Cuántas posibles contraseñas distintas podría elegirse?”. Es esperable que, como las únicas opciones son 0, 1, 2, ..., 9, algún estudiante conteste que son diez. “¡Así es, son diez! En ese caso, si algún intruso quisiera ingresar a la cuenta le resultaría muy sencillo: podría probar con cada una de las diez posibilidades hasta dar con la correcta, ¿no es cierto? ¿Qué pasaría si la contraseña tuviese que ser de dos dígitos?”. En tal caso, las opciones serían 00, 01, 02, ..., 99; es decir, 100 posibles contraseñas. “Si bien sería más molesto, el intruso también podría probar todas las posibilidades. ¿Y si pudiese tener tanto un dígito como dos dígitos?”. En ese caso hay que sumar todas las posibles opciones de un dígito (10) más todas las de dos dígitos (100), lo que da un total de 110 contraseñas posibles.

Proseguimos: “Si usáramos un programa, probar estas cantidades de posibilidades puede hacerse casi instantáneamente. Incluso, una computadora puede probar millones de combinaciones en cuestión de segundos. ¿Se les ocurre cómo podría incrementarse la cantidad de posibles contraseñas?”.

Invitamos a los estudiantes a que resuelvan la segunda consigna, que propone pensar cómo aumentar la cantidad de contraseñas posibles. Una posibilidad es ampliar el número de símbolos que pueden formar parte de la contraseña. Por ejemplo, si pueden usarse tanto los dígitos (10) como las letras del alfabeto castellano (27), hay 37 posibles contraseñas de longitud 1 (en lugar de 10); si consideramos las de longitud 2, tendríamos 37 posibilidades para el primer símbolo y, para cada uno de ellos, 37 para el segundo; es decir  $37 \times 37 = 1369$  combinaciones (en lugar de 100). En general, si contamos con  $n$  símbolos, la cantidad de combinaciones de longitud  $l$  es  $n^l$ . Por, ejemplo, usando dígitos (10), letras en

mayúscula (27) y letras en minúscula (27), la cantidad de posibles contraseñas de longitud 6 asciende a  $(10 + 27 + 27)^6 = 64^6 = 68.719.476.736$ ; más de sesenta y ocho mil millones! Incluso, serían muchísimas más si además se admitiesen otros símbolos (como @, #, etc.) y se permitieran contraseñas de mayores longitudes –como suele suceder actualmente–.

Una vez que concluyen la consigna, hacemos una puesta en común y enfatizamos que:

- **Es recomendable buscar una contraseña que no se desprenda fácilmente de la información personal, los gustos, los intereses, etc.** De esta manera se evita que se obtenga a través de ingeniería social.
- **Se debe evitar el uso de palabras comunes de algún idioma o secuencias de caracteres muy sencillas.** Contraseñas tales como “12345678”, “abcdef” y “qwerty” no son recomendables, pues constituyen cadenas fácilmente reproducibles, además de estar incluidas en diccionarios de claves habituales.
- **La longitud y la variedad de símbolos utilizados contribuyen a la seguridad de la contraseña.** A lo largo del tiempo, distintos expertos han ido recomendando diferentes técnicas para crear contraseñas, pero si estas no garantizan una longitud apropiada suelen ser fácilmente descubiertas usando fuerza bruta.
- **Las contraseñas deben poder recordarse.** Que una contraseña sea larga no necesariamente significa que tenga que ser difícil de recordar. Si nuestras contraseñas son demasiado complicadas –o si algún sitio nos pide que las modifiquemos con demasiada frecuencia–, se termina recurriendo a anotarlas en papel, lo cual puede ser tan peligroso como usar una contraseña corta o fácilmente deducible.

## CIERRE

Mencionamos a los estudiantes que no es buena idea usar la misma contraseña para varios sitios distintos. Si uno de los sitios que utilizamos tiene algún problema de seguridad y la contraseña se filtra, se podrá utilizar en cualquier otro. Así, puede pasar que un usuario malintencionado obtenga acceso a redes sociales, cuentas bancarias o de información delicada con contraseñas robadas, que consiguió mediante problemas de seguridad de otros sitios con información menos sensible, como un foro o un blog.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# CONTRASEÑAS ¿SEGURAS?

¿Hay contraseñas más seguras que otras? ¿De qué depende su nivel de vulnerabilidad? Acá veremos algunas características de las contraseñas que te van ayudar a saber cuán (in)seguras son las tuyas.

1. A continuación hay una serie de contraseñas que una persona eligió para usar en distintos sitios de Internet. ¿Se te ocurre qué motivos lo llevaron a elegirlas? Completá la tabla.



CONTRASEÑA	POSSIBLE MOTIVO PARA HABERLA ELEGIDO
Luis2006	
25062006	
Coco	
D4l3B0c4	
Kpo de Temperley	
Ceci te amo	

¿Te parecen seguras estas contraseñas? ¿Por qué?

---

---

## CONTRASEÑAS VULNERABLES

Una de las contraseñas más usadas es **qwerty**. Esta no es una contraseña segura. ¿Se te ocurre por qué tanta gente la usa? Ayuda: intentá escribirla en el teclado de tu computadora. Otras muy inseguras son: **contraseña, 1234, 1111, 123456, 12345678**. Si una es muy usada, es probable que sea insegura. En [bit.ly/2jZvdY9](https://bit.ly/2jZvdY9) podés encontrar las 25 contraseñas más utilizadas durante 2018.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

2. Es momento de pensar qué factores contribuyen a la seguridad o vulnerabilidad de las contraseñas. Contestá las siguientes preguntas:

¿Cuántas contraseñas distintas de un dígito pueden existir? ¿Y de dos dígitos?

---

¿Cuántas contraseñas distintas existen si consideramos que pueden tener tanto un dígito como dos?

---

¿Y cuántas hay si consideramos que son de una letra del alfabeto castellano? ¿Y si son de dos letras?

---

¿Qué cantidad de posibles contraseñas hay si permitimos tanto dígitos como letras para contraseñas de longitud 1? ¿Y para contraseñas de longitud 2?

---

En general, si contamos con  $n$  símbolos, ¿cuántas combinaciones distintas hay fijando una longitud  $l$ ?

---

Calculá la cantidad de posibilidades que existe si se pueden usar dígitos, letras en mayúscula y letras en minúscula para contraseñas de 6 símbolos. ¿Son muchas? ¿Cuántas?

---

#### FUERZA BRUTA

Hay programas que, para descubrir contraseñas, se valen de la **fuerza bruta**: analizan todas las posibles contraseñas hasta llegar a descubrirla. Si las posibles son relativamente pocas, el programa llegará a probar todas las combinaciones en poco tiempo. ¡Ojo: hablar de millones es muy poco para un computadora! Además, ¿notaste que hay muchos sitios que bloquean el acceso a una cuenta cuando un usuario se equivoca en muchos intentos sucesivos al ingresar su contraseña? Así, evitan ser vulnerados por el uso de esta técnica.

#### ¡ÁBRETE, SÉSAMO!

“Alí Babá y los cuarenta ladrones” es un cuento popular incluido en la célebre recopilación de cuentos árabes medievales *Las mil y una noches*. Alí Babá era un honrado leñador que, sin proponérselo, descubre a una banda de ladrones que esconden los tesoros robados en una cueva cuya boca queda sellada pero que, mágicamente, puede abrirse usando la contraseña “Ábrete, sésamo”. Entonces, decide usar la clave a espaldas de los malhechores, para ingresar y llevarse riquezas, de modo que se vuelve también él un saqueador.



## Actividad 2

### Claves compartidas

 INDIVIDUAL

#### OBJETIVOS

- Introducir la necesidad de mecanismos de cifrado.
- Comprender los principios del cifrado simétrico.
- Reconocer ejemplos de uso habituales de cifrado simétrico.

#### MATERIALES

 Papel y lápiz

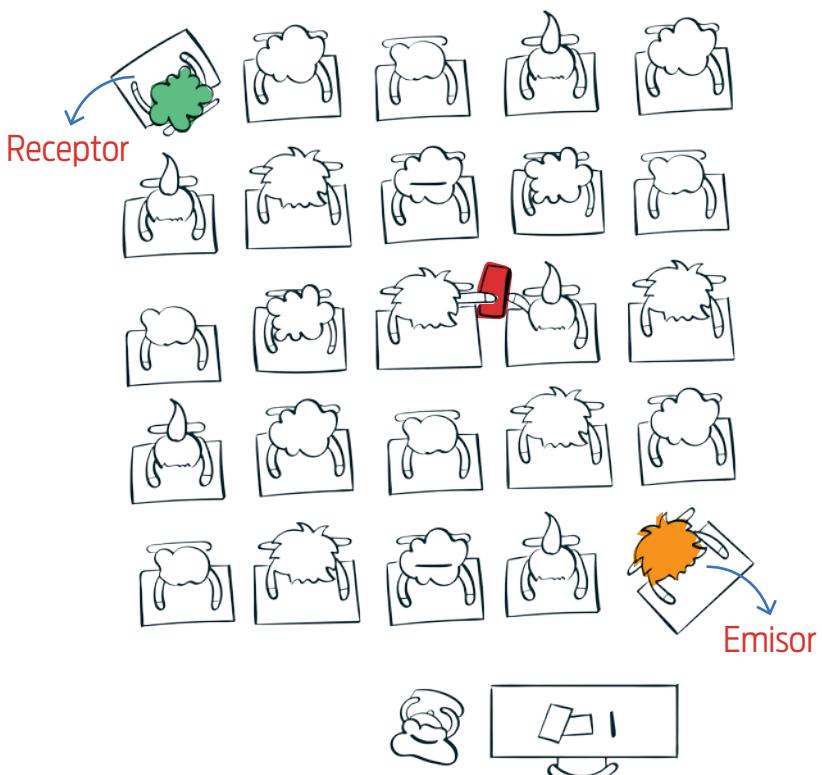
 Anexo “Código de cifrado simétrico”

 Ficha para estudiantes

#### DESARROLLO

En muchas ocasiones, al enviar mensajes, ya sea usando un soporte físico –por ejemplo, una carta en papel– o electrónico –como un correo electrónico–, resulta importante que el contenido solo sea leído por el destinatario. En otras palabras, que entre el despacho y la entrega, el mensaje no sea interceptado y observado por un tercero. En esta actividad presentaremos un mecanismo que permite el intercambio seguro de mensajes, que pertenece a la familia de los de **cifrado simétrico**; es decir, que requieren que tanto el emisor como el receptor (y solo ellos) conozcan una clave para poder escribir y leer mensajes.

Comenzamos eligiendo a dos estudiantes que harán de emisor y receptor de un mensaje privado, y los ubicamos en extremos opuestos del aula. Le pedimos a quien asuma el rol del receptor que se ubique de espaldas al curso; luego, le indicamos al emisor que escriba un mensaje en una hoja, se lo entregue a un compañero que tenga a su lado y, luego, que también él le dé la espalda al resto de sus compañeros. Entonces, los restantes estudiantes harán circular la hoja, pasándola de mano en mano, hasta hacerla llegar al pupitre del destinatario. Una vez que haya arribado, les indicamos tanto al emisor como al receptor que se den vuelta de modo de quedar de frente a sus compañeros nuevamente.



Mensaje viajando por un canal inseguro

A continuación les preguntamos al emisor y al receptor: “¿Pueden estar seguros de que solo ustedes dos leyeron el mensaje?”. Guiamos el intercambio de forma tal de concluir que cualquiera de los intermedios podría haber leído el mensaje sin que ellos lo sepan. Les comentamos: “Al comunicarse a través de Internet –por ejemplo, cuando envían un mensaje con un programa de mensajería instantánea– pasa lo mismo. La información entre quien manda un mensaje y quien lo recibe no viaja de manera directa entre sus respectivas computadoras, sino que pasa por varias otras intermedias; también en el universo digital, un mensaje podría ser interceptado y leído por terceros”.

Preguntamos a los estudiantes: “¿Se les ocurre alguna manera de evitar que esto suceda?”. Trabajamos con las ideas que surjan, tratando de destacar ventajas y desventajas de cada una. En caso de que no hayan surgido, podemos analizar y discutir las siguientes:

- **Poner el mensaje en un sobre cerrado.** De esta forma, el destinatario puede verificar si fue abierto en el camino; este enfoque permite saber si el mensaje fue interceptado, pero no impedir que eso suceda.
- **Usar una caja y cerrarla con un candado con combinación numérica.** Aquí, tanto el emisor como el receptor deben conocer la clave, tanto para poder cerrar el candado como para abrirlo.
- **Usar un lenguaje secreto solo conocido por el emisor y el receptor.** En este caso, hay que estar seguros de que nadie más conoce el lenguaje.

A continuación, indagamos: “¿Qué tienen en común las últimas dos ideas?”. Prestamos atención a sus observaciones y concluimos que, en ambas, es necesario que tanto el emisor como el receptor comparten una clave que nadie más conozca. En el primer caso, la **clave** es la combinación del candado; en el segundo, el lenguaje secreto. Les decimos: “La **criptografía** es un área de la matemática y la computación que se ocupa de desarrollar técnicas que permitan cifrar y descifrar mensajes de modo que, al enviarlos de un lado a otro en una red, pueda preservarse la confidencialidad. Cuando el emisor y el receptor comparten una clave, estamos en presencia de una técnica de cifrado simétrico”.

Para continuar, les entregamos una copia del anexo “Código de cifrado simétrico” al emisor y al receptor elegidos al comenzar la actividad (sin que nadie más pueda verlo), que contiene la clave que usarán para intercambiar mensajes de forma segura.

<b>ORIGINAL</b>	A	B	C	D	E	F	G	H	I	J	K	L	M	N
<b>CIFRADO</b>	F	R	J	B	O	X	V	I	D	Z	K	C	W	Q

<b>ORIGINAL</b>	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
<b>CIFRADO</b>	D	S	M	U	A	N	T	G	L	H	Y	E	P

Clave de cifrado

Les explicamos, solo a ellos, el mecanismo que usarán para cifrar y descifrar mensajes. El emisor, antes de enviar la misiva, sustituirá cada letra del texto por otra según lo indicado en la clave: cada *A* la reemplazará por una *F*, cada *B* por una *R*, y así siguiendo. Por ejemplo, la palabra *PIRULO* se reemplaza por *MDAGCS*. Para descifrar, el receptor llevará a cabo el proceso inverso: cambiará cada *F* por un *A*, cada *R* por un *B*, etc.

Invitamos al emisor, nuevamente, a que escriba un mensaje al receptor. En este caso, antes de despachar la carta, elegimos a un estudiante testigo al que el emisor le confiará en secreto el mensaje sin cifrar. Además, autorizamos a que los estudiantes intermedios lean el mensaje mientras lo dirigen hacia el pupitre del destinatario. Cuando el papel haya sido entregado preguntamos: “¿Alguno entendió el mensaje?”. Probablemente, todos contesten que no. Entonces, el receptor lo descifrará, lo escribirá en un papel y le pedirá al testigo que diga en voz alta el contenido de la nota. Finalmente, para que no queden dudas, mostrará la hoja con el mensaje descifrado.

A continuación, entregamos la ficha a los estudiantes y les pedimos que resuelvan las dos consignas. La primera pide cifrar algunos mensajes y la segunda descifrar otros, en ambos casos usando la misma clave presentada previamente. A continuación se observan las respuestas:

CIFRAR MENSAJES	→	JDXAFA WOQSFZON
LUCES ESTROBOSCÓPICAS	→	CGJON ONTASRSNJSMDJFN
CUCARACHA	→	JGJFAFJIF
TARTA DE CHOCLO	→	TFATF BO JISJCS

Solución de la consigna 1



Cifrado de Pirulo

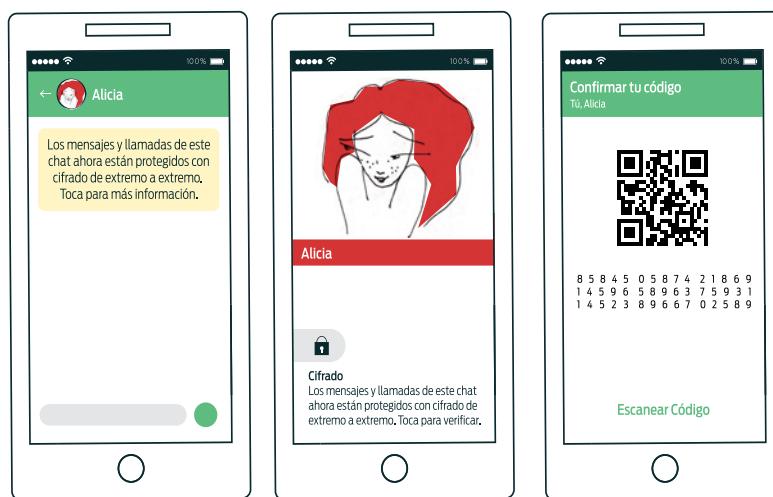


Pirulo descifrado

- |                            |   |                            |
|----------------------------|---|----------------------------|
| ¿UGO MAOTOQBO GNTOB BO WD? | → | ¿QUÉ PRETENDE USTED DE MÍ? |
| NO CO ONJFMS CF TSATGVF    | → | SE LE ESCAPÓ LA TORTUGA    |
| JSQ CFN WFQSN CDWMDFN      | → | CON LAS MANOS LIMPIAS      |
| CF MOCSTF QS BSRCF         | → | LA PELOTA NO DOBLA         |

Solución de la consigna 2

Hacemos una puesta en común y luego, para anclar lo trabajado con ejemplos reconocibles para los estudiantes, comentamos: “Las aplicaciones de mensajería instantánea que se utilizan con mayor frecuencia suelen cifrar los mensajes que enviamos a nuestros contactos e informarnos al respecto. Por ejemplo, al comenzar a hablar con un nuevo contacto, WhatsApp informa que los mensajes están protegidos con cifrado de extremo a extremo, lo que significa que se cifran en el teléfono del emisor y solo pueden ser descifrados en el teléfono del destinatario. Además, entrando a la pantalla de opciones de cualquier contacto y seleccionando la opción “Cifrado”, se pueden ver dos representaciones de la clave compartida con ese contacto: como código QR y como una serie de números. Las mismas coinciden con las que aparecen al verificar nuestra propia información en el teléfono del contacto en cuestión. Si bien la técnica de cifrado que usa WhatsApp es más sofisticada que la que usamos en la actividad, también en este caso se trata de un método de cifrado simétrico: el emisor y el receptor comparten la misma clave”.



Cifrado en WhatsApp

## CIERRE

Les contamos a los estudiantes que durante la Segunda Guerra Mundial, para realizar comunicaciones militares, la Alemania nazi utilizó una máquina llamada Enigma que usaba una técnica de cifrado simétrica. Sin embargo, llevando adelante actividades de inteligencia, los ejércitos aliados obtuvieron algunas de las claves, lo que les permitió descifrar una gran cantidad de mensajes secretos. Estos acontecimientos resultaron clave para el desenlace de la guerra.

## ANEXO: CÓDIGO DE CIFRADO SIMÉTRICO

ANEXO

ORIGINAL	A	B	C	D	E	F	G	H	I
CIFRADO	F	R	J	B	O	X	V		
<hr/>									

ORIGINAL	H	I	J	K	L	M	N	N	
CIFRADO	I	D	Z	K	C	W	Q	D	
<hr/>									

ORIGINAL	O	P	Q	R	S	T	U	V	
CIFRADO	S	M	U	A	N	T	G	L	
<hr/>									

ORIGINAL	V	W	X	Y	Z				
CIFRADO	L	H	Y	E	P				
<hr/>									

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# CLAVES COMPARTIDAS

La criptografía es un área de la matemática y la computación que se ocupa de desarrollar técnicas que permitan cifrar y descifrar mensajes de modo que, al enviarlos de un lado a otro, pueda preservarse la confidencialidad. Para garantizar que solo el emisor y el receptor comprendan los mensajes, hay técnicas que requieren que ambos compartan una clave (que nadie más conozca). Las técnicas con esta característica se llaman de **cifrado simétrico**.



Uno de los métodos simétricos más antiguos es el de cifrado por sustitución. Para cifrar un mensaje, hay que reemplazar cada letra por otra, siguiendo un criterio solo conocido por el emisor y el receptor. Luego, para descifrarlo, hay que hacer el reemplazo inverso.

A continuación se muestra un posible esquema de reemplazos:

ORIGINAL	A	B	C	D	E	F	G	H	I	J	K	L	M	N
CIFRADO	F	R	J	B	O	X	V	I	D	Z	K	C	W	Q

ORIGINAL	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z
CIFRADO	D	S	M	U	A	N	T	G	L	H	Y	E	P

1. ¿Cómo se codifican los siguientes mensajes con el esquema de cifrado propuesto?

CIFRAR MENSAJES



\_\_\_\_\_

LUCES ESTROBOSCÓPICAS



\_\_\_\_\_

CUCARACHA



\_\_\_\_\_

TARTA DE CHOCLO



\_\_\_\_\_

2. ¡Te llegaron estos mensajes! ¿Qué dicen?

¿UGO MAOTOQBO GNTOB BO WD?



\_\_\_\_\_

NO CO ONJFMS CF TSATGVF



\_\_\_\_\_

NOMBRE Y APELLIDO:

CURSO:

FECHA:

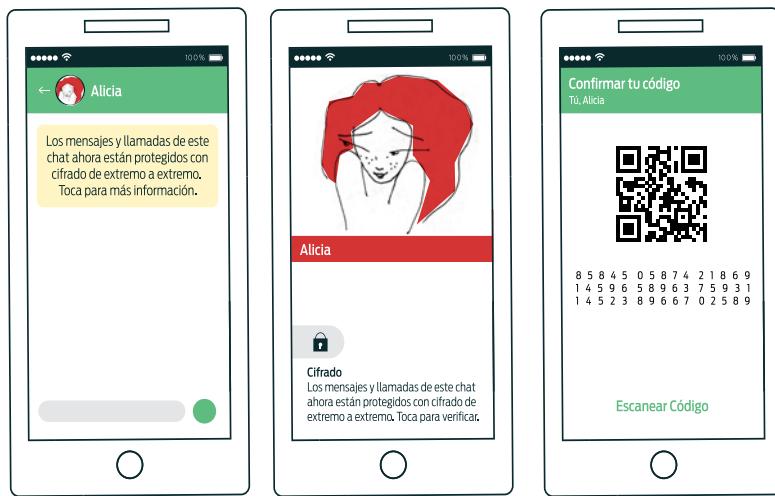
JSQ CFN WFQSN CDWMDFN → \_\_\_\_\_

CF MOCSTF QS BSRCF → \_\_\_\_\_

### ¿SABÍAS QUE...

... las aplicaciones de mensajería instantánea suelen cifrar los mensajes? Por ejemplo, la primera vez que enviamos un mensaje a un contacto, WhatsApp informa que el intercambio viaja cifrado de extremo a extremo, lo que significa que se cifran en el teléfono del emisor y solo pueden ser descifrados en el teléfono del destinatario.

Entrando a la pantalla de opciones de cualquier contacto y seleccionando la opción "Cifrado", se pueden ver dos representaciones de la clave compartida con ese contacto: como código QR y como una serie de números.



### SITIOS SEGUROS

Es sumamente importante que, antes de enviar información sensible a un sitio de Internet (contraseñas, mensajes privados, claves bancarias, etc.), verifiquemos que la comunicación con el sitio esté cifrada.



Cuando usamos un navegador de Internet podemos chequear si la comunicación es segura. En ese caso, en la barra de direcciones aparece un candado y la dirección debería comenzar con "https://", a diferencia de las no seguras, que comienzan con "http://".



## Actividad 3

### Claves públicas y privadas<sup>1</sup>



#### OBJETIVOS

- Introducir los métodos de cifrado asimétrico.
- Comprender la diferencia entre claves públicas y privadas.

#### MATERIALES

- Pizarrón
- Tizas
- Anexo “Clave pública”
- Anexo “Clave privada”

#### DESARROLLO

El objetivo de esta actividad es introducir los mecanismos de **cifrado asimétrico**. A diferencia de lo que sucede con los de cifrado simétrico, en estos no hay una clave secreta que comparten los involucrados en el intercambio de mensajes. En su lugar, una persona que quiera recibir mensajes que solo ella pueda descifrar, tiene que tener dos claves: una **clave pública**, que distribuye abiertamente para que terceros puedan encriptar mensajes, y una **clave privada** solo conocida por él, que permite (y es necesaria para) desencriptar los mensajes cifrados con la clave pública.

Comenzamos la actividad indagando entre los estudiantes: “Supongamos que quisieran poder recibir mensajes de algún amigo de la clase que solo ustedes pudiesen entender, ¿qué harían?”. Es esperable que algún estudiante conteste que, en ese caso, tendría que compartir una clave con su interlocutor para poder tanto cifrar como descifrar mensajes. Les decimos: “Claro, esto es como mandar un mensaje en un caja cerrada con un candado cuya combinación solo la conocen ustedes y su interlocutor, ¿no es cierto? Ahora bien, supongan que quieren poder recibir mensajes de más de una persona; por ejemplo, imaginen que quieren poder hacerlo de cualquiera de sus compañeros de la clase. ¿Les parecería una buena idea que todos conozcan la combinación?”. Guiamos el intercambio para concluir que, en ese caso, cualquiera podría abrir la caja en cualquier momento, con lo que la privacidad del mensaje podría vulnerarse con facilidad.

Les damos tiempo a los estudiantes para que piensen alternativas y las comparan con el resto de la clase. En caso de que no haya surgido de algún estudiante, les presentamos la siguiente opción: “¿Qué pasaría si uno tuviese un candado que se abre con una llave –y no una combinación numérica– cuya única copia la tiene el dueño? ¿Qué podría hacer?”. Reflexionamos acerca de que, de este modo, podría dejar el candado abierto junto a la caja, para que quien quiera mandarle un mensaje, lo coloque en la caja y cierre el candado. Entonces, el mensaje solo podría ser accesible para el dueño del candado. “¡Muy bien! Pero... si quisieramos que muchos nos puedan escribir harían falta un montón de candados! Por suerte, en el mundo digital podemos poner muchísimas copias de un único candado digital, con lo que la cantidad de interlocutores no es un problema”.

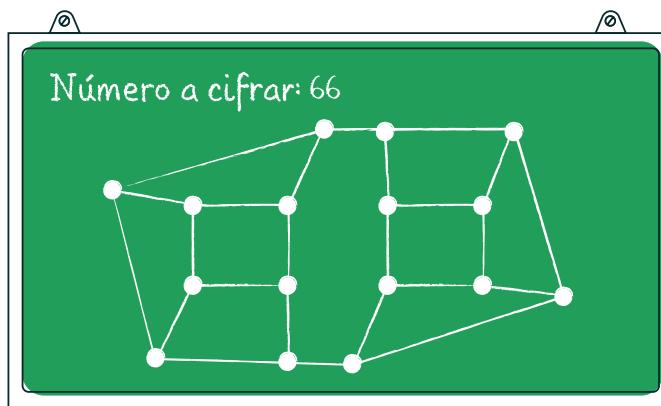
Continuamos: “Hay métodos de encriptación, llamados de **cifrado asimétrico**, en los que un usuario tiene dos claves: una que comparte públicamente con todo el mundo, que solo sirve para cifrar mensajes, y otra privada –solo conocida por él–, que se utiliza para descifrarlos. En el ejemplo, el candado sería la clave pública y la llave, la clave privada”.

<sup>1</sup> Adaptación de la actividad “Sharing Secrets” de CS Unplugged, disponible en <http://bit.ly/2kA2GbS>.

## Un juego de números secretos

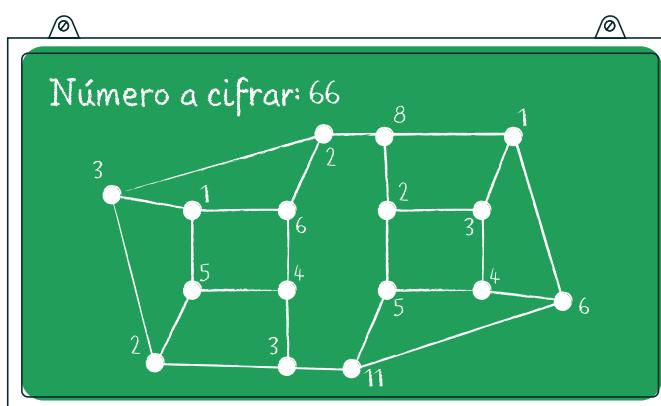
Una vez presentada la idea de cifrado asimétrico, propondremos a los estudiantes un juego en el que distribuiremos a todo el curso una clave pública, que usarán para cifrar números que solo podrán ser descifrados por uno de ellos, poseedor de la clave privada. Es importante que los estudiantes realicen el procedimiento de cifrado cuidadosamente, porque de ello depende que la actividad funcione de acuerdo a lo esperado.

Para explicar de manera gráfica el proceso de cifrado, cifraremos un número frente a toda la clase. Por ejemplo, podemos mostrar cómo encriptar el número 66. Comenzamos copiando en el pizarrón la clave pública, que consiste en una serie de vértices, algunos de ellos conectados por líneas.



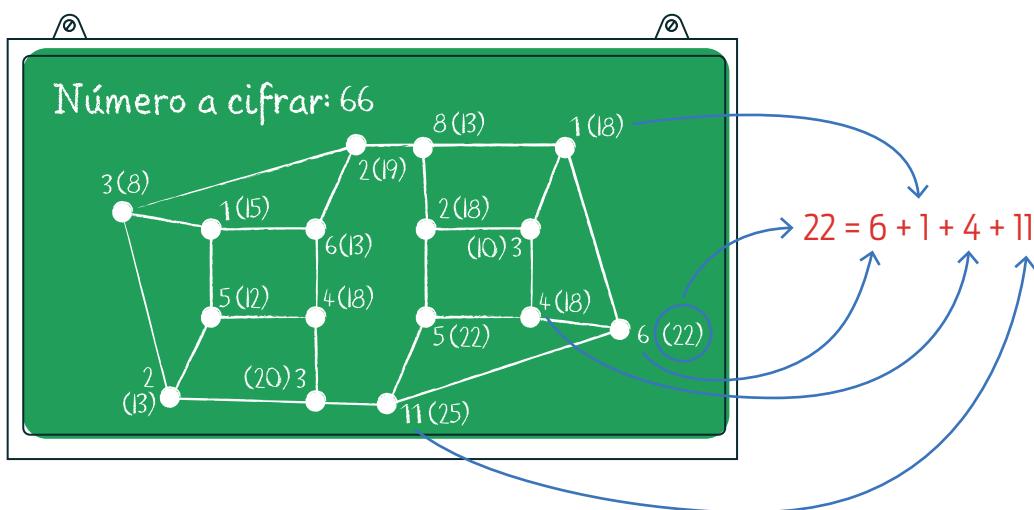
Clave pública

Les decimos: "Lo que ven en el pizarrón es la clave pública que nos permitirá cifrar un número. Una vez elegido el número que queremos cifrar, lo primero que hay que hacer es escribir un número junto a cada vértice de forma tal que, al sumarlos, el resultado sea el número a cifrar. Supongamos, por ejemplo, que queremos cifrar el sesenta y seis. Una posibilidad sería escribir los siguientes, ya que  $3 + 2 + 1 + 5 + 2 + 6 + 4 + 3 + 8 + 2 + 5 + 11 + 3 + 4 + 1 + 6 = 66$ ". Agregamos a la clave del pizarrón los correspondientes números.



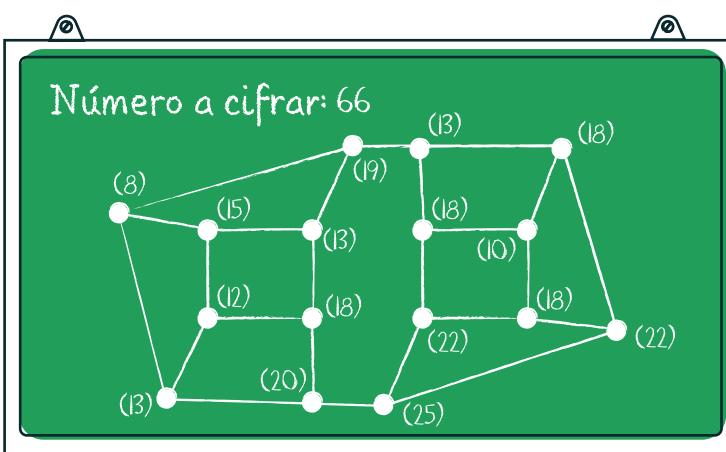
La suma de los números de los vértices es igual al número secreto

Continuamos: "Ahora, en cada vértice hay que escribir el resultado de sumarle a su valor los valores de todos aquellos vértices con los que se encuentra conectado. Por ejemplo, en el caso del vértice inferior del lateral derecho de la clave, hay que sumar 6 –su valor– más 1 más 4 más 11 –los valores de sus vecinos-. Entonces, allí, escribimos 22 ( $6 + 1 + 4 + 11 = 22$ ). Lo mismo hay que hacer con todo el resto de los vértices". Agregamos al dibujo los valores correspondientes a cada vértice.



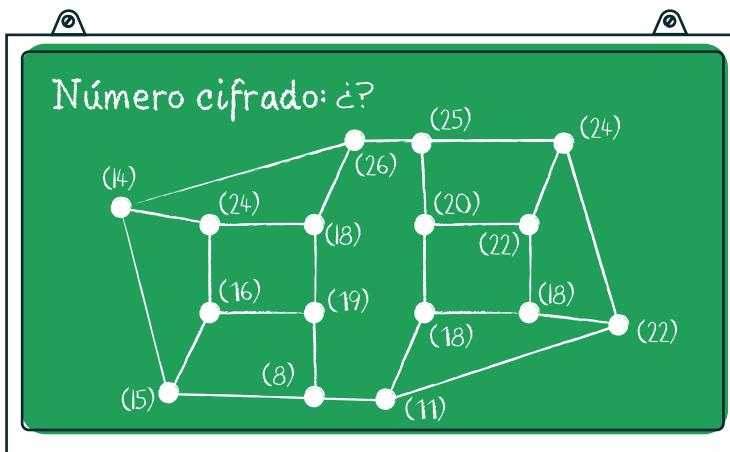
Incorporación de la suma de los vértices adyacentes

Seguimos: "Para completar el cifrado, solo debemos borrar los números que originalmente escribimos junto a los vértices, cuya suma es 66.



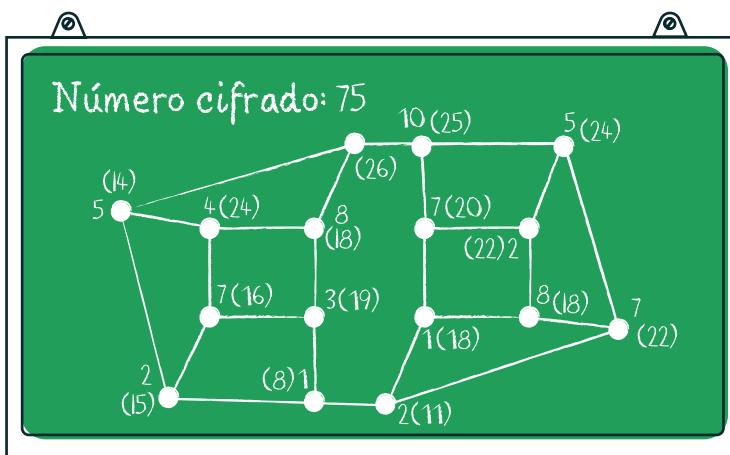
Se borran los números originales

Indagamos: "De este modo, finalizamos el proceso de cifrado. Ahora, aun cuando sepan cómo se construyó, ¿alguno sería capaz de reconocer que lo que ve es el resultado del proceso de cifrar el 66?". Copiamos entonces un nuevo ejemplo ya cifrado en el pizarrón –correspondiente al número 75– y desafiamos a la clase a que nos diga cuál era el número original.



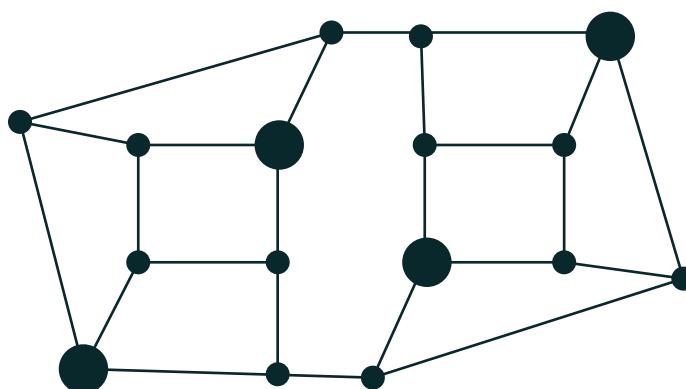
El número 75 cifrado

Les damos unos minutos para que traten de descifrar el número, lo que no conseguirán, pues, justamente, de eso se tratan los métodos asimétricos: una vez que se cifra un mensaje, solo puede descifrarse usando su correspondiente clave privada –que no tienen–. "Muy bien, en este caso el número cifrado es el 75". Entonces, para que puedan corroborarlo, copiamos la descomposición del 75 a partir de la cual se llegó al código del pizarrón. De este modo podrán ver que (i) si suman todos los nuevos números se obtiene 75 como resultado y (ii) que si realizan el proceso de cifrado obtendrían el gráfico tal como originalmente lo exhibimos.



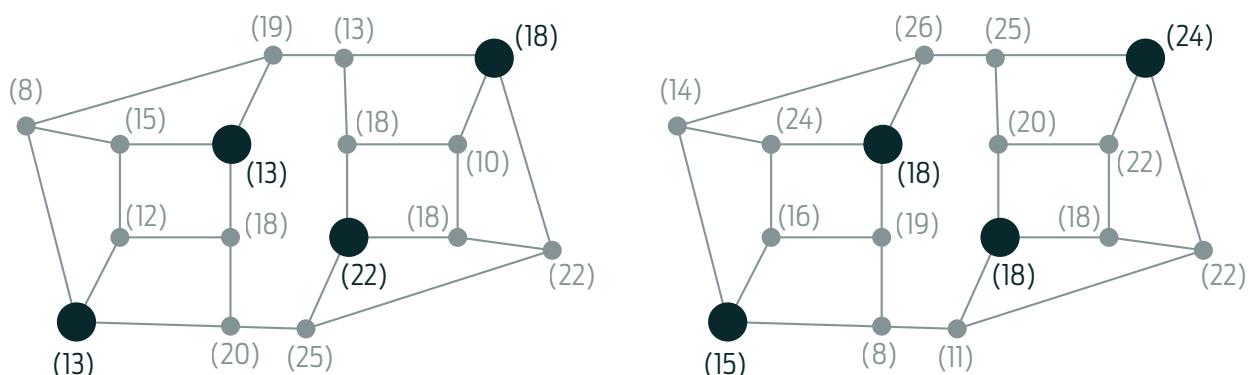
Descomposición del número 75

Una vez que todos se hayan convencido de que no pueden reconstruir fácilmente un número cifrado, elegimos a un estudiante al que le entregaremos la clave privada –que está en el anexo “Clave privada”– y le explicaremos cómo usarla para descifrar un número. Entonces, le decimos, sin que nadie más pueda escucharnos: “Mirá, acá tenés la clave privada. Descifrar un número con ella es realmente muy fácil: solo tenés que sumar los números entre paréntesis que se encuentran junto a los cuatro vértices grandes”.



Clave privada

Le damos unos instantes para que aplique la técnica de descifrado sobre los dos ejemplos vistos para convencerlo de que el método es correcto:  $13 + 13 + 22 + 18 = 66$  y  $15 + 18 + 18 + 24 = 75$ .



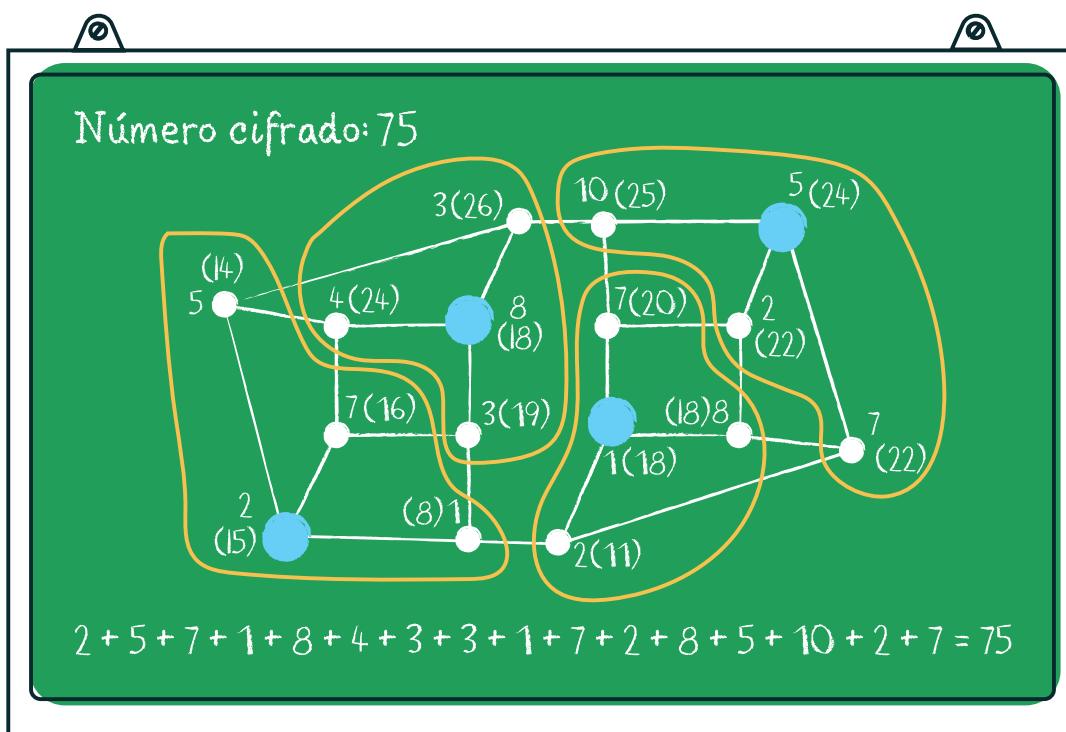
Clave privada que descifra el 66 y el 75

Excluyendo al estudiante al que le entregamos la clave privada, le pedimos al resto que formen grupos de cuatro integrantes. A cada grupo le entregamos 4 copias de la clave pública del anexo “Clave pública” para que cifren números que ellos secretamente elijan. Lo más conveniente es que, para cada número que quieran cifrar, usen dos de las copias: una para llevar adelante el proceso de cifrado con sus respectivas cuentas, y otra para copiar los números resultantes de dicho proceso. Insistimos: “¡Presten atención al realizar el cifrado! ¡Es muy importante que sigan bien todos los pasos!”.

Una vez que los grupos finalicen el cifrado de los números les pedimos que, en primer lugar, los hagan circular por el resto de la clase para que todos intenten descifrarlo; y, luego de los (infructuosos) intentos, que entreguen las copias al estudiante poseedor de la clave privada que, usándola (y sin mostrarla a nadie más), podrá descifrar cada uno de los números secretos.

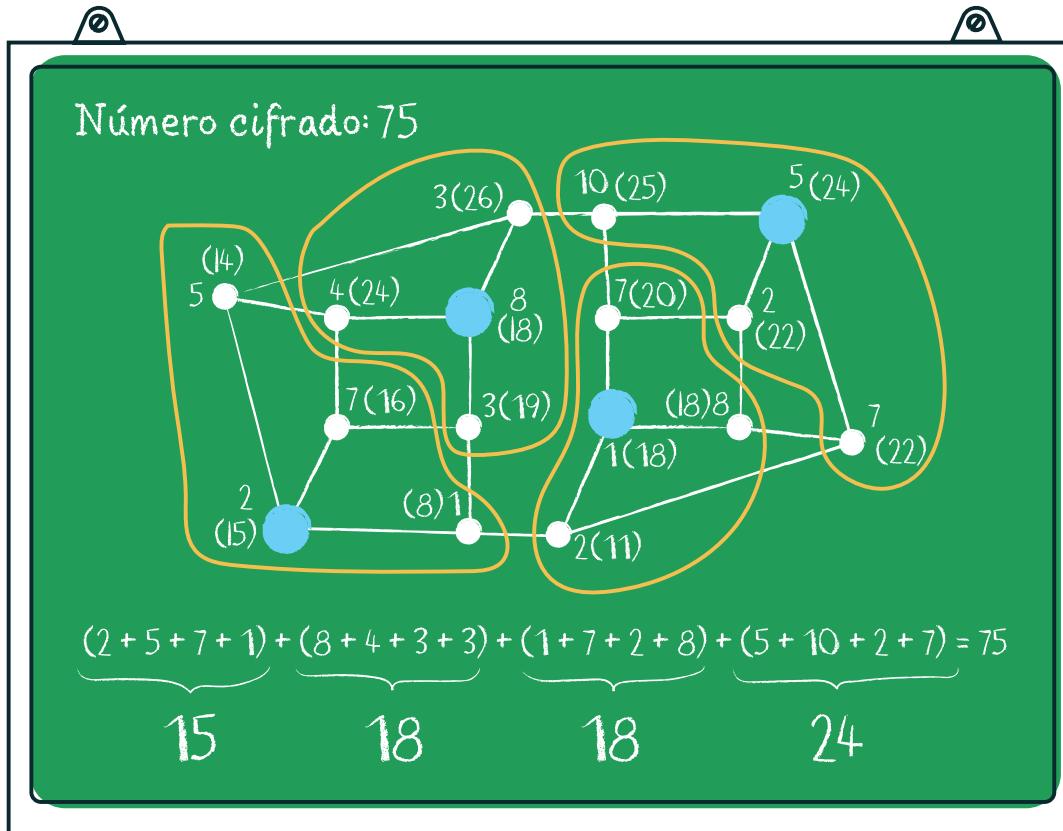
Les decimos: "En este caso, para mandar un número secreto no hizo falta que haya una clave compartida entre el emisor y el receptor. En su lugar hubo dos claves: una pública, que estuvo al alcance de todos y que permitió que cualquiera mandara un número confidencial; y una privada, solo conocida por el receptor, con la que este pudo descifrar los números recibidos. Extraño, ¿no?". Escuchamos las reflexiones de los estudiantes y, a continuación, les mostramos la clave privada y les explicamos el mecanismo de descifrado.

Podría ocurrir –y sería saludable que así fuese– que algún estudiante pregunte por qué el método funciona. Entonces, en el gráfico del pizarrón, agrupamos los vértices como se muestra en la figura.



La suma de los números sin paréntesis es el número a cifrar

Les decimos: "Recuerden, en primer lugar, que el número que queremos cifrar –en el ejemplo, el 75– coincide con la suma de todos los números que no están entre paréntesis". Escribimos en el pizarrón: " $2 + 5 + 7 + 1 + 8 + 4 + 3 + 3 + 1 + 7 + 2 + 8 + 5 + 10 + 2 + 7 = 75$ ". Seguimos: "Fíjense que los números entre paréntesis junto a los vértices grandes son el resultado de la suma de ellos mismos con todos sus vecinos". Acto seguido, asociamos en el pizarrón las sumas parciales que resultan en estos números.

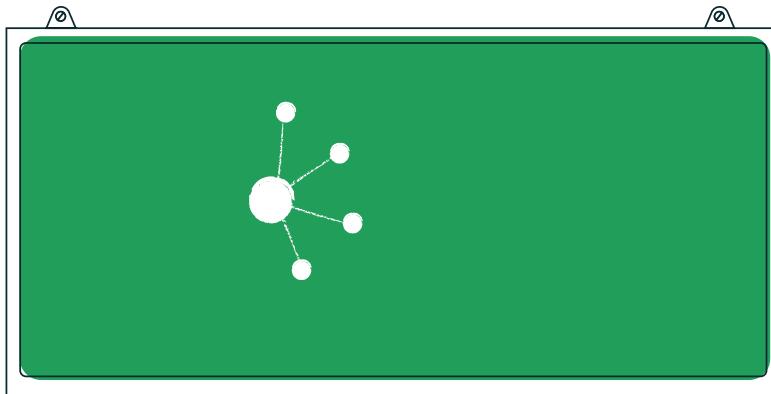


Los números entre paréntesis son el resultado de sumar un vértice con todos sus vecinos

Continuamos la explicación: “Aquí está el quid de la cuestión: (i) por un lado, todo vértice es parte de la suma de **algún** vértice grande, y (ii) todo vértice se suma en **un único** vértice grande. Entonces, al sumar los vértices grandes, iestamos sumando los valores de todos los vértices del dibujo sin sumar ninguno de ellos más de una vez! Esto, como resultado, da el valor que originalmente ciframos”.

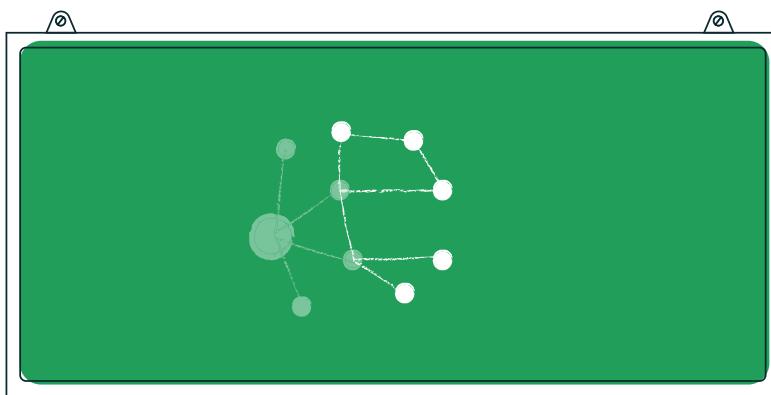
Les preguntamos: “Ahora, sabiendo esta propiedad y conociendo la clave pública, ¿les parece que ustedes podrían haber deducido cuáles eran los vértices grandes de la clave privada y, en consecuencia, ser capaces de descubrir los números cifrados?”. Es posible que algún estudiante conteste que sí. “Es cierto que, en este caso, se pueden seleccionar distintos vértices buscando que se cumpla la propiedad descrita. Sin embargo, créanme que si tomamos una clave lo suficientemente grande, con cincuenta o cien vértices por ejemplo, encontrar puntos grandes que tengan esta propiedad –es decir, que sirvan como clave privada– es extremadamente difícil, incluso para una computadora. ¡Podría llevarle años quebrar el código!”.

A continuación les decimos a los estudiantes: “Muy bien, es muy difícil obtener la clave privada solo conociendo la clave pública, pero... ¿no será también difícil generar pares de claves públicas y privadas?”. Damos unos instantes para que piensen al respecto y comparten sus ideas. Luego, proseguimos: “Realmente no lo es; veamos un método simple para generarlas. Comenzamos agregando un vértice grande que conectamos con otros chiquitos (tantos como queramos)”.



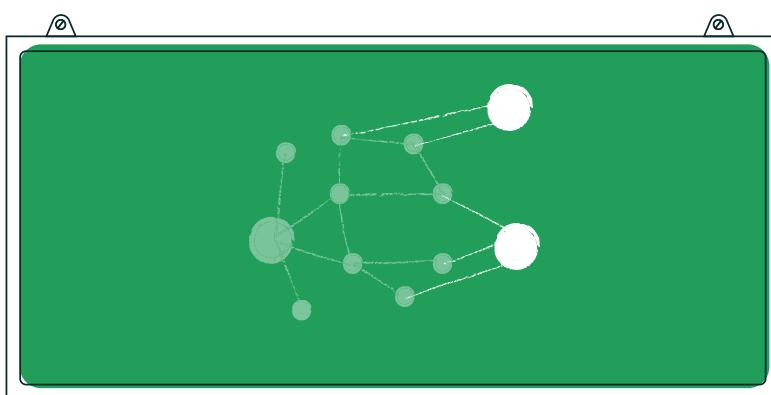
Generación de una clave, paso I

Seguimos: "Luego, a algunos vértices chiquitos los conectamos con otros chiquitos. Además de conectarlos a nuevos vértices que agreguemos, también podríamos conectarlo a otros de los previamente incorporados".



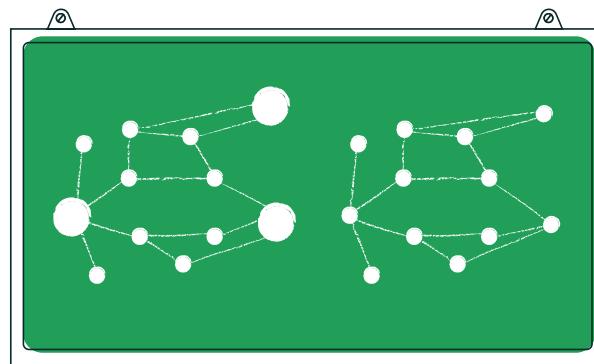
Generación de una clave, paso II

Continuamos: "Ahora agregamos otra vez vértices grandes, siempre teniendo cuidado de que cada vértice pequeño quede conectado a uno y solo uno de los grandes".



Generación de una clave, paso III

Proseguimos: “Para generar claves más grandes, podemos repetir este proceso tantas veces como queramos teniendo cuidado de que: (i) de todos los vértices chicos salga una línea a algún vértice grande; (ii) que ningún vértice chico quede conectado a más de uno grande; y (iii) que los vértices grandes no queden conectados entre sí. Finalmente, armamos la clave pública reemplazando los vértices grandes por vértices chicos”.



Una clave privada y su correspondiente clave pública

Les decimos a los estudiantes: “¿Ven? ¡Es muy fácil generar claves privadas (con sus correspondientes públicas), pero muy difícil descubrirlas!”.

Entonces, invitamos a que cada grupo (i) genere una clave privada y una pública; (ii) distribuya la clave pública a los otros grupos; (iii) cifren números con las claves públicas recibidas de otros grupos; (iv) entreguen los números cifrados a los compañeros que crearon la respectiva clave; y (v) que descifren los números que se cifraron usando sus claves.



#### SOBRE EL MÉTODO PRESENTADO

Si bien es cierto que, partiendo de una clave pública, es computacionalmente muy costoso (en tiempo) encontrar una privada con las características presentadas, el número cifrado puede descubrirse por otro camino, resolviendo un conjunto de ecuaciones lineales. Sin embargo, sugerimos no introducirlo porque (i) es matemáticamente complejo para estudiantes de la edad a la que está dirigido este material; y (ii) lo importante es introducir los procedimientos de cifrado asimétrico que usan pares de claves públicas y privadas (y no formas de quebrarlos). Una explicación del método expuesto se encuentra en <http://bit.ly/2kA2GbS>.

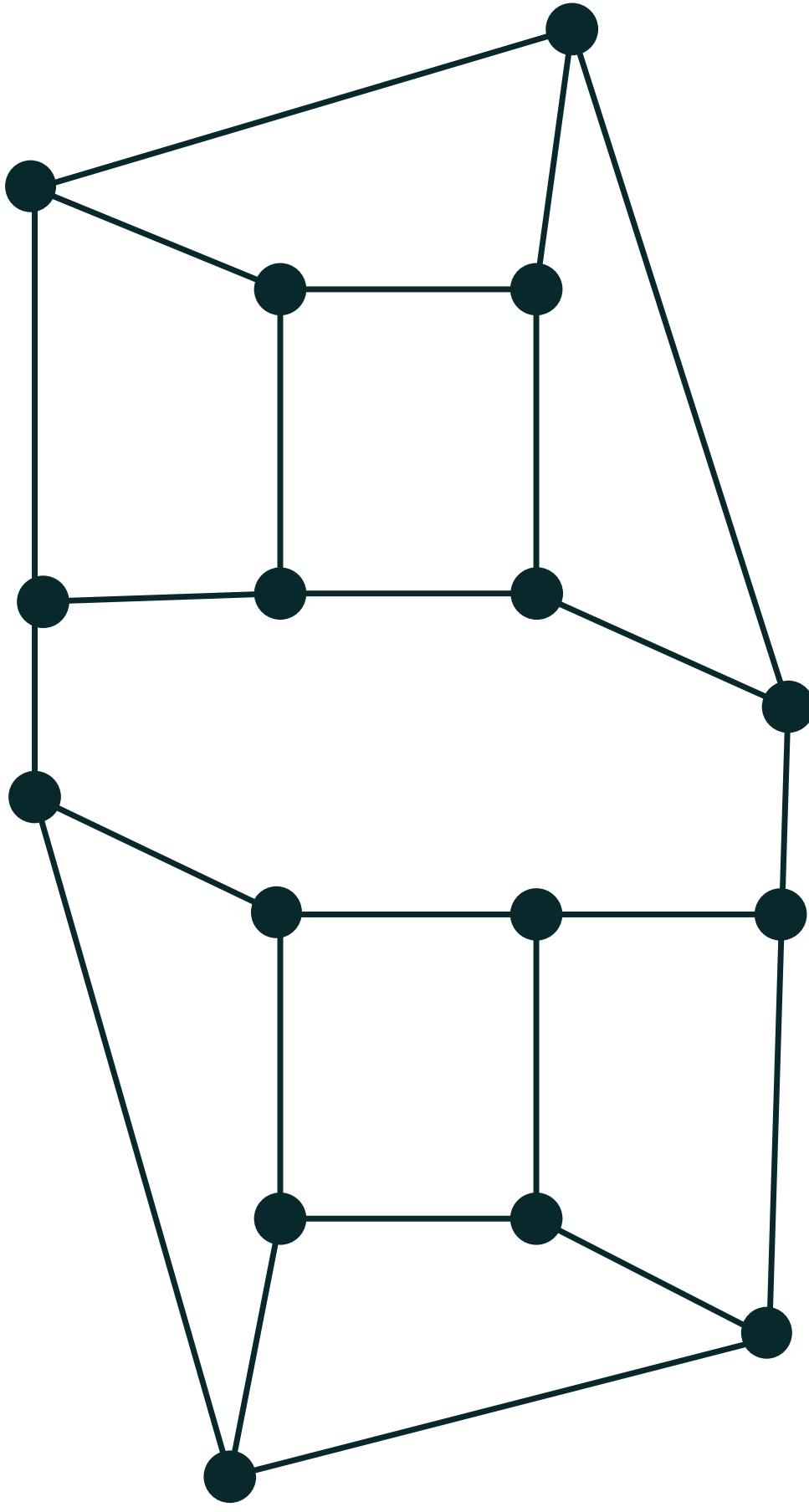
#### CIERRE

Les comentamos a los estudiantes que, en general, los mensajes que se envían entre dos computadoras –por ejemplo, usando un programa de mensajería instantánea– no consisten únicamente en números, ni mucho menos en un único número. Sin embargo, toda la información que maneja una computadora es internamente codificada con números, con lo que el mecanismo presentado puede generalizarse para el cifrado de datos de distinta naturaleza. Igual, aclaramos, los cifrados asimétricos que se usan actualmente son sustancialmente más sofisticados, y se basan en complejas teorías matemáticas.<sup>1</sup>

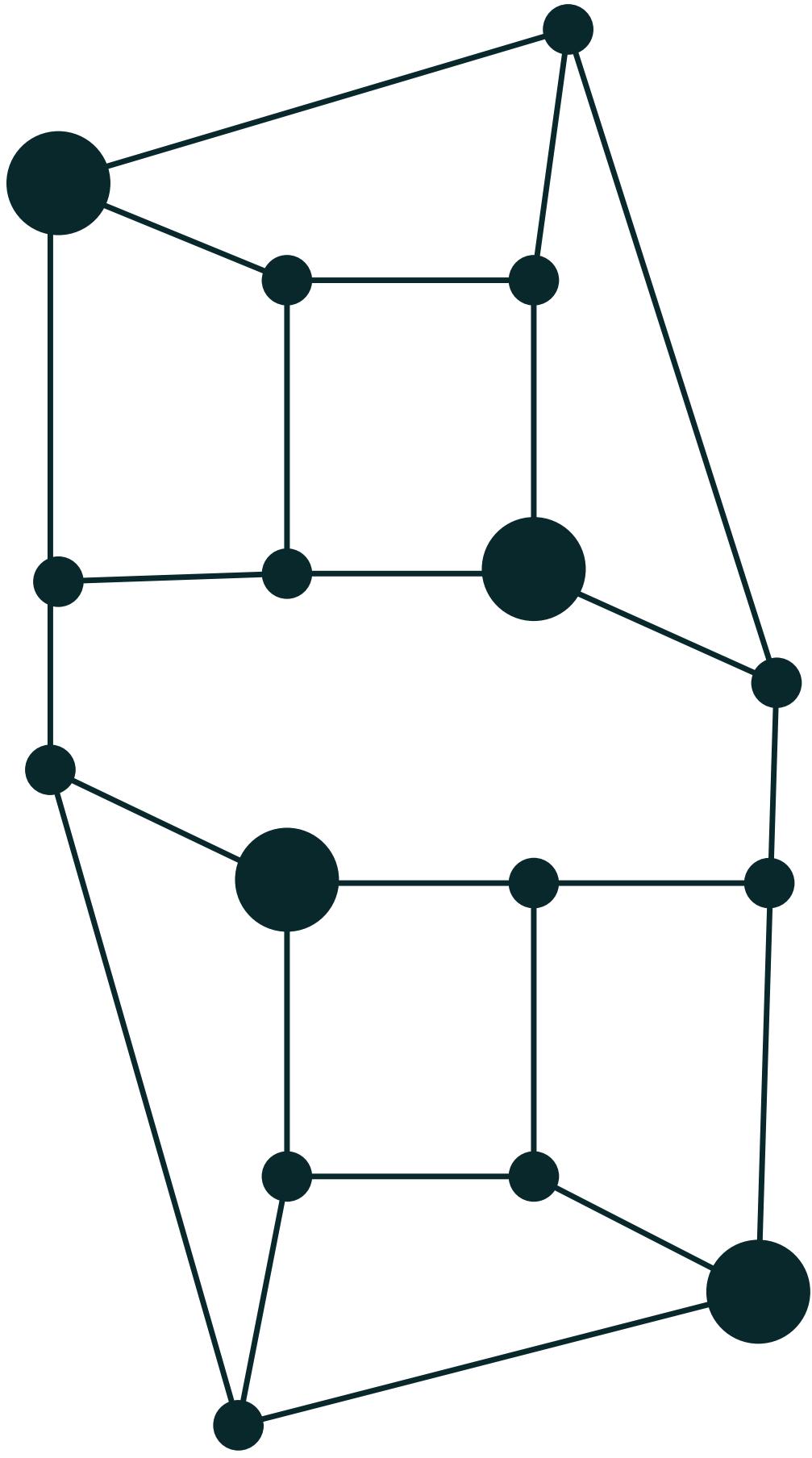
<sup>1</sup> Las técnicas más usadas se basan en que no se conoce un método eficiente para descomponer un número entero como producto de sus factores primos.

## ANEXO: CLAVE PÚBLICA

ANEXO



ANEXO

**ANEXO: CLAVE PRIVADA**

# 02

# PROGRAMAS Y EVENTOS

**SECUENCIA DIDÁCTICA 1**  
PROGRAMACIÓN DIRIGIDA POR  
EVENTOS  
Obedientes y reactivos  
¡Hola, amigo!

**SECUENCIA DIDÁCTICA 2**  
INTERFAZ GRÁFICA,  
COMPORTAMIENTO Y PUBLICACIÓN  
DE APLICACIONES  
La interfaz de *Dibujarte*  
*Dibujarte* en acción

Un programa es, ante todo, una descripción muy precisa de cómo esperamos que se comporte una máquina. En general, se trata de una colección de instrucciones escrita en un lenguaje de programación que, a diferencia de un lenguaje coloquial, posee una sintaxis muy estricta y una semántica que no admite múltiples interpretaciones.

En este capítulo se introduce la noción de **programa**, y se hace especial énfasis en aquellos en los que existe una interacción con el usuario. Las secuencias didácticas llevarán a los estudiantes a construir sus primeras aplicaciones, comprender qué son los **eventos**, cómo se programa una respuesta frente a ellos y la importancia de las **interfaces gráficas**.



## Secuencia Didáctica 1

# PROGRAMACIÓN DIRIGIDA POR EVENTOS

La **programación dirigida por eventos** es un paradigma de programación en el cual la ejecución de las instrucciones de un **programa** está determinada por **eventos** que, *a priori*, no se sabe cuándo ocurrirán: por ejemplo, que un usuario presione un botón, que un sensor de temperatura indique que se ha alcanzado una cierta cantidad de grados, etc. En estos casos lo que se programa es, justamente, el **manejo de eventos**; es decir, la respuesta que tendrá el programa frente a la ocurrencia de los eventos.

En esta secuencia didáctica se introducen las nociones de programa, evento y manejo de eventos. Además, se propone un recorrido que permitirá a los estudiantes conocer el entorno de MIT App Inventor 2 para luego construir un programa sencillo que podrán ver en funcionamiento en sus teléfonos.

### OBJETIVOS

- Introducir la noción de programa.
- Presentar la programación dirigida por eventos.
- Realizar una práctica de programación en la computadora.

## Actividad 1

### Obedientes y reactivos



#### OBJETIVOS

- Introducir el concepto de programa.
- Introducir la noción de evento.
- Diferenciar los eventos de su manejo.

#### MATERIALES

- Papel
- Bolígrafo
- Anexo “Cartas para manejo de eventos”

#### DESARROLLO

El objetivo de esta actividad es introducir la idea de **programa** como una descripción del comportamiento que se espera que tenga una máquina. Se hará hincapié en que los programas se escriben usando **lenguajes de programación** que, a diferencia de los lenguajes coloquiales, tienen una sintaxis muy rígida y una semántica inequívoca. Por último, se presenta las ideas de **evento** –como un suceso que ocurre en un momento *a priori* desconocido– y de **manejo de eventos** –como la respuesta de un programa frente a la ocurrencia de un evento.

Antes de comenzar la actividad, es recomendable tener impresas las copias de los dos juegos de cartas del anexo “Cartas para manejo de eventos”. De cada uno, hace falta la mitad de la cantidad de estudiantes que tenga el curso. Por ejemplo, para un curso de 30 alumnos se necesitan 15 copias de cada juego de cartas.

#### Instrucciones y lenguajes

Comenzamos contándoles a los estudiantes: “¿Sabían que, por más que sea obediente, a veces me comporto como un robot? Eso sí, soy un robot bastante rústico: lo único que puedo hacer es sentarme en una silla, pararme, aplaudir y saltar. Vamos hacer una prueba: ustedes denme instrucciones y veamos qué pasa. Eso sí, vayan turnándose para dar las órdenes, porque el robot solo las entiende cuando no hay barullo alrededor”.

Nosotros, como robots, solo reconoceremos las instrucciones **Sentate**, **Parate**, **Aplaudi** y **Saltá**. Cada vez que alguien diga alguna de ellas, realizaremos la acción correspondiente. Ante cualquier instrucción que no sea una de estas cuatro, contestaremos con voz robótica “No entiendo la instrucción”, aun cuando el sentido sea equivalente a alguna de las mencionados –por ejemplo, si alguien dijera: “Ponete de pie” en lugar de “Parate”–. Una vez que se hayan mencionado las cuatro instrucciones, las escribimos en el pizarrón.



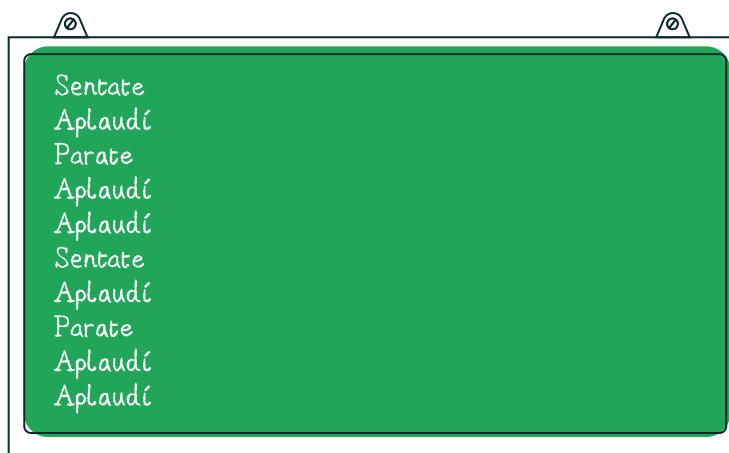
Instrucciones que reconoce el robot

Luego les preguntamos: “¿Por qué les parece que el robot no reaccionaba cuando le decían ‘Ponete de pie’, ‘Colocá los glúteos sobre la silla’ o ‘Aplaudir’?”. Buscamos, de este modo, que reconozcan que, como robots, seguimos instrucciones al pie de la letra, sin cuestionarlas ni reflexionar sobre ellas. Continuamos: “Al igual que a este robot, a las computadoras les podemos dar instrucciones para que lleven adelante algunas acciones. Las computadoras siguen indicaciones al pie de la letra. No tienen la posibilidad de reflexionar ni de inferir la semántica de lo que les indicamos. Para definir el comportamiento de una computadora, usamos lenguajes de programación, que nos proveen un conjunto acotado de instrucciones cuya semántica es inequívoca”.

### Programas

Continuamos la actividad preguntando a los estudiantes: “¿Qué creen que son los programas?”. Escuchamos con atención sus respuestas y les comentamos: “Como hemos dicho, los lenguajes de programación proveen instrucciones que podemos darle a una máquina. Un **programa** se construye combinando instrucciones para que la máquina realice alguna tarea”.

Les indicamos que saquen una hoja y escriban un programa para que el robot se siente y se pare dos veces y, además, que cada vez que se siente aplauda una vez, y cada vez que se pare lo haga dos veces. Les damos unos minutos para que lo piensen y luego les pedimos que compartan en voz alta sus programas con el resto de la clase. A medida que los digan, iremos escribiendo en el pizarrón distintas propuestas. La solución esperada se muestra a continuación.



Un programa

Nuevamente, como robots, “ejecutaremos” cada propuesta diferente. En caso de que surjan programas que contienen errores –es decir, que al seguir las instrucciones no se consigue el objetivo buscado–, lo analizamos grupalmente para detectar el problema y corregirlo.

## Eventos

A continuación entregamos las copias de las cartas del anexo a los estudiantes. A la mitad de ellos les daremos las del primer juego de cartas y a la otra mitad, las del segundo. Les decimos: "Ahora ustedes serán los robots. Observen las cartas y prepárense para moverse y aplaudir. ¿Están listos?".

Cuando el docente dice "Uno"	Cuando el docente dice "Dos"	Cuando el docente dice "Tres"
<b>PARATE SENTATE</b>	<b>APLAUDÍ APLAUDÍ</b>	<b>PARATE APLAUDÍ SENTATE</b>
Cuando el docente dice "Uno"	Cuando el docente dice "Dos"	Cuando el docente dice "Tres"
<b>APLAUDÍ</b>	<b>PARATE SALTÁ SENTATE</b>	<b>PARATE APLAUDÍ SENTATE</b>

JUEGO 1

JUEGO 2

Cartas para los estudiantes

Sin dar mayores explicaciones, comenzamos a decir "Uno", "Dos" y "Tres" alternadamente, sin seguir ningún orden en particular. Cada vez que digamos "Uno", los estudiantes que tengan el primer juego de cartas deberán pararse y sentarse y, los que tengan el segundo, aplaudir una vez; cuando digamos "Dos", los primeros deberán aplaudir dos veces y los segundos pararse, saltar y sentarse; y al decir "Tres", todos tendrán que pararse, aplaudir y sentarse. Luego de divertirnos un rato, los dejamos descansar.

Proseguimos: "¿Qué diferencia encuentran entre lo que hicimos recién y lo que habíamos hecho antes?". Guiamos la discusión para llegar a la conclusión de que, a diferencia de lo realizado anteriormente, en este caso las instrucciones se usaron para describir la respuesta del robot frente a la ocurrencia de ciertos eventos. "Efectivamente, antes habíamos escrito programas que pudimos 'ejecutar' de punta a punta, sin esperar a que algo suceda para seguir sus instrucciones. En este caso, los programas son distintos: en cada uno se describe cómo debe responder el robot cuando alguien dice 'uno' o 'dos' o 'tres'".

Continuamos: “Es importante que distingamos dos cosas. Por un lado, están los **eventos**, que son sucesos que *a priori* no sabemos cuándo van a ocurrir –en la actividad, que alguien diga ‘uno’, ‘dos’ o ‘tres’–; por otro, el **manejo de los eventos**, que consiste en la reacción de la máquina frente a la ocurrencia de los eventos. Los programas de este tipo, en los que se programa la respuesta frente a determinados eventos, pertenecen a un paradigma de programación llamado **programación dirigida por eventos**”.

### CIERRE

Les preguntamos a los estudiantes si se les ocurren programas que respondan a este modelo. En caso de que no surjan respuestas, indagamos sobre las aplicaciones que usan en sus teléfono y concluimos que, en general, todos los programas interactivos –es decir, aquellos en los que existe interacción con el usuario– se inscriben dentro de este paradigma: presionamos un botón y se dispara la impresión de un documento; movemos el ratón y un puntero se desplaza por la pantalla, etc.

---

# CARTAS PARA MANEJO DE EVENTOS

Cuando el docente dice “Uno”

**PARATE  
SENTATE**

Cuando el docente dice “Dos”

**APLAUDÍ  
APLAUDÍ**

Cuando el docente dice “Tres”

**PARATE  
APLAUDÍ  
SENTATE**

Cuando el docente dice “Uno”

**APLAUDÍ**

Cuando el docente dice “Dos”

**PARATE  
SALTÁ  
SENTATE**

Cuando el docente dice “Tres”

**PARATE  
APLAUDÍ  
SENTATE**

ANEXO

## Actividad 2

### iHola, amigo!



DE A DOS

#### OBJETIVOS

- Presentar el entorno MIT App Inventor 2.
- Programar una aplicación sencilla.
- Manejar eventos.

#### MATERIALES

- Computadora
- Internet
- MIT App Inventor 2
- Ficha para estudiantes “Puesta a punto de App Inventor”
- Ficha para estudiantes “iHola, amigo!”
- Teléfono con Android (opcional)

#### DESARROLLO

A lo largo de esta actividad, los estudiantes usarán por primera vez MIT App Inventor 2 y empezarán a familiarizarse con su interfaz. Comenzarán creando sus cuentas de usuario, que utilizarán a lo largo de todos los proyectos del manual. Luego, harán una primera experiencia de programación en computadoras.

#### Preparación del entorno

##### ¿QUÉ ES MIT APP INVENTOR?

MIT App Inventor es un entorno de programación visual de código abierto y uso libre y gratuito que permite crear aplicaciones para teléfonos y tabletas que corran el sistema operativo Android. Se trata de una herramienta basada en bloques, lo que facilita la creación de aplicaciones para aquellos que no están familiarizados con el mundo de la programación.



Originalmente creado en los laboratorios de Google, App Inventor es actualmente mantenido por el Laboratorio de Ciencias de la Computación e Inteligencia Artificial (CSAIL) del Instituto de Tecnología de Massachusetts (MIT). Por tratarse de una aplicación web, para usarla es necesario contar con una conexión a Internet y un navegador.<sup>1</sup> Sin embargo, terceros han construido entornos que permiten el uso fuera de línea de la aplicación.<sup>2</sup>

Si bien en principio las aplicaciones creadas con App Inventor son para ejecutarse en dispositivos que corran el sistema operativo Android, también puede instalarse un emulador de teléfonos con Android provisto por los desarrolladores de App Inventor que permite verlas en funcionamiento directamente en la computadora. El emulador está disponible para Windows, GNU/Linux y Mac OS, y las instrucciones de instalación se encuentran en <http://bit.ly/2J6n0tW>. En cada sistema operativo el aspecto del



Emulador de teléfono con Android

<sup>1</sup> Actualmente, los navegadores compatibles con la aplicación son Google Chrome, Safari y Firefox.

<sup>2</sup> Algunas opciones disponibles para el uso del entorno fuera de línea se encuentran documentadas en <http://bit.ly/2ZwuznN>.

emulador es diferente, aunque todos muestran en la pantalla las aplicaciones que se construyen con App Inventor. Recomendamos instalar el emulador en todas las computadoras que se usen a lo largo del curso, antes de comenzar con la primera actividad en las aulas o laboratorios.

### Creación de una cuenta App Inventor

Repartimos a los estudiantes la ficha “Puesta a punto de App Inventor”–que contiene una guía paso a paso de cómo crear una cuenta de usuario– y los guiamos para que sigan las indicaciones.

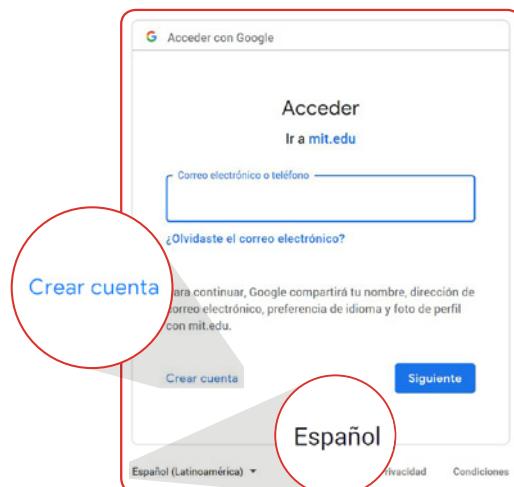
Les indicamos que ingresen al sitio web de App Inventor en <http://ai2.appinventor.mit.edu/>, y ayudamos a quienes tengan dificultades para llegar hasta ahí. Si al acceder se muestra la página en inglés, les sugerimos que seleccionen el idioma español en la lista desplegable que se encuentra en la parte inferior izquierda.

Las cuentas de usuario de App Inventor están asociadas a una dirección de correo electrónico de Gmail. En caso de que los estudiantes cuenten con una, podrán usarla directamente; en caso contrario, deberán crear una, lo cual pueden hacer haciendo clic sobre el link *Crear cuenta*.

Al iniciar sesión por primera vez, se le pide al usuario que acepte los términos y condiciones de uso de la aplicación.<sup>1</sup> A modo de resumen, al aceptarlos se consiente (*i*) que la dirección de correo electrónico proporcionada se rige por las mismas pautas de privacidad que cualquier otra cuenta de Gmail; (*ii*) que el MIT no accederá a información personal del usuario, a menos que explícitamente se dé consentimiento; (*iii*) que se autoriza a MIT a guardar las aplicaciones en sus servidores, pero que es el usuario (y no MIT) el único responsable de las aplicaciones y su uso; y (*iv*) que está prohibida la generación de contenido que difame, hostigue o amenace a terceros, que promueva actividades ilegales, que infrinja la propiedad intelectual de terceros, que difunda contenido inapropiado (deshonesto, pornográfico, etc.), que haga publicidad con fines comerciales y que tenga contenido relacionado con actividades políticas partidarias.

### Primera aplicación

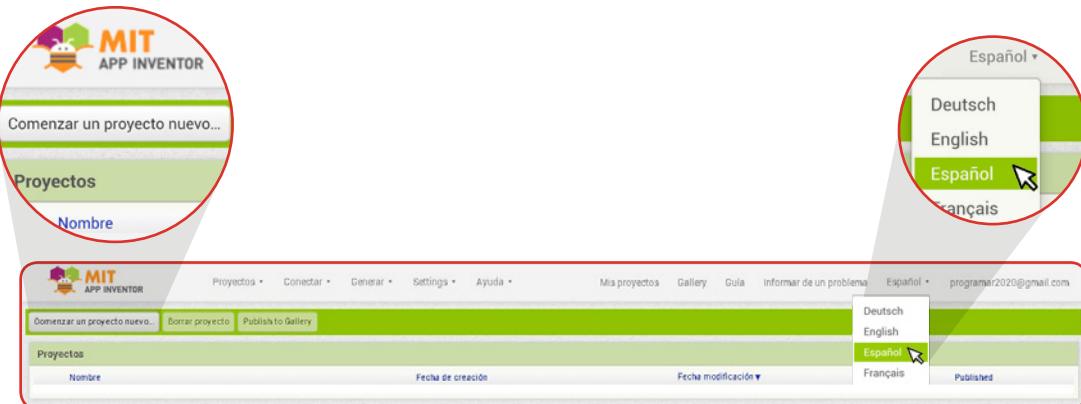
Una vez aceptados los términos y condiciones, encontrarán una pantalla que lista los proyectos creados por el usuario. De tratarse del primer ingreso, la lista estará vacía. Les sugerimos, nuevamente, que cambien el idioma a español.<sup>2</sup>



Pantalla de inicio de sesión

<sup>1</sup> Esta página no se encuentra disponible en español.

<sup>2</sup> No todo el entorno de App Inventor se encuentra disponible en español. Probablemente, en algunas ocasiones, los estudiantes encontrarán algunos textos en inglés.



Pantalla de inicio del entorno de App Inventor

Para crear un proyecto se debe hacer clic en el botón *Comenzar un proyecto nuevo...*, ubicado sobre la izquierda de la barra superior. Entonces, se nos solicitará que ingresemos el nombre del proyecto. En App Inventor, los nombres deben comenzar con una letra y solo pueden contener letras, números y guiones bajos (\_).

#### Crear un nuevo proyecto de App Inventor

Nombre del proyecto:

Creación de un nuevo proyecto

El entorno de App Inventor tiene dos editores principales: el editor de diseño y el editor de bloques. El primero permite incorporar los elementos que formarán parte de la aplicación (como botones, imágenes, campos de texto, etc.) y diseñar su aspecto; y el segundo, programar el comportamiento de la aplicación que estamos creando (por ejemplo, cómo reaccionará la aplicación cuando se presiona determinado botón). Al ingresar, nos encontraremos en el editor de diseño.<sup>1</sup>

The screenshot shows the App Inventor Design Editor. On the left, a red circle highlights the 'Componentes' tab under the 'Components' section. In the center, a red arrow points to the 'Visor' (Preview) window showing a smartphone screen. A red circle highlights the 'Bloques' tab under the 'Blocks' section on the right. Another red circle highlights the 'Propiedades' (Properties) panel on the far right, which shows the properties for 'Screen1'.

Listado de componentes incorporados a la aplicación

Componentes

Bloques

Propiedades

Visor del teléfono en el que se incorporan componentes

Al seleccionar un componente de la aplicación, muestra sus propiedades (tamaño, color, etc.)

Editor de diseño de App Inventor

<sup>1</sup>El aspecto del visor del teléfono varía al usar App Inventor en distintos sistemas operativos; por ejemplo, al abrir el entorno en una computadora con Mac OS, no se observa la línea verde que se ve sobre el lateral derecho de la pantalla.

Cuando todos los estudiantes se encuentren en el editor de diseño, les damos un tiempo y los motivamos para explorar la herramienta. Hacemos una puesta en común explicando de qué se trata cada sección del entorno. Podemos ir anotando en el pizarrón lo que vayan descubriendo los estudiantes. Es importante hacer hincapié en los siguientes elementos:

**Paleta de componentes.** En este sector aparecen todos los componentes que se pueden incorporar a las aplicaciones. Algunos son visibles y se verán en la pantalla –como los botones, campos de texto, etc.– y otros no –como los sensores, el micrófono del teléfono, etc.–.

**Visor del teléfono.** Simula la pantalla de un teléfono. Allí se arrastran los componentes que formarán parte de la aplicación y, a medida que se agregan, se los muestra en la pantalla.

**Componentes.** Lista de componentes que forman parte de la aplicación. Inicialmente solo se encuentra el que corresponde a la pantalla. A medida que se agregan otros, irán apareciendo allí.

**Botones Diseño y Bloques.** Permiten cambiar del editor de diseño al de bloques, y viceversa.

**Propiedades.** Al seleccionarse un componente, allí se mostrarán sus propiedades.<sup>1</sup>

A continuación, les repartimos a los estudiantes la ficha “¡Hola, amigo!” y les proponemos que trabajen sobre las primeras cuatro consignas, que piden programar una aplicación que tenga un botón cuyo texto sea *Saludar* de modo que, al presionarlo, el teléfono diga “¡Hola, amigo!”. Sin darles indicaciones dejamos que, para conseguirlo, exploren el entorno autónomamente.

Para incorporar un componente a la aplicación solo hay que arrastrarlo desde el menú de componentes –bajo el título *Paleta*– hasta el visor. Los botones se encuentran en *Paleta > Interfaz de usuario* y, por tratarse de un componente visible –es decir, que aparecerá en la pantalla al correr la aplicación–, al depositarlo lo veremos aparecer en el visor del teléfono.

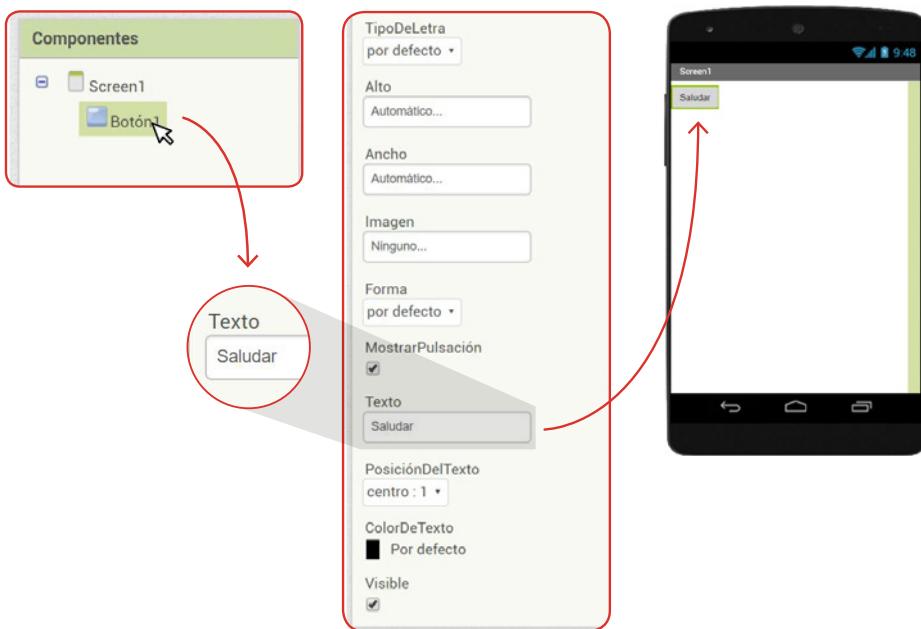


Incorporación de un botón a la aplicación

<sup>1</sup> El conjunto de propiedades varía entre distintos tipos componentes (botones, campos de texto, etc.). Al seleccionar uno, la barra de propiedades muestra las que son propias del tipo de componente seleccionado.

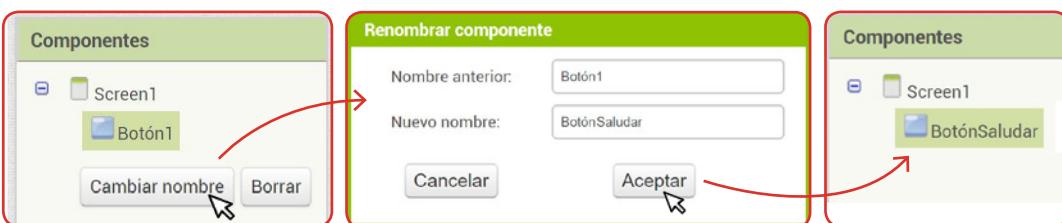
Además, en el listado *Componentes*, notarán que el botón aparecerá debajo de *Screen1* (Pantalla1, en inglés).<sup>1</sup>

Para cambiar la leyenda que aparece en el botón, en primer lugar hay que seleccionarlo. Esto puede hacerse tanto en el visor como en el sector *Componentes*. Al hacerlo, en el panel *Propiedades* aparecerán aquellas propiedades que son propias de los botones, como por ejemplo su altura, su ancho, su forma, el texto que muestra, etc. Allí puede cambiarse la leyenda del botón modificando el texto que se encuentra bajo el título *Texto*. Al hacerlo, en el visor verán que el botón se habrá actualizado.



Cambio de la leyenda del botón

Si prestamos atención, en el panel *Componentes* notaremos que el nombre del botón es *Botón1*. Si bien en este caso se trata del único componente que forma parte de la aplicación, en sucesivos proyectos se crearán aplicaciones con varios componentes distintos. Para poder reconocerlos con facilidad, una buena práctica es cambiar el nombre que App Inventor les asigna por defecto, por otro que nos resulte claro para identificarlos. Para hacerlo, hay que seleccionarlo en el menú y presionar el botón *Cambiar nombre*. Una buena opción, en este caso, es llamarlo *BotónSaludar*.



Cambio de nombre del botón

<sup>1</sup> *Screen1* es el componente que representa a la pantalla de la aplicación. Todos los componentes que se agreguen (botones, campos de texto, etiquetas, etc.) aparecerán, entonces, bajo este. Si bien App Inventor permite que en una aplicación haya más de una pantalla, en los proyectos que se presentan en este manual se usa solo una: *Screen1*.

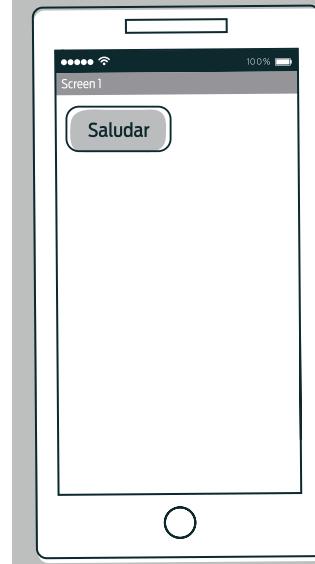
Una vez que los estudiantes consigan disponer el botón en la pantalla, los invitamos a que ejecuten la aplicación en el emulador. Para hacerlo, deben seleccionar la opción *Emulador* del menú *Conectar*, que se encuentra en la barra superior del entorno. El emulador tarda unos instantes en arrancar, pero finalmente mostrará en la pantalla la aplicación.



App Inventor se conecta al emulador

Cuando los estudiantes consigan correr el emulador en la pantalla de la computadora, les proponemos que presionen el botón. Verán, entonces, que no sucede nada. Preguntamos: “¿Hemos finalizado la aplicación? ¿Por qué no pasa nada cuando apretamos el botón?”. Escuchamos atentamente sus respuestas y continuamos: “Hasta aquí, lo único que hemos hecho es diseñar el aspecto de la aplicación, pero en ningún momento hemos indicado cómo tiene que responder cuando hacemos clic sobre *Saludar*, ¿no es cierto? Que un botón tenga la leyenda *Saludar* no significa nada para un teléfono; los dispositivos no interpretan el castellano. Todavía resta programar el comportamiento que esperamos que tenga”. Los invitamos a que indaguen el entorno en busca de cómo hacerlo.

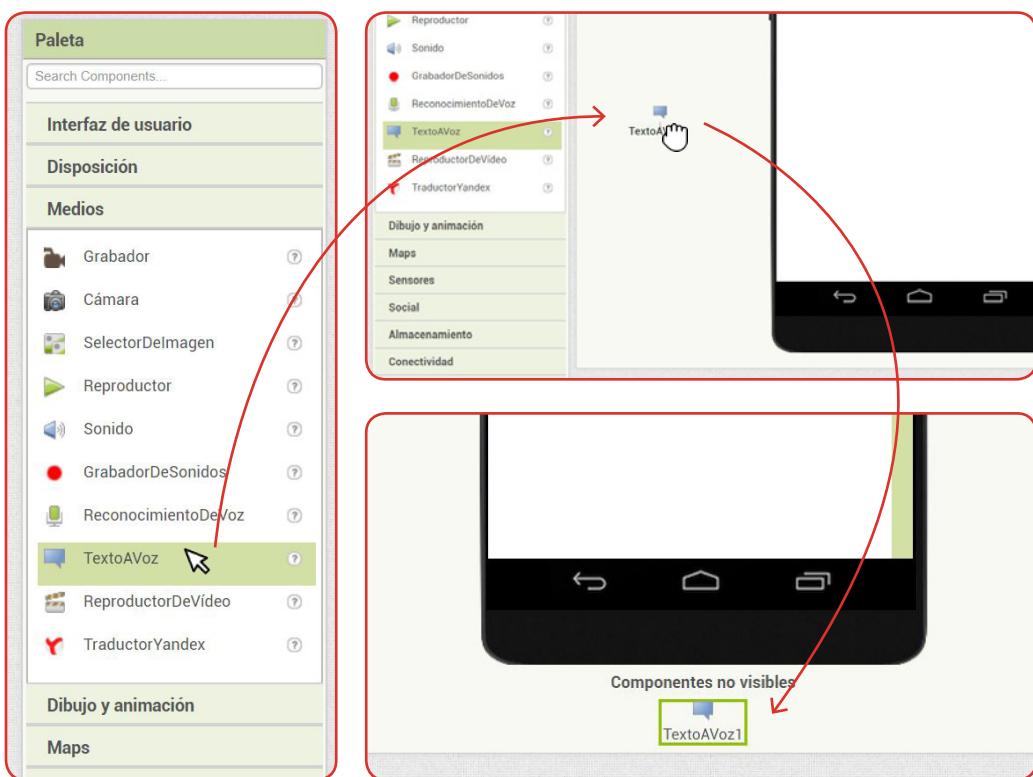
Emulador de Android-App Inventor



Ejecución del programa en el emulador

App Inventor da la posibilidad de agregar componentes que no serán visibles en la pantalla. Por ejemplo, se puede incorporar un giroscopio –que sirve para detectar las rotaciones del teléfono y, por lo tanto, nos permite establecer qué hace nuestra aplicación en ese caso– o una cámara, que nos habilita a utilizar la cámara de fotos del dispositivo para tomar fotos. En esta actividad, para conseguir el objetivo planteado, hay que agregar un conversor de texto a voz. Este componente se encuentra disponible

en Paleta > Medios. Una vez que lo arrastramos hasta el visor, no lo veremos aparecer en la pantalla; en el entorno, los componentes no visibles se muestran bajo la pantalla del teléfono.



Incorporación del componente no visible que convierte texto a voz

Al igual que antes, recomendamos cambiar el nombre del componente. Como en este caso no habrá otro conversor de texto a voz, podemos simplemente llamarlo *ConversorTextoAVoz*.

Para establecer la lógica del programa –es decir, su comportamiento– hay que ir al editor de bloques, presionando el botón *Bloques* que se encuentra en el extremo derecho del menú superior del entorno.

Diseñador

Bloques



Botones para cambiar de un editor al otro



Editor de bloques de App Inventor

Una vez que los estudiantes se encuentren en el editor de bloques, nos aseguramos de que todos comprendan los elementos que se visualizan en la pantalla. De ser necesario, los repasamos grupalmente:

**Bloques > Integrados.** Menú para acceder a los bloques disponibles para cualquier aplicación que se programe en App Inventor.

**Bloques > Screen1.** Menú para acceder a los bloques asociados a los componentes que forman parte de la aplicación que se está creando.

**Visor.** Espacio en el que se arrastran bloques para construir programas.

El objetivo del desafío propuesto es que, al presionar el botón *Saludar*, el teléfono diga “¡Hola, amigo!”. Hay que diferenciar, en este punto, dos elementos importantes en programación: los **eventos** y el **manejo de eventos**. Los eventos son sucesos que acontecen y que, *a priori*, no sabemos cuándo ocurrirán –por ejemplo, en el caso de esta actividad, que el usuario presione el botón–. Por su parte, el manejo de eventos es el conjunto de acciones que se dispara cuando ocurre el evento –en este caso, que el teléfono diga “¡Hola, amigo!”–. Se trata, en definitiva, de la parte del programa que se ejecuta cuando ocurre el evento.

Al hacer clic sobre *Bloques > Screen1 > BotónSaludar*, se desplegarán varios bloques relacionados con el botón en cuestión. Los primeros que aparecen, de color ocre, son para realizar el manejo de distintos eventos relacionados con el botón: cuando se haga clic sobre él, cuando obtenga foco, cuando se lo suelte, etc. En este desafío, nos interesa programar el comportamiento de la aplicación cuando se haga clic sobre el componente, por lo que arrastramos el bloque **cuando BotónSaludar.Clic / ejecutar { }** al espacio del programa.

Resta aún definir el manejo asociado al evento: qué acciones realizará el programa cuando se presione *BotónSaludar*. Hay muchos componentes de App Inventor a los que se les puede indicar que lleven a cabo distintas acciones. Estas acciones, a las que habitualmente se las designa como **comandos**, están representadas en el entorno con bloques de color púrpura.

Al hacer clic sobre *Bloques > Screen1 > ConversorTextoAVoz*, aparecerán distintos bloques relacionados con el conversor de texto a voz que podemos utilizar en el programa. En este caso usaremos el bloque **llamar ConversorTextoAVoz.Hablar / mensaje [ ]**, que nos permitirá que el teléfono “hable”.

Finalmente, para incorporar el mensaje, en *Bloques > Integrados > Texto*, podemos seleccionar el bloque **“[ ]”**, que sirve para definir textos. Luego de incorporarlo al espacio del programa, escribimos entre las comillas “¡Hola, amigo!”. Encastrando los bloques adecuadamente, habremos completado la aplicación. Ahora sí, en el emulador, funcionará de acuerdo a lo esperado.

```
cuando BotónSaludar.Clic
ejecutar llamar ConversorTextoAVoz.Hablar
mensaje "¡Hola, amigo!"
```

Programa para que el teléfono salude



Bloques para manejar eventos sobre botones

```
llamar ConversorTextoAVoz.Hablar
mensaje
```

Bloque para que el teléfono “hable”



### NO ES NECESARIO REINICIAR EL EMULADOR

Al hacer cambios en nuestras aplicaciones, el emulador los reconoce automáticamente. Por lo tanto, para probar las modificaciones realizadas, no hace falta detenerlo y volver a lanzarlo.

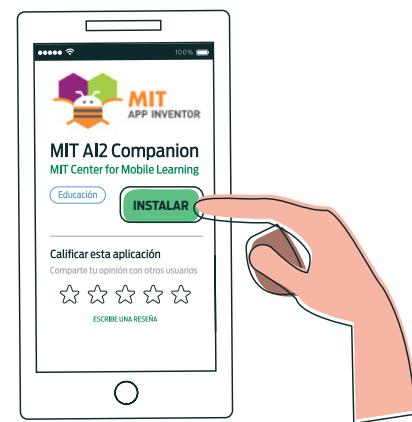


### Ejecución en el teléfono

Para ejecutar programas creados en APP Inventor en un teléfono o tableta que corra el sistema operativo Android, hay que tener instalada en el dispositivo la aplicación MIT AI2 Companion. Al igual que con cualquier otra aplicación, esta puede instalarse entrando en la tienda de aplicaciones Google Play y haciendo clic sobre **INSTALAR**.

Les proponemos a los estudiantes, entonces, que corran la aplicación en sus teléfonos. Para hacerlo deberán, en primer lugar, reiniciar la conexión para no continuar ejecutándola en el emulador.

A continuación, en el mismo menú, tendrán que seleccionar la opción **AI Companion**. Entonces, verán aparecer dos códigos, uno QR y otro alfanumérico de 6 caracteres. Luego, al abrir MIT AI2 Companion en el teléfono, podrán iniciar la aplicación que saluda usando cualquiera de los dos códigos.



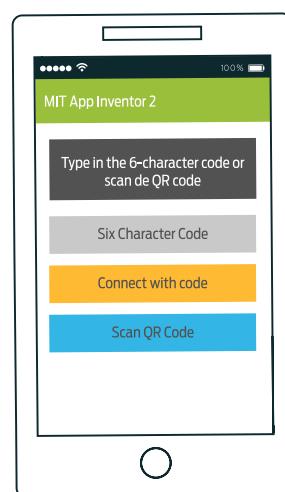
Instalación de MIT AI2 Companion



Reinicio de conexión



Carga de la aplicación en el teléfono



## Más funcionalidades

Luego de que los estudiantes hayan conseguido ejecutar la aplicación en sus teléfonos, los invitamos a que completen las dos últimas consignas. Al hacerlo, irán descubriendo distintos componentes y posibilidades que da App Inventor para crear aplicaciones.

La primera pide que, cuando lo agitamos, el teléfono diga “¡No te muevas así!”. Para conseguirlo, en el editor de diseño deben agregar un acelerómetro (disponible en Paleta > Sensores), que permite detectar cuando el teléfono se agita y, luego, en el editor de bloques la respuesta esperada por el programa cuando se zarandeá el dispositivo. A la derecha se observan los bloques incorporados al programa.



Porción del programa para cuando se sacude el teléfono

La última consigna invita a los estudiantes a investigar App Inventor para agregarle nuevas funcionalidades a la aplicación. Algunas posibilidades son: usar el componente Cámara de Paleta > Medios para tomar fotografías, agregar una imagen y mostrarla en la pantalla o usar el componente de reconocimiento de voz, entre otras. Por último, invitamos a que muestren sus producciones y expliquen cómo las pensaron y las programaron.

## CIERRE

Les comentamos a los estudiantes que hay una amplia comunidad de desarrolladores de aplicaciones App Inventor a lo largo y ancho del planeta. Los miembros suelen publicar sus producciones, que se pueden descargar, copiar y modificar. Los alentamos a que exploren distintos proyectos publicados por miembros de la comunidad.

Galería de proyectos publicados

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# PUESTA A PUNTO DE APP INVENTOR

¡Vamos a aprender a usar App Inventor! MIT App Inventor 2 es un entorno de programación en el que podés desarrollar tus aplicaciones para teléfonos y tabletas. Para eso, vas a necesitar abrir una cuenta en el sitio en línea de App Inventor, que te va a permitir crear, guardar y compartir tus proyectos.

¿Ya estás frente a una compu con Internet? Seguí estas instrucciones:

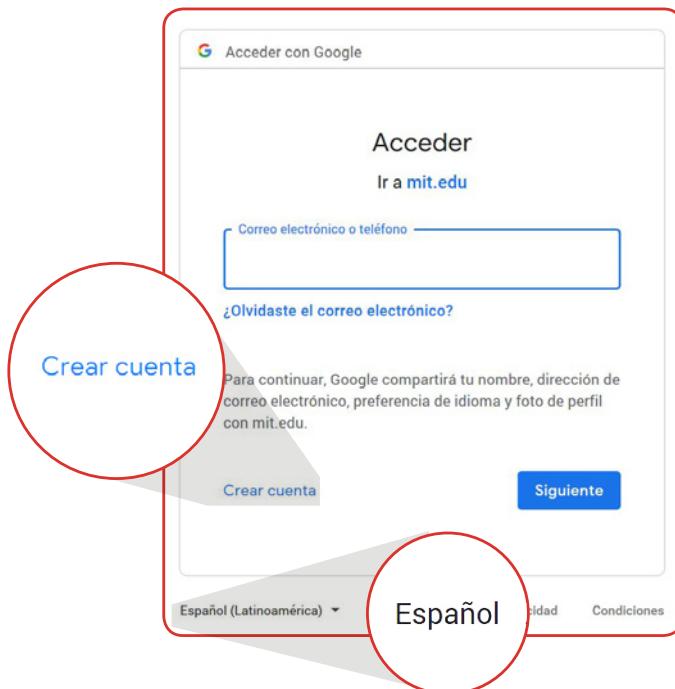
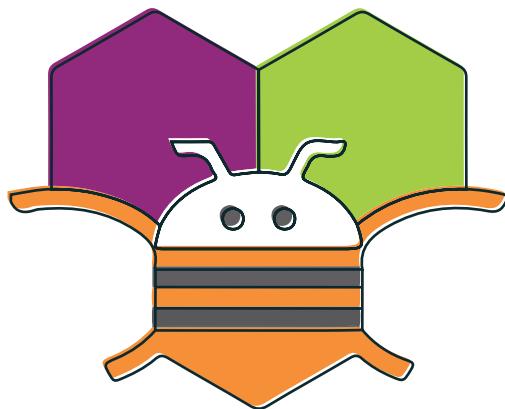
## CREAMOS UNA CUENTA

1. Abrí un navegador web y cargá la dirección de MIT App Inventor 2:  
<http://ai2.appinventor.mit.edu>.

2. ¿La página está en inglés? Abajo a la izquierda hay un menú en el que podés seleccionar el idioma español.

3. Las cuentas de usuario de App Inventor están asociadas a una dirección de correo electrónico de Gmail. Si ya tenés una, podés usarla directamente. Si no, podés crear una haciendo clic en *Crear cuenta*.

4. Cuando inicies sesión por primera vez, vas a ver los términos y condiciones de uso de la aplicación. Al aceptarlos, te comprometés a hacerte cargo de tus producciones, a no generar contenidos inapropiados, a no infringir normas legales y a no hacer publicidad.



NOMBRE Y APELLIDO:

CURSO:

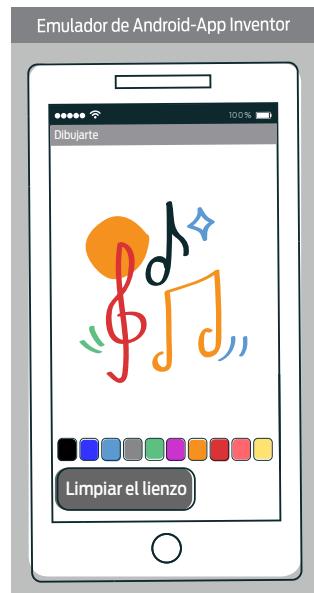
FECHA:

**5.** Listo, ¡ya estás adentro! Lo que ves es la lista de proyectos creados con tu usuario. Como todavía no hiciste ninguno, la lista está vacía. Si la página está en inglés, podés cambiarla a español.



## INSTALAMOS EL EMULADOR

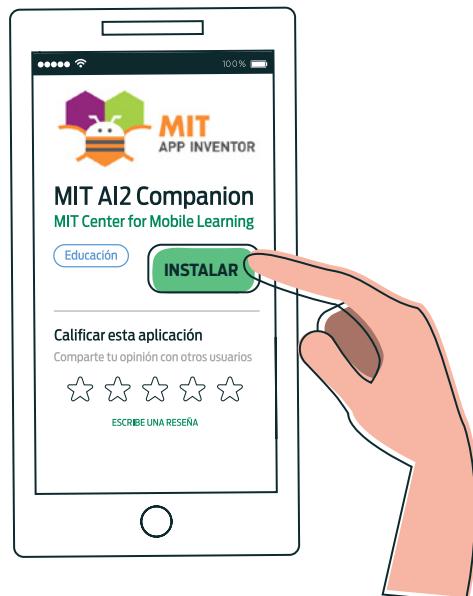
Si bien en principio las aplicaciones creadas con App Inventor son para ejecutarse en dispositivos que corran el sistema operativo Android, también puede instalarse un emulador de teléfonos con Android provisto por los desarrolladores de App Inventor, que permite verlas en funcionamiento directamente en la computadora. El emulador está disponible para Windows, GNU/Linux y Mac OS, y las instrucciones de instalación se encuentran en <http://bit.ly/2J6n0tW>.



## INSTALAMOS LA APLICACIÓN PARA EJECUTAR EN TELÉFONOS

Para ejecutar programas creados en APP Inventor en un teléfono o tableta que corra el sistema operativo Android hay que tener instalado en el dispositivo la aplicación MIT AI2 Companion. Al igual que con cualquier otra aplicación, esto puede hacerse entrando en la tienda de aplicaciones Google Play y haciendo clic sobre INSTALAR.

¡Ahora ya tenés todo lo que hace falta para crear tus propias aplicaciones!



NOMBRE Y APELLIDO:

CURSO:

FECHA:

# ¡HOLA, AMIGO!

Saludar es, ciertamente, de buena educación. Usando App Inventor, vas a ocuparte de que tu teléfono sea muy pero muy cortés.

1. Creá un proyecto con tu usuario. Al ingresar, vas a encontrar una pantalla como esta.



The screenshot shows the MIT App Inventor interface with several callouts highlighting different parts:

- Paleta de componentes que se pueden incorporar a la aplicación**: Points to the "Components" palette on the left, which lists various UI components like Button, Checklist, SelectorDeFecha, Image, etc.
- Listado de componentes incorporados a la aplicación**: Points to the "Components" list on the right, showing "Screen1" and other components.
- Botón para acceder al editor de bloques**: Points to the "Blocks" tab at the top right.
- Propiedades**: Points to the "Properties" panel on the right, which shows properties for "Screen1" such as "PantallaAcercaDe".
- Visor del teléfono en el que se incorporan componentes**: Points to the "Phone View" in the center, showing a smartphone icon with a white screen where components can be placed.

2. Agregá un botón al teléfono y ocupate de que la leyenda diga "Saludar". Despues, ejecutá la aplicación en el emulador y hacé clic sobre el botón. ¿Ya saluda el teléfono? ¿Por qué?

NOMBRE Y APELLIDO:

CURSO:

FECHA:

### PARA VER LA APLICACIÓN EN LA COMPU

Para ejecutar la aplicación en la computadora tenés que ir al menú *Conectar* y seleccionar la opción *Emulador*.



### ¡CUIDADO CON LOS NOMBRES!

Hasta ahora agregaste solo un botón al que App Inventor le puso por defecto el nombre *Botón1*. Sin embargo, generalmente las aplicaciones van a tener muchos componentes y va a resultar indispensable poder reconocerlos por su nombre. En este caso podríamos llamarlo *BotónSaludar*. ¡Cambio!



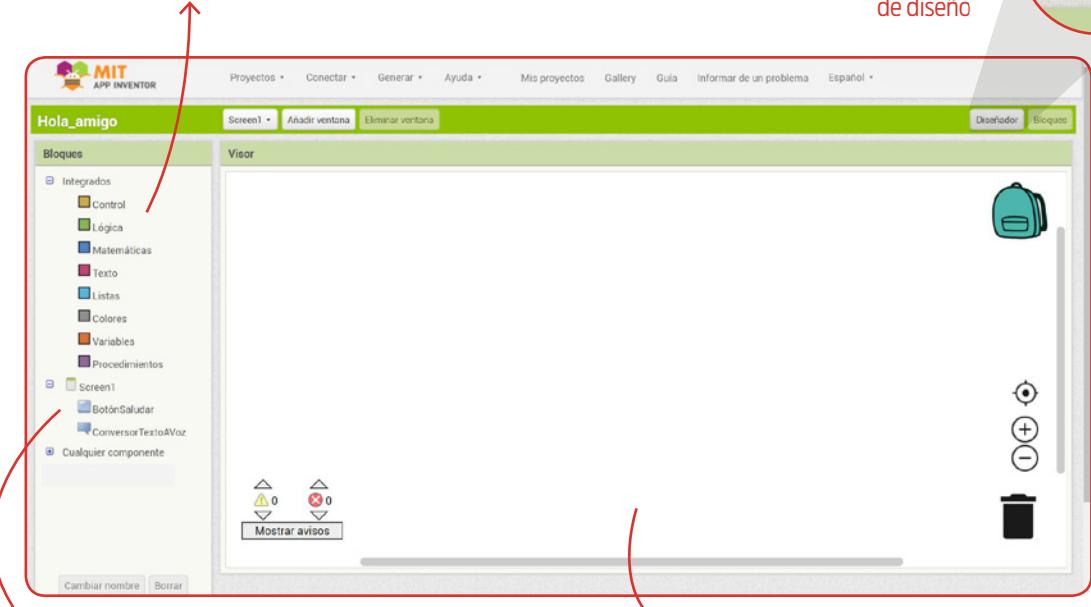
3. Para hacer que tu programa hable, tenés que agregar un componente más a la aplicación. Insepccióná los que están disponibles en *Paleta* y arrastrá el que te sirva sobre el visor del teléfono. ¿Cuál agregaste? ¿Lo ves en el visor? ¿Por qué?

4. Ingresá al editor de bloques. Vas a encontrarte con algo así.

Bloques disponible para todas las aplicaciones

Botón para volver al editor de diseño

Diseñador



Bloques específicos de los componentes de la aplicación

Espacio en el que se arman los programas arrastrando bloques

NOMBRE Y APELLIDO:

CURSO:

FECHA:

Usá los bloques que hagan falta para que la aplicación hable.



### EVENTOS Y MANEJO DE EVENTOS

Hay dos elementos importantes que tenés que diferenciar: los **eventos** y el **manejo de eventos**.

Los eventos son sucesos que acontecen y que, *a priori*, no sabemos cuándo ocurrirán –por ejemplo, que el usuario presione el botón–. Por su parte, el manejo de eventos es el conjunto de acciones que se disparará cuando ocurra el evento –en este caso, que el teléfono diga “*Hola, amigo!*”–. Se trata, en definitiva, de la parte del programa que se ejecuta cuando ocurre un evento.

5. Ahora le tenés que agregar una nueva funcionalidad a la aplicación. Cuando alguien agite el teléfono, el programa tiene que decir “*¡No te muevas así!*”. Investigá los sensores que podés usar para reconocer el zarandeo. ¿Cómo lo resolviste?
- 
- 
- 

### PARA EJECUTAR LA APLICACIÓN EN EL TELÉFONO

Para ejecutar la aplicación en teléfono, en primer lugar tenés que reiniciar la conexión para no continuar ejecutándola en el emulador. A continuación, en el menú Conectar seleccioná la opción AI Companion. Entonces, verás aparecer dos códigos, uno QR y otro alfanumérico de 6 caracteres. Al abrir MIT AI2 Companion en el teléfono, podrás cargar la aplicación usando cualquiera de los dos.

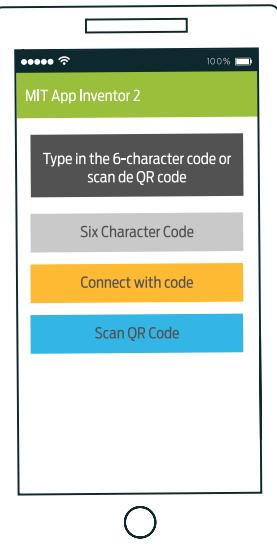
#### Conectado a Companion

Launch the MIT AI2 Companion on your device and then scan the barcode or type in the code to connect for live testing of your app.  
[Need help finding the Companion App?](#)



Tu código es:  
qsphft

Cancelar





## Secuencia Didáctica 2

# INTERFAZ GRÁFICA, COMPORTAMIENTO Y PUBLICACIÓN DE APLICACIONES

El diseño de la **interfaz gráfica** es una parte importante en el desarrollo de aplicaciones en las que existe una interacción persona-computadora. La cara visible de un programa resulta fundamental, pues actúa como intermediaria entre los usuarios de un programa y las funcionalidades que este provee.

En esta secuencia didáctica se proponen dos actividades para que los estudiantes diseñen la interfaz y programen el **comportamiento** de una aplicación para dibujar: *Dibujarte*. Además, se brinda una guía para que puedan compartir sus producciones con amigos, parientes y público en general.

### ..... **OBJETIVOS**

- Presentar la importancia de las interfaces gráficas.
  - Separar el diseño de la interfaz de una aplicación de la programación de su comportamiento.
  - Publicar aplicaciones.
- .....

# Actividad 1

## La interfaz de Dibujarte



### OBJETIVOS

- Reconocer la importancia de las interfaces gráficas.
- Familiarizarse con herramientas para diseñar interfaces.

### MATERIALES

- Computadora
- Internet
- MIT App Inventor 2
- Ficha para estudiantes
- Teléfono con Android (opcional)

### DESARROLLO

El diseño de la composición visual de una aplicación (habitualmente nombrada como GUI, por las siglas en inglés de *graphical user interface*) es una parte importante en el desarrollo de aplicaciones en las que existe una interacción persona-computadora. Mientras que un buen diseño contribuye a la facilidad con la que un usuario puede usar una aplicación, uno malo puede atentar contra la experiencia de uso, al generar desagrado e incluso rechazo en quien utiliza el programa.

El objetivo de esta actividad es que los estudiantes adviertan la importancia que tienen las interfaces gráficas en los programas que, al funcionar, interactúan con uno o más usuarios. Al finalizar, sabrán cómo distribuir en la pantalla los componentes que forman parte de una aplicación App Inventor y desarrollarán ideas sobre cómo presentar un programa a un usuario final.

### Introducción

Comenzamos pidiéndoles a los estudiantes que, usando un navegador, ingresen al sitio <https://goosh.org/>. De hacer falta, copiamos la dirección en el pizarrón. Luego de unos instantes les preguntamos: “¿Qué es lo que están viendo?”. En caso de que nadie lo haya notado, les contamos que se trata de una página que permite hacer búsquedas en Internet, pero que, a diferencia de las que usan habitualmente, la interfaz se asemeja a la que tenían los programas antes del nacimiento y desarrollo de las interfaces de usuario gráficas. “Así eran las cosas antes, ¿sabían?”.

Les indicamos entonces que realicen una búsqueda de algo que les interese, como su grupo de música favorito o el club del cual son hinchas, por ejemplo. Para hacerlo, alcanza con tippear lo que se quiere buscar y luego presionar *Enter*.

```
ooo
← → [Search Bar]

guest@goosh.org:/web> argentinos juniors
1) Argentinos Juniors - AAAJ
   Las inferiores de Argentinos se midieron con su par de Avellaneda y, en una jornada difícil, aunque los chicos tuvieron... 1598; 15/07/2019. Fútbol Amateur.
   https://www.argentinosjuniors.com.ar/

2) Argentinos Juniors - Wikipedia
   Asociación Atlética Argentinos Juniors is an Argentine sports club based in La Paternal, Buenos Aires. The club is mostly known for its football team, which ...
   https://en.wikipedia.org/wiki/Argentinos\_Juniors

3) Argentina - Argentinos Juniors - Results, fixtures, squad, statistics ...
   Argentinos Juniors. Official website. Founded: 1904; Address: Punta Arenas 1271, La Paternal 1427. Capital Federal, Ciudad de Buenos Aires; Country ...
   https://us.soccerway.com/teams/argentina/argentinos-juniors/113/

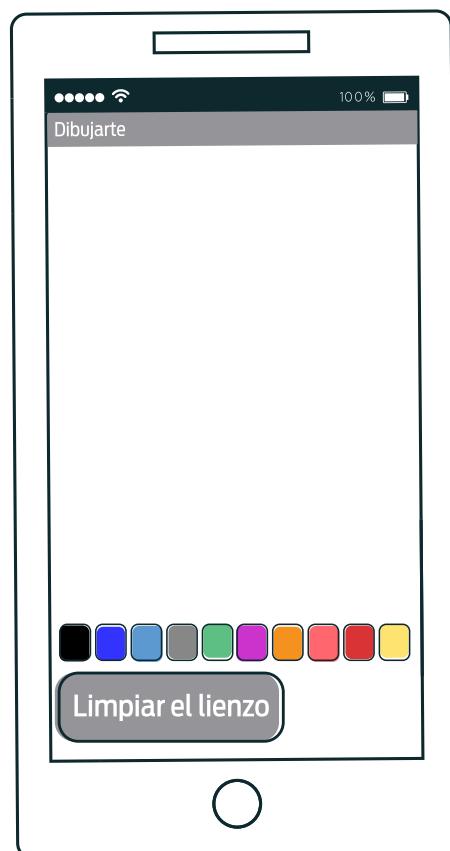
4) Asociación Atlética Argentinos Juniors - Club Profile | Transfermarkt
   Squad - Asociación Atlética Argentinos Juniors. The club's landing page - find all relevant information like the actual squad, related news, recent
```

Resultado de buscar “Argentinos Juniors”

Luego preguntamos: “¿Qué les parece esta página para buscar información? ¿Les gusta? ¿La van a volver a usar?”. Es probable que digan que no, y que surjan palabras como *incómoda*, *fea* o *rara*. Les comentamos: “Ya desde hace años que, en el campo del desarrollo de aplicaciones, el diseño de interfaces se ha transformado en algo muy importante. Una presentación simple, armónica y estética seduce fácilmente a los usuarios, mientras que una difícil de usar o poco vistosa puede generar rechazo y hacer que la aplicación termine en desuso”.

### Diseño de la interfaz de *Dibujarte*<sup>1</sup>

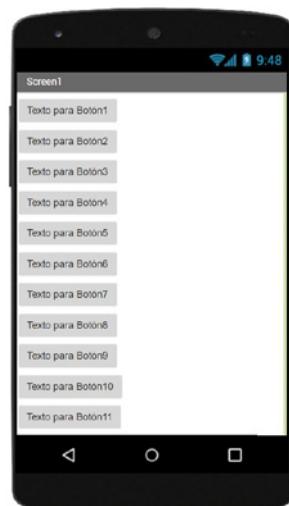
Les contamos que vamos a programar una aplicación para teléfonos inteligentes que permite dibujar con los dedos usando diversos colores. A continuación, les repartimos la ficha y les pedimos que observen la composición gráfica que allí se muestra. Les comentamos: “Lo que observan es la interfaz gráfica que tendrá la aplicación *Dibujarte*. Al correrla, con los dedos podremos seleccionar colores y pintar sobre la región blanca de la pantalla. Además, al presionar el botón *Limpiar el lienzo*, se deberá borrar lo dibujado, volviendo a dejar en blanco el sector de dibujo. ¿Qué tipos de componentes pueden identificar?”. Es esperable que alguien reconozca que *Limpiar el lienzo* se trata de un botón, pero que no tengan certeza de qué son los cuadraditos de colores. Indagamos: “¿Qué haremos con los cuadraditos de colores? ¿Cómo se usarán en la aplicación?”. Escuchamos con atención sus respuestas y continuamos: “Efectivamente, los usaremos para seleccionar el color con el que pintaremos la pantalla. Por ejemplo, si estamos pintando en rojo y quisieramos pasar al azul, tendremos que presionar sobre el cuadradito azul. Entonces, ¿de qué tipo de componente les parece que se trata?”. Guiamos la discusión de forma tal de concluir que, si bien no presentan su aspecto habitual, los cuadraditos de colores son botones.



Interfaz gráfica de *Dibujarte*

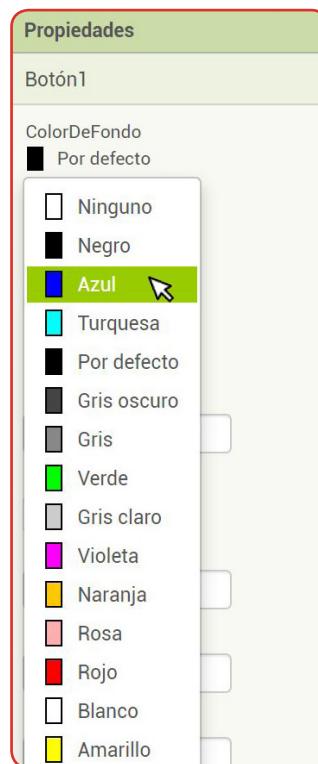
<sup>1</sup> Puede accederse a una versión completa de *Dibujarte* buscando “programar2020” en la galería de aplicaciones de App Inventor.

A continuación les proponemos que resuelvan la primera consigna de la ficha. Allí se pide que creen un nuevo proyecto de App Inventor y que agreguen todos los botones que forman parte de *Dibujarte*. Son 11 botones en total: 10 para seleccionar colores y 1 para limpiar la pantalla. Al agregarlos, los verán aparecer uno debajo del otro. Además, si observan con atención, notarán que los nombres de los componentes son *Botón1*, *Botón2*, *Botón3*, etc.



Los 11 botones de la aplicación en el visor

Comentamos: "Como ven, aún estamos lejos de conseguir que la interfaz tenga el aspecto que buscamos. A medida que agregamos componentes, App Inventor los ubica uno debajo del otro. Aquí hay varios problemas por solucionar. En primer lugar, vamos a ocuparnos del texto y los colores de los botones.<sup>1</sup> Selecciónen cada uno y, modificando sus propiedades, consigan que los diez primeros tengan el color que pretendemos y no muestren ningún texto. Además, encárguense de que el undécimo tenga la leyenda *Limpiar el lienzo*". También les recordamos la importancia de escoger nombres que nos permitan identificar a cada componente y los instamos a que los cambien. Podrían pasar a llamarse, por ejemplo, *BotónNegro*, *BotónAzul*, *BotónTurquesa*, y así siguiendo hasta *BotónLimpiarLienzo*.



Cambio de color de fondo de un botón

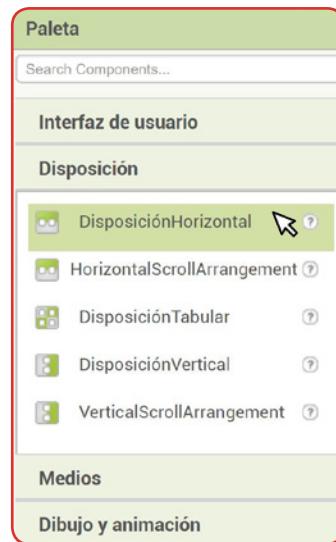
<sup>1</sup> Entre los colores que ofrece por defecto App Inventor al desplegar el menú del panel de propiedades se encuentran: negro, azul, turquesa, gris oscuro, gris, verde, gris claro, violeta, naranja, rosa, rojo, blanco y amarillo. En esta actividad usaremos todos ellos menos gris oscuro, gris claro y blanco.

Una vez que todos hayan cambiado el color y el texto de los botones –en el caso de aquellos para seleccionar colores, dejándolo vacío–, continuamos: “Esto ya se ve mejor, vamos avanzando. Sin embargo, todavía no conseguimos que la aplicación luzca como queremos: los botones siguen apareciendo uno debajo del otro”. Invitamos a los estudiantes a que exploren el entorno en busca de componentes que nos permitan ubicar los botones de otro modo.



Visor, luego de modificar las propiedades de los botones

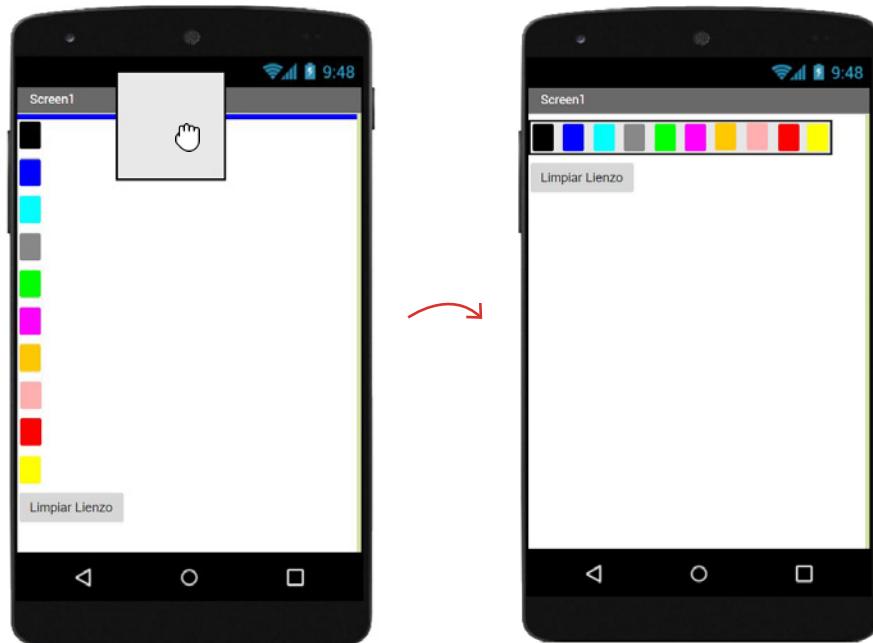
En *Paleta > Disposición* se presenta una serie de opciones provista por App Inventor que permiten ubicar y distribuir componentes visibles en la pantalla. Hay tres que se distinguen: *DisposiciónHorizontal* –para ubicar componentes uno a continuación del otro de izquierda a derecha–; *DisposiciónVertical* –para ubicarlos a continuación, de arriba hacia abajo–; y *DisposiciónTabular* –para ubicarlos en una tabla de  $n$  filas por  $m$  columnas, donde  $n$  y  $m$  son valores que definimos nosotros.<sup>1</sup>



Opciones para distribuir componentes visibles en la pantalla de la aplicación

Como cualquier otro componente, para incorporarlo a la aplicación hay que arrastrarlo desde la paleta hasta el visor. En este caso usaremos un componente *DisposiciónHorizontal*, para colocar los botones de colores uno al lado del otro. Una vez incorporado, aparecerá también en el panel *Componentes*. Luego, hay que arrastrar cada uno de los botones de colores dentro. Entonces, los veremos aparecer dispuestos horizontalmente, uno a continuación del otro. Recordamos la conveniencia de renombrar los componentes; en este ejemplo, llamamos *BotoneraDeColores* al panel de distribución horizontal.

<sup>1</sup> Las alternativas *HorizontalScrollArrangement* y *VerticalScrollArrangement* son parecidas a *DisposiciónHorizontal* y *DisposiciónVertical*, con la diferencia de que incorporan una barra de scrolling.



Incorporación de *DisposiciónHorizontal*

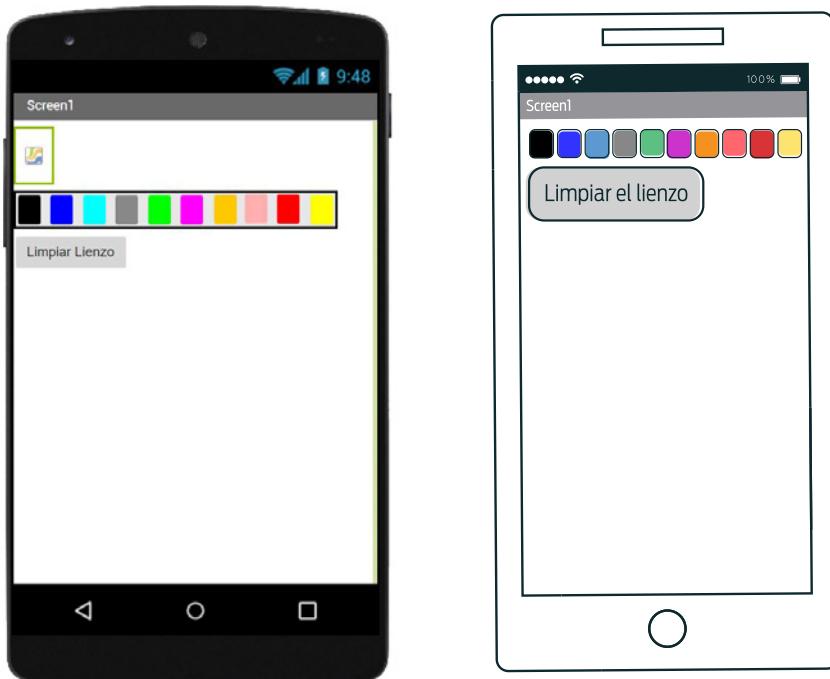
En el panel *Componentes*, los componentes se muestran organizados jerárquicamente, en forma arbórea.<sup>1</sup> Al incorporar los botones de colores al panel de distribución horizontal, este cambio se refleja en *Componentes*. Antes de moverlos, todos se encontraban en “el mismo nivel”, justo debajo de *Screen1*; luego de hacerlo, en “el primer nivel” –directamente bajo *Screen1*– solo quedan *BotoneraDeColores* y *BotónLimpiarLienzo* (que en el visor aparecen uno debajo del otro, como todo lo que se incorpora a *Screen1*); y, debajo de *BotoneraDeColores*, todos los botones de colores.



Organización jerárquica de los componentes de la aplicación

<sup>1</sup> Esta presentación es similar a la que utilizan los exploradores de archivos para mostrar la estructura de carpetas –o directorios– en la que los archivos se encuentran distribuidos.

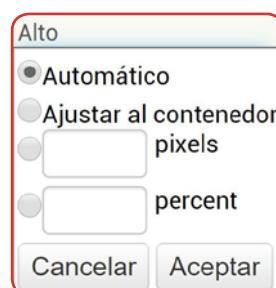
Una vez que todos los estudiantes hayan incorporado el panel de distribución horizontal y colocado allí los botones de colores, los orientamos para que agreguen el único componente restante. “Finalmente pudimos distribuir los botones de colores para que aparezcan uno al lado del otro. Sin embargo, todavía falta que agreguemos un componente importantísimo: el lienzo sobre el que dibujaremos”. Entonces, les indicamos que en *Panel > Dibujo y animación* se encuentra *Lienzo*: un panel sobre el que se puede, entre otras cosas, dibujar usando colores. Una vez que lo hayan incorporado arriba de la botonera de colores, les sugerimos que ejecuten la aplicación para observar el aspecto del programa mientras corre.



Incorporación del lienzo en el visor y la aplicación en ejecución

Continuamos: “Si bien ya incorporamos todos los componentes a la aplicación, todavía no luce como esperamos. Aún falta trabajar sobre las dimensiones de cada componente”. Damos tiempo a los estudiantes para que autónomamente exploren el entorno en busca de las opciones que permiten modificar el tamaño de los componentes y prueben distintas opciones.

Todos los componentes visibles tienen las propiedades *Alto* y *Ancho* (se muestran en el panel *Propiedades*). Al desplegarlo, se presentan cuatro opciones: *Automático*, en el que la elección de dimensiones se delega en App Inventor; *Ajustar al contenedor*, que produce que el componente ocupe todo el espacio disponible; [ ] *pixels*, para especificar una cierta cantidad fija de píxeles; y [ ] *percent*, para indicar un porcentaje de la dimensión de la pan-



Opciones para definir la altura de un componente

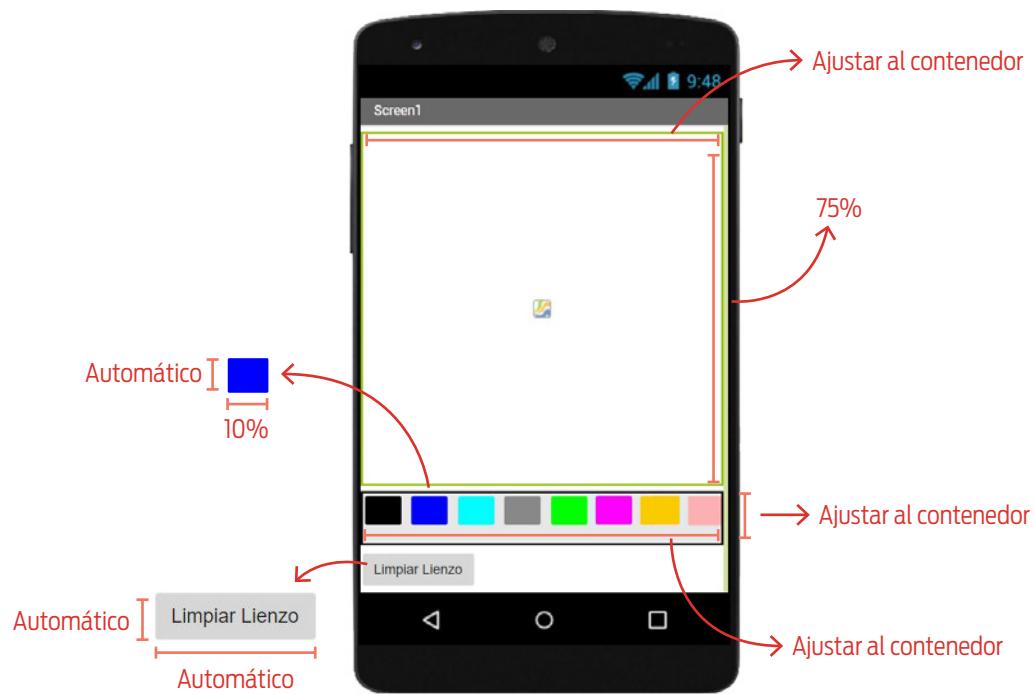
talla. Como la definición de la pantalla –es decir, la cantidad de píxeles que tiene– varía entre distintos modelos de teléfonos y tabletas, no es aconsejable usar la opción [ ] pixels. Tampoco es una buena idea usar en todos los componentes la opción *Automático*: rara vez se obtiene de este modo el aspecto buscado. En general, lo más conveniente es trabajar con *Ajustar al contenedor* y [ ] percent.

A continuación se muestran las opciones y valores que resultan en la distribución presentada en el desarrollo de la actividad.

COMPONENTE	ALTO	ANCHO
Lienzo	75%	Ajustar al contenedor
BotoneraDeColores	Ajustar al contenedor	Ajustar al contenedor
Todos los botones de colores	Automático	10%
BotónLimpiarLienzo	Automático	Automático

Dimensiones de los componentes

Les comentamos: “Es importante tener presente que no siempre lo que se muestra en el visor del editor de diseño refleja en forma exacta lo que se verá en la pantalla cuando la aplicación se esté ejecutando. Por ejemplo, en este caso, ya no se ven los botones rojo y amarillo. Sin embargo, aparecerán al correr el programa. Para estar seguros de que la aplicación se ve como esperamos, es importante correrla, ya sea en el emulador o en un teléfono o tableta”.



Dimensiones de *Dibujarte*

Los últimos detalles que falta acomodar para que la aplicación luzca tal como se presenta en la actividad se consiguen modificando algunas últimas propiedades de los componentes, que se exhiben a continuación.

COMPONENTE	PROPIEDAD	VALOR
Screen1	Título	Dibujarte
	Color de fondo	Gris
BotónLimpiarLienzo	Negrita	✓
	Color de texto	Blanco

Ajustes de propiedades

Les pedimos, entonces, que ejecuten la aplicación para corroborar que su aspecto sea el esperado.

### CIERRE

Reflexionamos junto a los estudiantes sobre la importancia de las interfaces gráficas al trabajar con aplicaciones en las que hay interacción persona-computadora. Mientras que una buena interfaz seduce y hace sentir cómodo a un usuario, una mala produce desagrado y, eventualmente, puede generar resistencia e incluso rechazo al uso de la aplicación. Les comentamos, también, que en general esta tarea no suele recaer solo sobre los programadores; también participan actores con otros perfiles, como analistas funcionales –que conocen exactamente el propósito de la aplicación o programa–, diseñadores gráficos, etc.

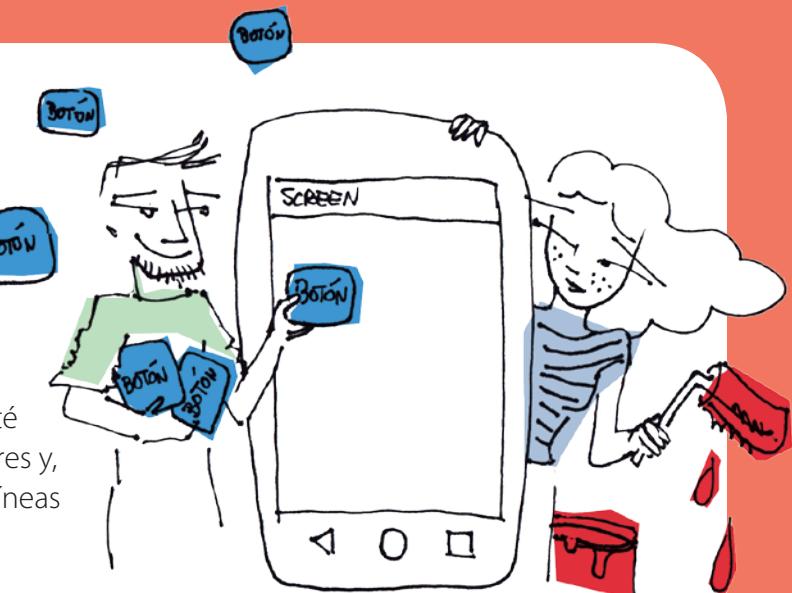
NOMBRE Y APELLIDO:

CURSO:

FECHA:

# LA INTERFAZ DE DIBUJARTE

¡Vamos a diseñar la interfaz gráfica de una aplicación para dibujar! Cuando la aplicación esté terminada, con nuestros dedos elegiremos colores y, arrastrándolos por la pantalla, podremos trazar líneas y colorear figuras. Por ahora, nos encargaremos de su aspecto. No solo del contenido viven las aplicaciones, ¡también de la forma!

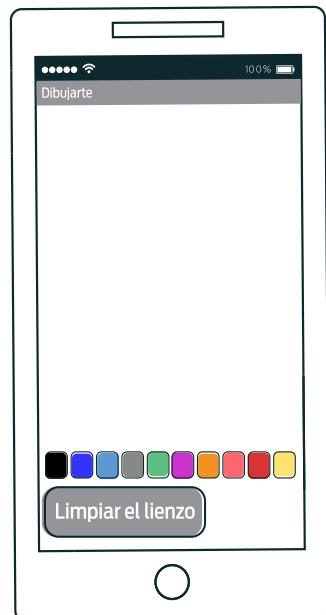


1. Mirá la interfaz de la aplicación e identificá sus componentes. ¿Qué tipo de componente son los cuadraditos de colores?



## LAS APARIENCIAS ENGAÑAN

Los componentes suelen lucir de un modo pero, en general, ese aspecto es algo que podemos modificar. A los botones, por ejemplo, les podemos quitar el texto, cambiarles el color de fondo, asociarlos a una imagen, etc. ¡Inspeccioná el panel de propiedades y fijate qué posibilidades te da cada tipo de componente!



2. Crea un proyecto llamado *Dibujarte* y agregá al visor todos los componentes que hayas identificado. ¿Te muestra algo parecido a esto? Si es así, ¿por qué? ¿Cómo ubica App Inventor por defecto los componentes que agregamos a nuestras aplicaciones?

---

---

---

---



NOMBRE Y APELLIDO:

CURSO:

FECHA:

- 3.** Cambiá el color de fondo de los botones y acomodá los textos.  
Deberías conseguir ver algo así. ¿Lo lograste? ¿Cómo lo hiciste?

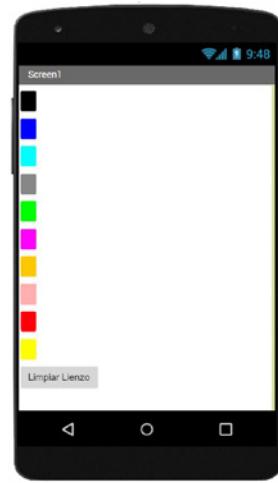
---

---

---

---

---

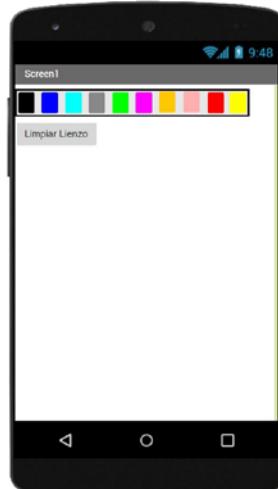


- 4.** Los botones deberían aparecer uno al lado del otro. Por favor, ocupate de corregirlo. Deberías conseguir verlo como se muestra en la figura. ¿Para qué te parece que sirven los componentes que se encuentran en Paleta > Disposición listados a continuación?

Disposición horizontal: \_\_\_\_\_

Disposición vertical: \_\_\_\_\_

Disposición tabular: \_\_\_\_\_



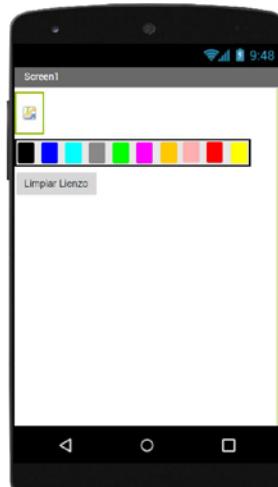
- 5.** ¡Todavía falta el lienzo en el que vamos a dibujar! Buscalo en Paleta > Dibujo y animación y agregalo. ¿Qué cosas se pueden hacer con un lienzo?

---

---

---

---



NOMBRE Y APELLIDO:

CURSO:

FECHA:

6. Ahora, ja ocuparse de las dimensiones! Fijate en las propiedades de cada componente que agregaste si podés jugar con los tamaños. Para ir viendo cómo queda, ejecutá la aplicación en el emulador. ¿Cómo hiciste para encontrar la medida justa de cada uno?

---

---

---

---

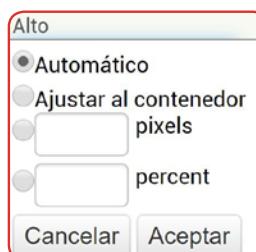


#### NO TODO LO QUE VES, ES

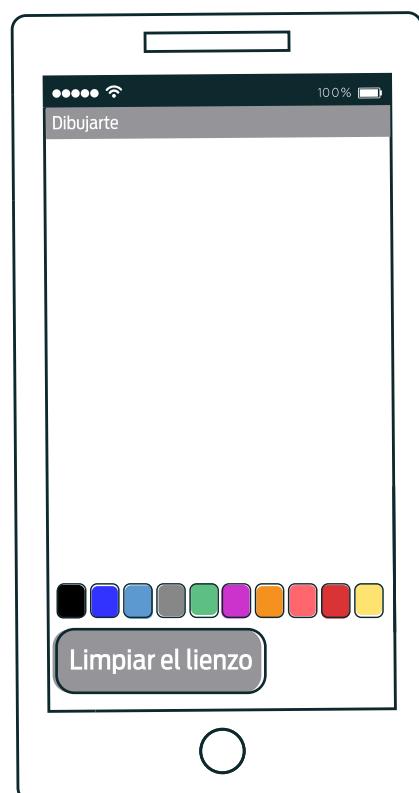
Si desaparecieron algunos botones, no te preocupes. Lo que muestra el visor de diseño no es exactamente lo que se va a ver cuando ejecutes la aplicación.

#### ALTO Y ANCHO

App Inventor presenta cuatro opciones para establecer el alto y el ancho de un componente visible: **Automático**, en el que la elección de dimensiones se delega en App Inventor; **Ajustar al contenedor**, que produce que el componente ocupe todo el espacio disponible; **[] pixels**, para especificar una cierta cantidad fija de píxeles; y **[] percent**, para indicar un porcentaje de la dimensión de la pantalla.



7. Últimos detalles: cambiá el título del componente Screen1 y hacé que el botón para limpiar el lienzo tenga fondo gris y letras blancas y en negrita. Ahora sí, ejecutá la aplicación y fijate si se ve como esperamos.



## Actividad 2

### Dibujarte en acción



#### OBJETIVOS

- Diferenciar presentación y lógica de una aplicación.
- Publicar aplicaciones.

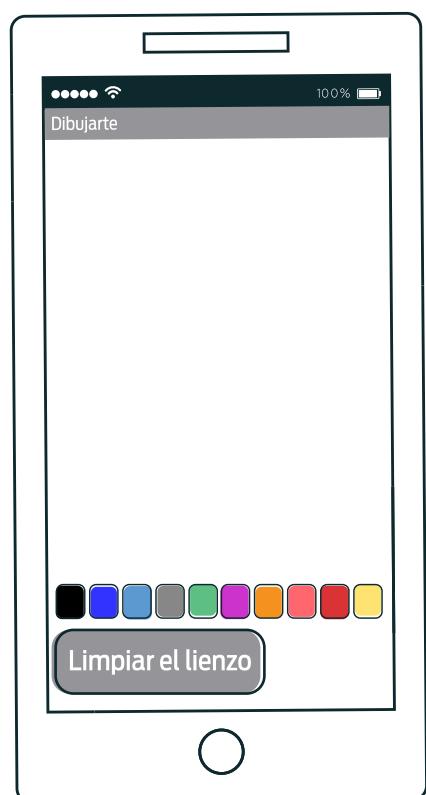
#### MATERIALES

- Computadora
- Internet
- MIT App Inventor 2
- Ficha para estudiantes
- Anexo “Compartimos aplicaciones”
- Teléfono con Android (opcional)

#### DESARROLLO

En esta actividad, los estudiantes programarán el comportamiento de la aplicación *Dibujarte*. Al finalizar, reconocerán la diferencia entre la **lógica de una aplicación** –es decir, su comportamiento– y su **presentación** –su aspecto–. Además, aprenderán cómo compartir sus programas con miembros de la comunidad de App Inventor, familiares, amigos, etc.

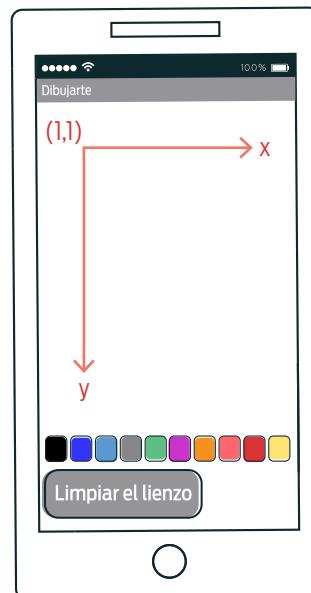
Comenzamos la actividad pidiéndoles a los estudiantes que abran y recuperen el proyecto de App Inventor *Dibujarte* y lo ejecuten en el emulador. A continuación les decimos: “Seleccione un color y dibujen sobre el lienzo. ¿Pudieron dibujar? ¿Por qué?”. Prestamos atención a sus observaciones y les comentamos: “Hasta ahora lo único que hemos hecho es disponer componentes sobre la pantalla. Es decir, solo trabajamos sobre su presentación. Aún resta definir su lógica. O, lo que es lo mismo, su comportamiento: cómo debe responder la aplicación cuando se hace tal o cual cosa”.



Interfaz gráfica de *Dibujarte*

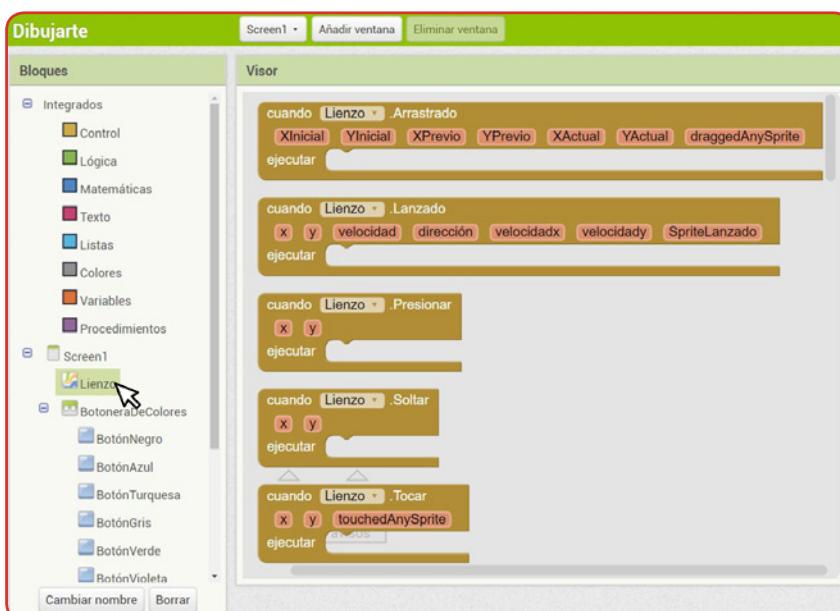
Repartimos la ficha de la actividad y los invitamos a que resuelvan la primera consigna. Allí se pide que, al presionar sobre el lienzo, la aplicación dibuje un punto en el lugar presionado. Les damos tiempo para que intenten resolver el desafío autónomamente.

Un lienzo es un panel rectangular bidimensional sensible al tacto en el que se puede, entre otras cosas, hacer un dibujo. Cualquier ubicación en el lienzo puede identificarse con un par de valores ( $x, y$ ), donde  $x$  es la distancia al borde izquierdo del lienzo e  $y$  es la distancia al borde superior del lienzo –en ambos casos, medida en cantidad de píxeles–.<sup>1</sup> Con (1,1) se identifica al vértice superior izquierdo del lienzo.



Ejes de coordenadas de los lienzos

Como cualquier otro componente de App Inventor, los lienzos también tienen asociados bloques para manejar eventos. Una vez que hayamos entrado al editor de bloques, al hacer clic sobre *Bloques > Screen1 > Lienzo* los veremos aparecer en la pantalla.



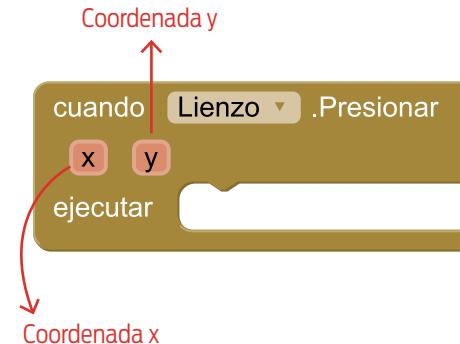
Bloques para manejar los eventos de un lienzo

<sup>1</sup> Un píxel es la superficie homogénea (en brillo y color) más pequeña que compone una imagen digital. En las pantallas de las computadoras, una imagen se muestra casi siempre mediante una cuadrícula de píxeles.

Para resolver la consigna debemos usar cuando `Lienzo.Presionar (x) (y) / ejecutar { }`. Cada vez que se presione el lienzo, se ejecutarán todas las instrucciones que coloquemos en su interior. Para definir el comportamiento frente a este evento contamos con dos valores, llamados **parámetros**,<sup>1</sup> de suma utilidad: las coordenadas `x` e `y` en las que el lienzo fue presionado. Es importante tener en claro que los bloques que incluyamos dentro se ejecutarán cada vez que se presione el lienzo, y el lienzo puede presionarse en distintos lugares. Por lo tanto, los valores que adquieran `x` e `y` variarán entre distintos manejos del evento.

En general, para utilizar un parámetro `p` hay que posicionar el puntero del ratón sobre él. Entonces, aparecerán dos bloques: `tomar (p)` y `poner (p) a [ ]`, que pueden arrastrarse y soltarse como cualquier otro. El primero permite obtener el valor contenido en `p`, y el segundo, asignarle un valor nuevo. En este caso, solo nos interesará leer los valores de `x` e `y`.

Finalmente, para dibujar un punto sobre el lienzo usaremos, junto con los parámetros, el comando `Lienzo.DibujarPunto x [ ] y [ ]`, que se encuentra disponible en *Bloques > Screen1 > Lienzo*. En la figura puede observarse la solución.



Bloque para definir el comportamiento de la aplicación cuando se presiona el lienzo



Parámetros que indican dónde se presionó el lienzo



Solución de la primera consigna

<sup>1</sup> En el capítulo 3, “Procedimientos”, se estudian los parámetros en profundidad.

Una vez que todos hayan finalizado, hacemos una puesta en común. Luego, los invitamos a que continúen con la segunda consigna, que pide que, al presionar el lienzo con el dedo y arrastrarlo por la pantalla, se dibuje una línea que siga el recorrido del dedo hasta que este se levante. Nuevamente, les damos tiempo para que exploren el entorno en busca de nuevas herramientas para completar el desafío.

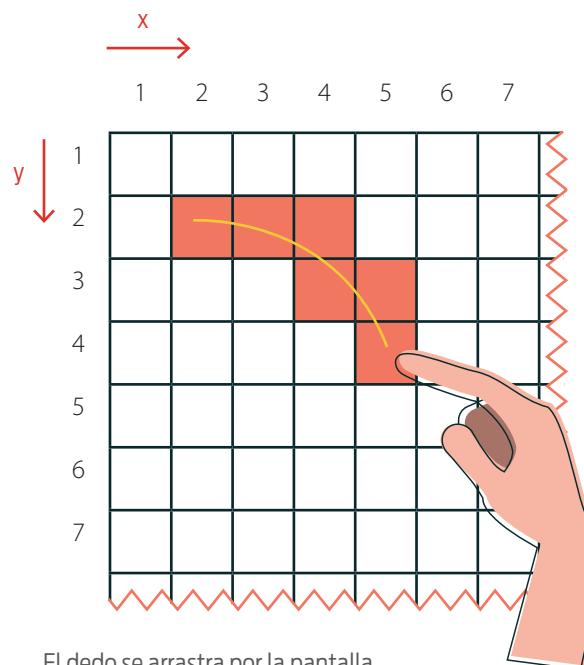
En este caso, deberán usar el bloque `cuando Lienzo.Arrastrado (XInicial) (YInicial) (XPrevio) (YPrevio) (XActual) (YActual) (draggedAnySprite) / ejecutar { }`, que sirve para definir el comportamiento del programa cuando se presiona la pantalla y se arrastra el dedo sobre el lienzo.



Bloque para definir qué hace la aplicación cuando se arrastra el dedo sobre el lienzo

Es probable que a los estudiantes no les resulte del todo claro cuándo se dispara la ejecución del bloque y qué representa cada uno de los parámetros. Por lo tanto, es importante que vayamos monitoreando cómo avanzan y, de ser necesario, los orientaremos.

Lo primero que deben comprender es que, cada vez que el dedo se desplaza de un píxel a otro del lienzo, se dispara la ejecución de las instrucciones que coloquemos dentro del bloque. Por ejemplo, si presionamos el lienzo en la posición (2,2) y lo arrastramos hasta la posición (5,4), pasando en el medio por las posiciones (3,2), (4,2), (4,3) y (5,3), se disparará 5 veces el manejo del evento: la primera cuando el dedo pase de (2,2) a (3,2), la segunda cuando pase de (3,2) a (4,2), y así siguiendo hasta que pase de (5,3) a (5,4).



El dedo se arrastra por la pantalla

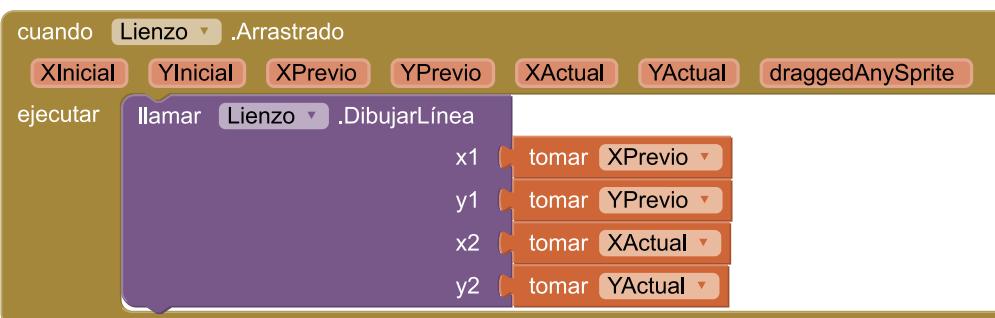
También tienen que vislumbrar lo que representan los parámetros: los valores de `xPrevio`, `yPrevio`, `xActual` e `yActual` indican que se produjo un desplazamiento desde la posición del lienzo (`xPrevio`, `yPrevio`) hasta (`xActual`, `yActual`); `xInicial` e `yInicial` contienen las coordenadas donde el usuario presionó el lienzo por primera vez –al comenzar el desplazamiento–; por último, `draggedAnySprite` indica si el arrastre se hizo sobre algún objeto posicionado sobre el lienzo o directamente sobre este. Para evitar confusiones, sugerimos ignorar este último parámetro.

Siguiendo con el ejemplo, los valores que adquirirán los parámetros en las sucesivas llamadas pueden observarse en la siguiente tabla:

#LLAMADA	XINICIAL	YINICIAL	XPREVIO	YPREVIO	XACTUAL	YACTUAL
1	2	2	2	2	3	2
2	2	2	3	2	4	2
3	2	2	4	2	4	3
4	2	2	4	3	5	3
5	2	2	5	3	5	4

Bloque para definir qué hace la aplicación cuando se arrastra el dedo sobre el lienzo

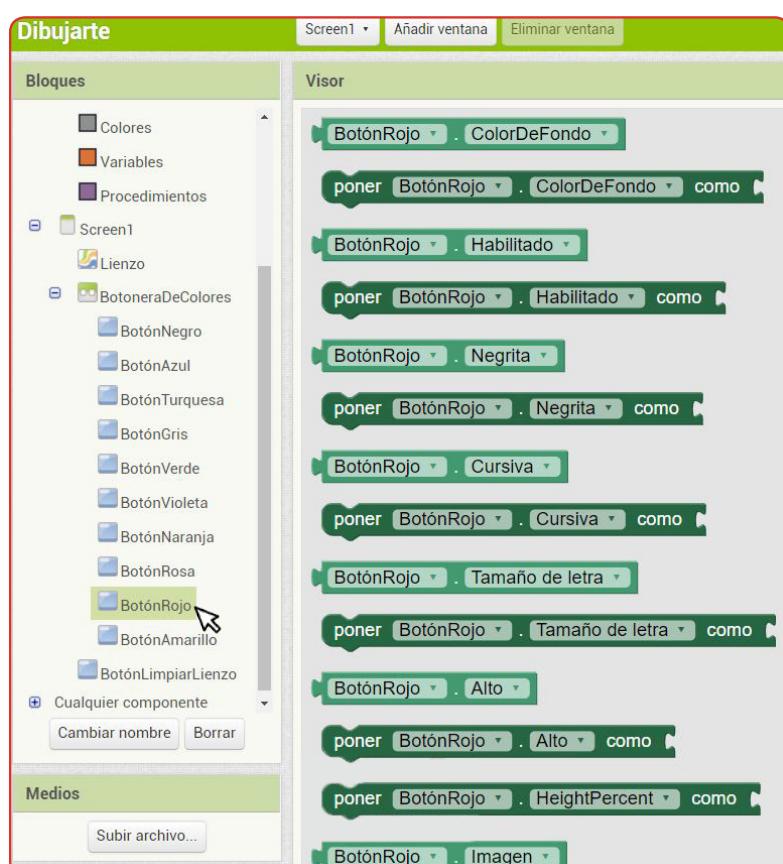
Una vez presionado el lienzo, el programa tiene que dibujar una línea cada vez que el dedo se desplace de un píxel a otro. Para hacerlo, hay que usar el comando `Lienzo.DibujarLínea x1 [ ] y1 [ ] x2 [ ] y2 [ ]`, al que hay que indicar las coordenadas de inicio y fin de la línea que queremos dibujar.



Solución de la segunda consigna

Luego de hacer una nueva puesta en común les proponemos que resuelvan la tercera consigna. En este caso, deben programar el manejo de los eventos de todos los botones: tanto los que se usan para establecer el color para pintar el lienzo como el que borra todo lo que se haya dibujado.

Al diseñar la interfaz gráfica de la aplicación modificamos algunas propiedades de los componentes que incluimos en la pantalla. Por ejemplo, a los botones de selección de color les eliminamos el texto y les cambiamos el color de fondo. En App Inventor, el valor de las propiedades no solo puede modificarse desde el editor de diseño; también puede hacerse programáticamente en el editor de bloques. Al seleccionar un componente del panel *Bloques*, aparecerán una serie de bloques verdes que nos permite tanto obtener como modificar los valores de sus propiedades. Por ejemplo, si seleccionamos un botón, veremos bloques para acceder a su color de fondo, su tamaño de letra, etc.



Bloques para acceder a las propiedades de un botón

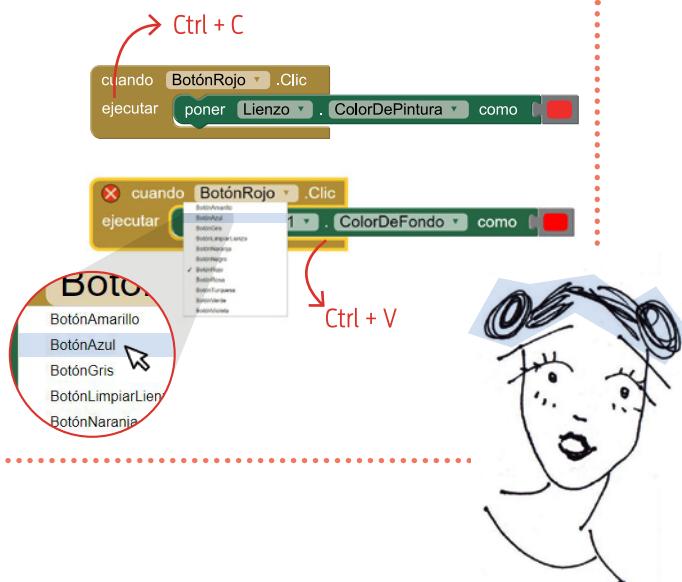
Una de las propiedades de un lienzo es su color de pintura –es decir, el color que tiene lo que se dibuja sobre él–. Entonces, cada vez que se presione un botón para seleccionar un color, lo que hay que hacer es establecer el color de la pintura del lienzo en el color que corresponda. Esto debemos hacerlo con cada uno de los 10 botones dispuestos para seleccionar colores.

```
cuando BotónRojo .Clic
ejecutar poner Lienzo . ColorDePintura como
```

El color del dibujo sobre el lienzo pasa a ser rojo

### PARA COPIAR, PEGAR Y DESHACER

Una forma sencilla de copiar y pegar bloques es usar la tecla **CTRL** junto a las letras **C** (para copiar) y **V** (para pegar). Al nuevo conjunto de bloques se lo puede modificar de acuerdo a lo que se necesite. Además, para deshacer lo último que se hizo mientras se arma un programa, se puede presionar **CTRL + Z**.



Para limpiar el lienzo usaremos uno de los comandos que ofrecen los componentes de este tipo:  
**Lienzo.Limpiar**. De esta forma completamos la tercera consigna.

**cuando BotónLimpiarLienzo .Clic**  
ejecutar [llamar Lienzo .Limpiar]

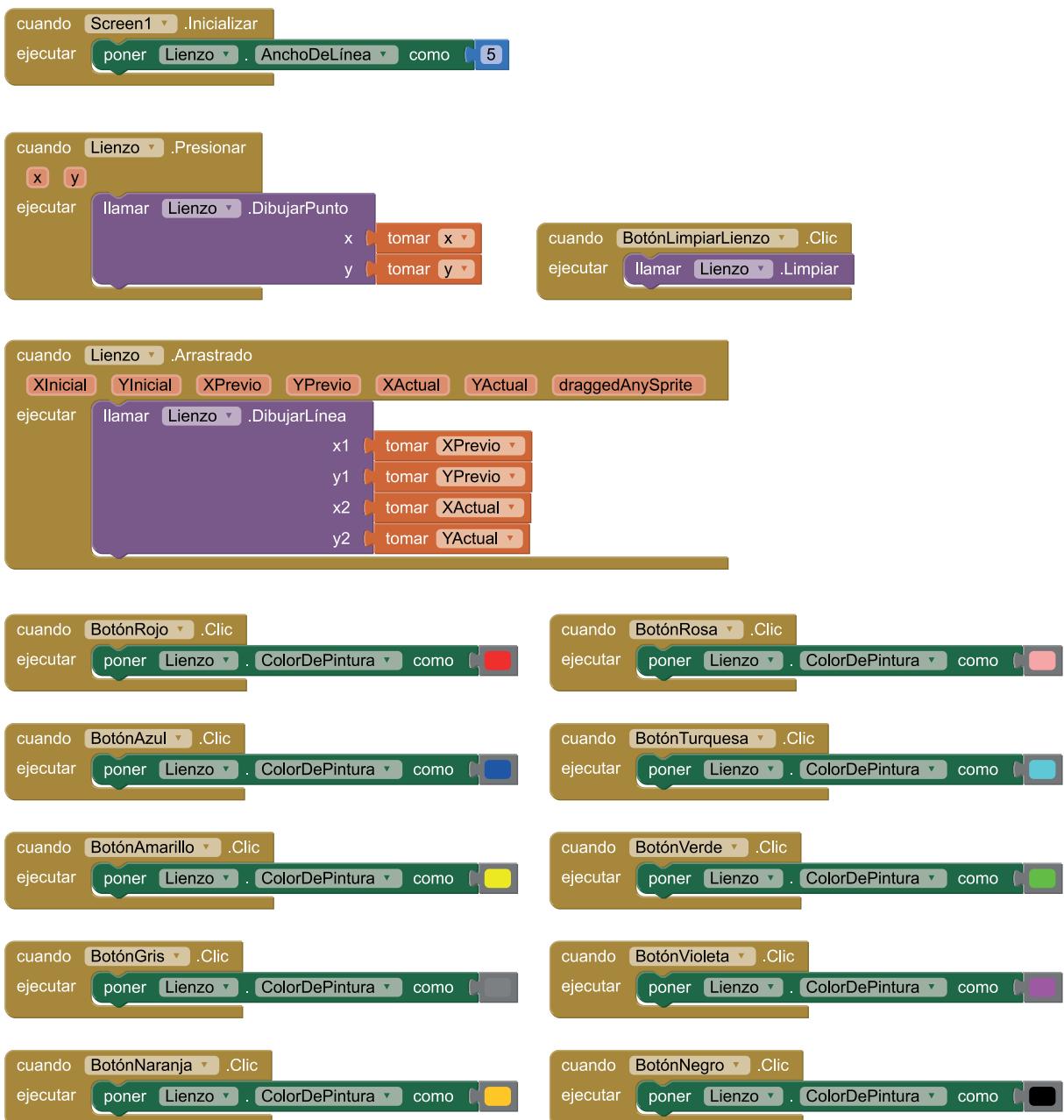
Se limpia el lienzo cuando se presiona  
**BotónLimpiarLienzo**

Resta resolver la cuarta y última consigna. Allí se pide ajustar a 5 el ancho de la línea que aparecerá sobre el lienzo cuando dibujemos (por defecto es 1), lo que se consigue modificando la propiedad **Lienzo.AnchoDeLínea** y usando el bloque para definir valores numéricos disponible en *Bloques > Integrados > Matemática*. A diferencia de lo que programamos hasta ahora, esto no debe realizarse como respuesta a un evento disparado por un usuario, sino que tiene que definirse cuando la aplicación comience a ejecutarse. Para establecer valores iniciales –a lo que comúnmente se llama **inicializar la aplicación**–, hay que manejar un evento especial que se produce en el instante en que la pantalla se carga en el teléfono: **screen1.Inicializar**. Al fijar allí el ancho de línea –cuando la aplicación se carga–, completamos el programa.

**cuando Screen1 .Inicializar**  
ejecutar [poner Lienzo . AnchoDeLínea como 5]

Se inicializa el ancho de línea del lienzo

A continuación puede observarse la solución completa de la actividad:



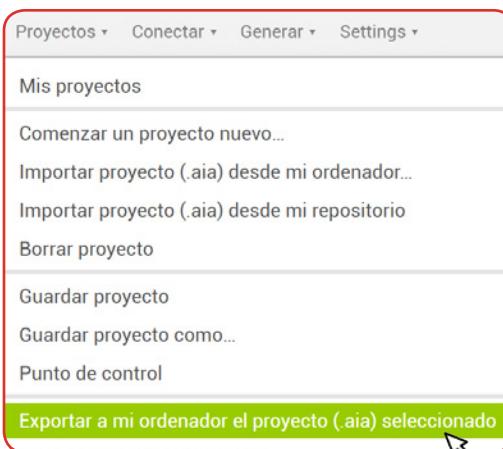
Programa completo

## Compartimos la aplicación

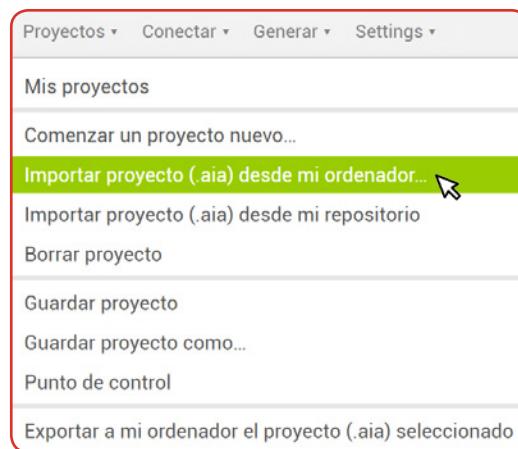
Una vez que todos hayan completado el programa, les repartimos el anexo de la actividad y les mostramos distintas formas en las que pueden compartir su aplicación. Existen varias alternativas, que se describen a continuación.

### Exportar e importar el proyecto

Para compartir el proyecto con otras personas que tengan una cuenta de usuario App Inventor, se puede (i) exportar el proyecto como un archivo con extensión .aia; (ii) enviárselo a la otra persona (ya sea mediante correo electrónico, dándoselo en un pendrive, etc.); y (iii) que la otra persona lo importe. Una vez hecho esto, lo verá aparecer en la lista de proyectos de su usuario.



Exportación de un proyecto



Importación de un proyecto

### Publicar en la galería de App Inventor

La galería de MIT App Inventor es el sitio principal para compartir aplicaciones construidas con App Inventor y acceder a aplicaciones desarrolladas por otros. Todo lo que se publica en la galería es de código abierto: no solo se pueden usar las aplicaciones publicadas sino que también se puede acceder a su interior –es decir, a su diseño y a sus bloques–. Es sorprendente observar cómo aumenta el nivel de motivación de los estudiantes cuando saben que sus amigos y el público en general pueden ver sus producciones, descargarlas, etc.

Para publicar una aplicación en la galería, en primer lugar hay que acceder a la lista de nuestros proyectos y, luego, seleccionar el proyecto que queremos compartir y hacer clic en el botón *Publish to Gallery*.



Publicación de un proyecto

Entonces aparecerá un formulario que permite complementar la publicación con una foto, una descripción, un enlace a un video que muestre la aplicación en funcionamiento, etc. Para completar la publicación, solo hay que hacer clic en *PUBLISH*.

Cargá un ícono para la aplicación

Nombre de la aplicación

Si hiciste un tutorial, poné la URL

Descripción de la aplicación

Upload your project image!

Dibujarte

programar2020

Created Date: 2019/08/01  
Changed Date: 2019/08/01

If this app has a tutorial or video, please enter the URL here.

Are you remixing code from other apps? Credit them here.

Please write the description of the app here.

PUBLISH

CANCEL

By submitting an app in the gallery, you are publishing it under a [Creative Commons Attribution License](#), and affirming that you have the authority to do so.

Información adicional al publicar un proyecto

### Empaquetar la aplicación

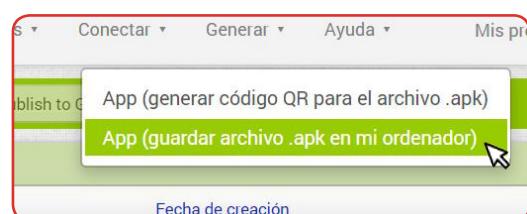
A diferencia de las formas descritas anteriormente, empaquetar la aplicación permite que esta se instale en cualquier dispositivo que corra el sistema operativo Android, sin necesidad de tener una cuenta de usuario de App Inventor. De esta forma, los estudiantes podrán compartir sus producciones con todos sus amigos, familiares, etc.

En primer lugar, hay que generar un archivo con extensión .apk y descargarlo a la computadora.<sup>1</sup> Esto puede hacerse desplegando el menú *Generar* y seleccionando la opción *App (guardar archivo .apk en mi ordenador)*. Luego, este archivo puede enviarse a otros por medio de correo electrónico, una aplicación de mensajería instantánea, etc. Cuando los receptores descarguen y abran el archivo en sus teléfonos, la aplicación se instalará.

Si se quisiese distribuir la aplicación de forma más amplia, también se la puede publicar en la tienda de aplicaciones *Google Play*. Sin embargo, esta opción no es gratuita. Para más información sobre cómo hacerlo, consultar la página <https://bit.ly/3lQqcCl>.

### CIERRE

Reflexionamos junto a los estudiantes acerca de la importancia tanto de la presentación como de la lógica de una aplicación. Si bien la interfaz gráfica solo determina su aspecto, su diseño puede provocar que sea aceptada o rechazada, independientemente de cuán bien esté programado su comportamiento. En este contexto, la forma y el contenido son ambos igualmente importantes.



Generación de archivo .apk

### PERMISOS DE INSTALACIÓN

Para instalar una aplicación que no provenga de la tienda oficial *Google Play* en un teléfono o una tableta Android, es necesario habilitar ciertos permisos. En general, esto puede hacerse editando las opciones de seguridad del dispositivo.



<sup>1</sup> Los archivos con extensión .apk son aquellos que se usan para distribuir e instalar aplicaciones en dispositivos con sistema operativo Android.

NOMBRE Y APELLIDO:

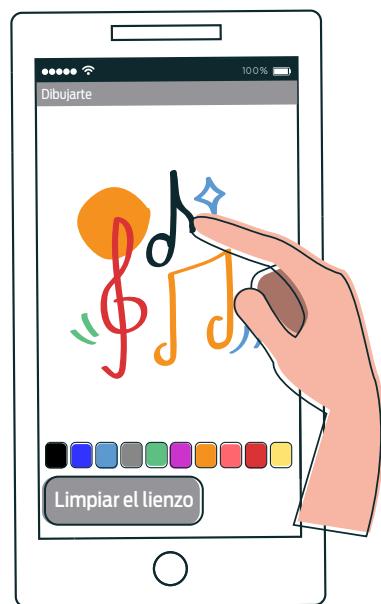
CURSO:

FECHA:

# DIBUJARTE EN ACCIÓN

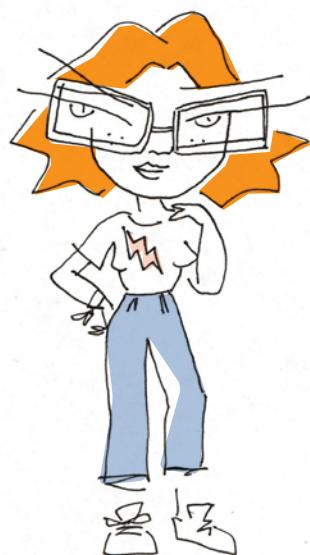
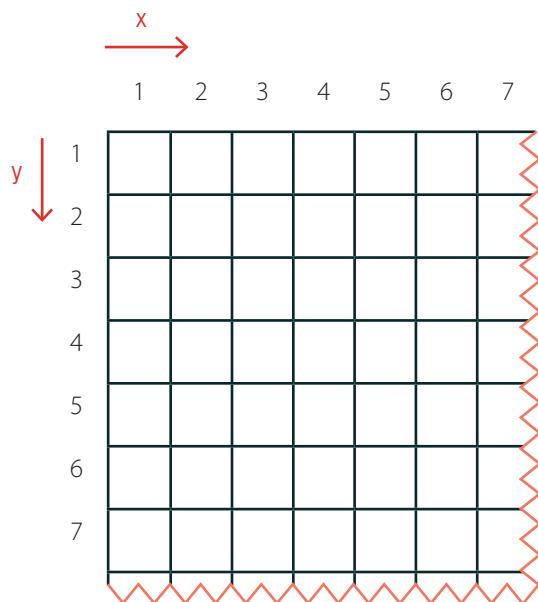
¡A encargarte de que *Dibujarte* nos permita dibujar!  
Seguí las consignas para poder jugar con puntos,  
líneas y colores, y volver a empezar.

1. Ocupate de que, cada vez que se presione con el dedo en algún lugar de lienzo, allí se dibuje un punto. Investigá el entorno en busca de cómo manejar los eventos que involucran a este componente.  
¿Cómo hiciste para identificar dónde se presiona el lienzo?



## LOS PUNTOS DE UN LIENZO

Un lienzo es un panel rectangular bidimensional sensible al tacto en el que se puede, entre otras cosas, hacer un dibujo. Cualquier ubicación en el lienzo puede identificarse con un par de valores ( $x, y$ ), donde  $x$  es la distancia al borde izquierdo del lienzo e  $y$  es la distancia al borde superior del lienzo –en ambos casos, medida en cantidad de píxeles–. Con (1,1) se identifica el vértice superior izquierdo del lienzo.



NOMBRE Y APELLIDO:

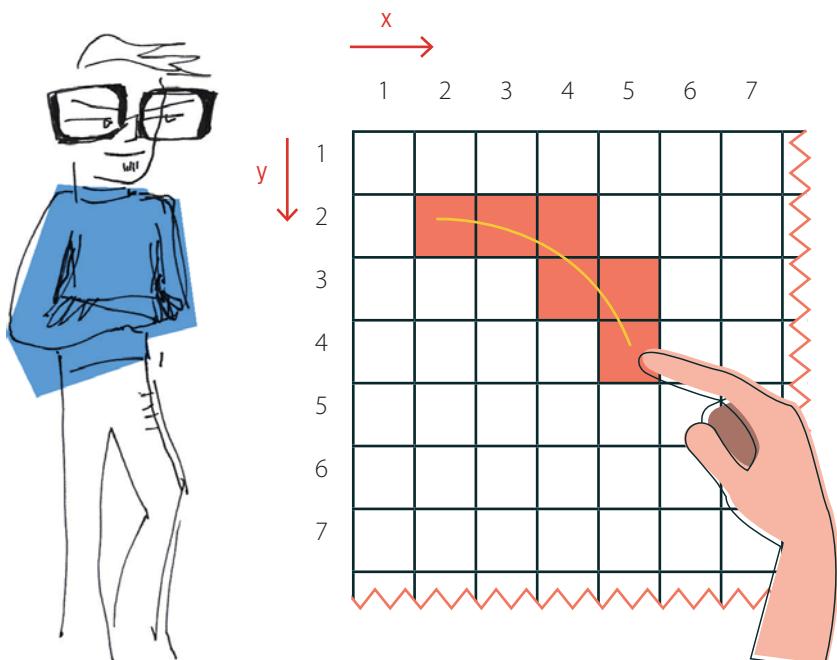
CURSO:

FECHA:

2. Ahora tenés que encargarte de que, cuando presionamos el lienzo con el dedo y lo arrastramos por la pantalla, se dibuje una línea que siga su recorrido hasta que lo levantemos.

### ARRASTRAMOS EL DEDO SOBRE EL LIENZO

En App Inventor tenemos un bloque que nos permite manejar el evento que se produce al arrastrar el dedo por un lienzo: **cuando Lienzo.Arrastrado (XInicial) (YInicial) (XPrevio) (YPrevio) (XActual) (YActual) (draggedAnySprite)** / **ejecutar { } .** Tené en cuenta que la ejecución de los bloques que pongas dentro se disparará cada vez que muevas el dedo de un píxel a otro. Por ejemplo, en el caso del gráfico, se producirá 5 veces.



¿Qué indican los parámetros **(XInicial)** **(YInicial)** **(XPrevio)** **(YPrevio)** **(XActual)** y **(YActual)** ?

PARÁMETRO	XINICIAL	YINICIAL	XPREVIO	YPREVIO	XACTUAL	YACTUAL
INDICA						

3. Llegó el momento de dibujar en colores y poder borrar lo que hicimos si no nos gusta. ¡A ocuparte de los botones! Una pista: las propiedades de los componentes no solo las podés modificar desde el editor de diseño; también podés hacerlo programáticamente. ¿Qué propiedad te sirvió para poder dibujar con distintos colores? ¿De qué componente es esa propiedad?

---

---

---

---

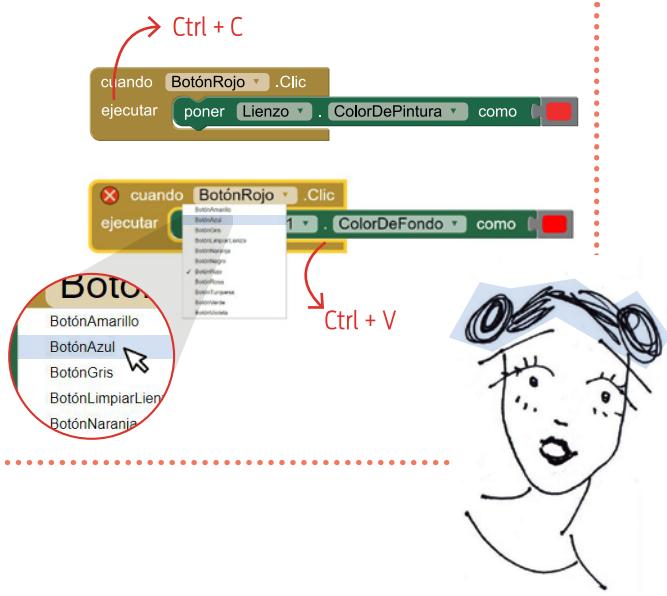
NOMBRE Y APELLIDO:

CURSO:

FECHA:

### PARA COPIAR, PEGAR Y DESHACER

Una forma sencilla de copiar y pegar bloques es usar la tecla CTRL junto a las letras C (para copiar) y V (para pegar). Al nuevo conjunto de bloques se lo puede modificar de acuerdo a lo que se necesite. Además, para deshacer lo último que se hizo mientras se arma un programa, se puede presionar CTRL + Z.



4. La línea de dibujo está muy finita. Cambiale el ancho para que tus dibujos sobre el lienzo luzcan mejor aún. ¡Terminantemente prohibido hacerlo desde el editor de diseño! ¿Qué evento manejaste para que, al cargar la aplicación, se defina el ancho de la línea? En general, ¿para qué puede ser útil manejar este evento?

---

---

---

---

---

---

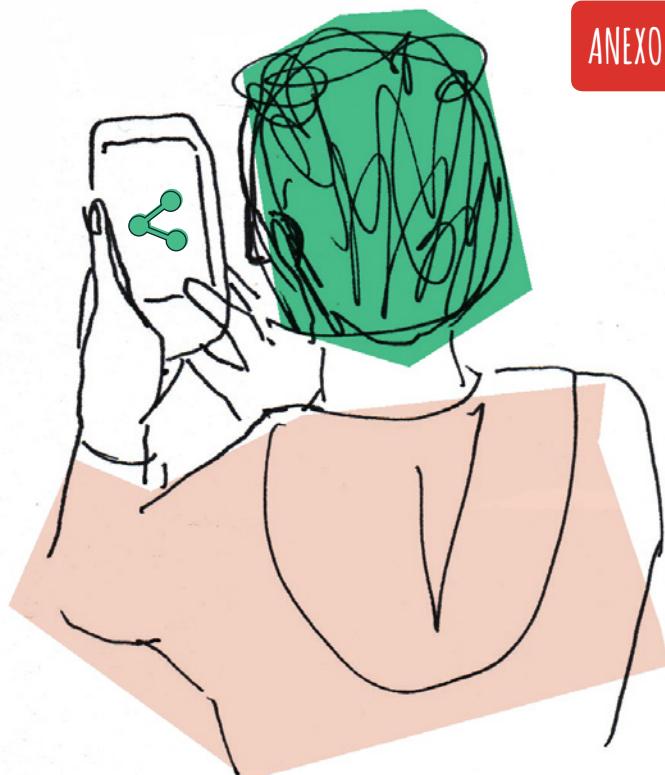
# COMPARTIMOS APLICACIONES

Todas las aplicaciones que hagas las podés compartir con amigos, familiares, miembros de la comunidad de App Inventor y público en general. Acá te mostramos algunas formas de hacerlo.

ANEXO

## A. Exportar e importar el proyecto

Para compartir el proyecto con otras personas que tengan una cuenta de usuario App Inventor, podés (i) exportar el proyecto, generando un archivo con extensión .aia; (ii) enviárselo a la otra persona (ya sea mediante correo electrónico, dándoselo en un *pendrive*, etc.); y (iii) indicarle a esta persona que lo importe. Una vez hecho esto, lo verá aparecer en la lista de proyectos de su usuario.



Proyectos ▾ Conectar ▾ Generar ▾ Settings ▾

Mis proyectos

Comenzar un proyecto nuevo...

Importar proyecto (.aia) desde mi ordenador...

Importar proyecto (.aia) desde mi repositorio

Borrar proyecto

Guardar proyecto

Guardar proyecto como...

Punto de control

**Exportar a mi ordenador el proyecto (.aia) seleccionado**

Proyectos ▾ Conectar ▾ Generar ▾ Settings ▾

Mis proyectos

Comenzar un proyecto nuevo...

**Importar proyecto (.aia) desde mi ordenador...**

Importar proyecto (.aia) desde mi repositorio

Borrar proyecto

Guardar proyecto

Guardar proyecto como...

Punto de control

Exportar a mi ordenador el proyecto (.aia) seleccionado

Se exporta un proyecto



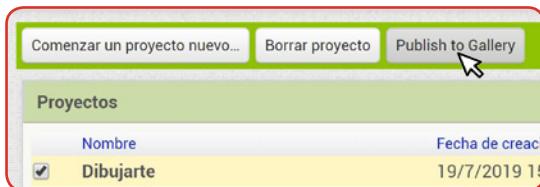
Se envía el archivo .aia

Se importa un proyecto

## B. Publicar en la galería de App Inventor

La galería de MIT App Inventor es el sitio principal para compartir aplicaciones construidas con App Inventor y acceder a aplicaciones desarrolladas por otros. Todo lo que se publica en la galería es de código abierto: no solo se pueden usar las aplicaciones publicadas sino que también se puede acceder a su interior –es decir, a su diseño y a sus bloques–.

Para publicar una aplicación en la galería, en primer lugar hay que acceder a la lista de nuestros proyectos y, luego, seleccionar el proyecto que queremos compartir y hacer clic en el botón *Publish to Gallery*.



Entonces aparecerá un formulario que permite complementar la publicación con una foto, una descripción, un enlace a un video que muestre la aplicación en funcionamiento, etc. Para completar la publicación, solo hay que hacer clic en *PUBLISH*.

Cargá un ícono para la aplicación

Upload your project image!

PUBLISH →

CANCEL

Nombre de la aplicación

Dibujarte

programar2020

Created Date: 2019/08/01  
Changed Date: 2019/08/01

If this app has a tutorial or video, please enter the URL here.

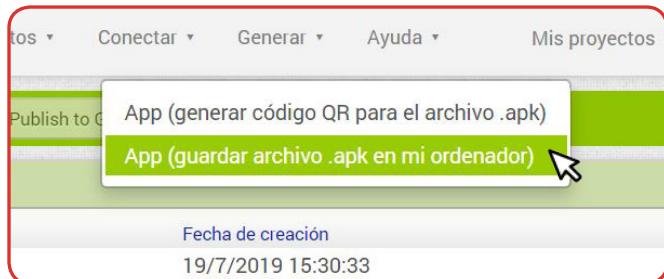
Are you remixing code from other apps? Credit them here.

Please write the description of the app here.

Si hiciste un tutorial, poné la URL

Si el proyecto está basado en otros proyectos de App Inventor, referencialos acá

Descripción de la aplicación



### PERMISOS DE INSTALACIÓN

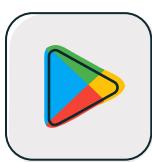
Para instalar una aplicación que no provenga de la tienda oficial Google Play en un teléfono o una tableta Android, es necesario habilitar ciertos permisos. En general, esto puede hacerse editando las opciones de seguridad del dispositivo.



### C. Empaquetar la aplicación

A diferencia de las otras opciones, si empaquetás la aplicación vas a poder instalarla en cualquier dispositivo que corra el sistema operativo Android.

En primer lugar, hay que generar un archivo con extensión .apk y descargarlo a la computadora (los archivos con extensión .apk son aquellos que se usan para distribuir e instalar aplicaciones en dispositivos con sistema operativo Android). Esto podés hacerlo desplegando el menú *Generar* y seleccionando la opción *App (guardar archivo .apk en mi ordenador)*. Luego, podés enviar este archivo a otros por medio de correo electrónico, una aplicación de mensajería instantánea, etc. Cuando los receptores lo descarguen y lo abran en sus teléfonos, la aplicación se instalará.



Si quisieras distribuir la aplicación de forma más amplia, también la podés publicar en la tienda de aplicaciones Google Play. Sin embargo, esta opción no es gratuita. Para más información sobre cómo hacerlo, consultá la página <https://bit.ly/31QqcCl>.

# 03

# PROCEDIMIENTOS

**SECUENCIA DIDÁCTICA 1**  
ELIMINACIÓN DE REDUNDANCIA  
Cielito lindo  
Filtros de colores

**SECUENCIA DIDÁCTICA 2**  
PARÁMETROS  
Elefantes que se balancean  
Tu arte pop

Un procedimiento es una porción de un programa más grande que encapsula una tarea específica. Los procedimientos suelen usarse para descomponer problemas complejos en piezas más simples que, al combinarlas, resuelven el problema original. Bien utilizados, mejoran notablemente la legibilidad de los programas.

Las secuencias didácticas de este capítulo introducen cómo definir y usar procedimientos. Las actividades proponen la reflexión sobre la redundancia en los programas y muestran de qué manera puede evitarse mediante el uso de procedimientos. Además, presenta el uso de parámetros para obtener soluciones generales que resuelven distintos problemas particulares.



## Secuencia Didáctica 1

# ELIMINACIÓN DE REDUNDANCIA

Es habitual que, al programar, aparezcan distintas porciones del programa idénticas entre sí; por ejemplo, cuando queremos que dos acciones del usuario (como presionar un botón y seleccionar una opción de un menú) generen la misma respuesta por parte del programa. En estos casos, si quisieramos modificar la reacción de la aplicación, estaríamos obligados a introducir cambios en todas las partes del programa en las que se encuentran esas acciones.

En general, la generación de redundancia es una mala práctica de programación. En esta secuencia didáctica se introduce el uso de procedimientos como solución a la reiteración de fragmentos iguales en los programas.

.....  
**OBJETIVOS**

- Introducir la noción de procedimiento.
  - Presentar a los procedimientos como herramienta para evitar redundancia.
- .....

## Actividad 1

### Cielito lindo



TODA LA CLASE

**OBJETIVO**

- Presentar la noción de procedimiento.

**MATERIALES**

- Reproductor de audio
- Papel
- Bolígrafo

**DESARROLLO**

Con el fin de presentar la noción de procedimiento, trabajaremos con una canción. La analogía entre los estribillos de las canciones y los procedimientos usados en programación resulta adecuada para acercarnos a esta herramienta. A modo de ejemplo, se ilustra la actividad con la popular canción mexicana *Cielito lindo*, pero se puede seleccionar una canción que resulte significativa para el curso.

Comenzamos la actividad indicándoles a los estudiantes que tendrán que escribir la letra completa de la canción que van a escuchar. Podemos reproducirla más de una vez, porque es probable que no todos consigan transcribir toda la canción en un primer intento. Además, sugerimos disponer de algunas fotocopias con la letra para repartir entre los estudiantes que no lleguen a completarla.<sup>1</sup>

**Cielito Lindo**

De la Sierra Morena,  
cielito lindo, vienen bajando  
un par de ojitos negros,  
cielito lindo, de contrabando.

iAy, ay, ay, ay!  
canta y no llores,  
porque cantando se alegran,  
cielito lindo, los corazones.

Ese lunar que tienes,  
cielito lindo, junto a la boca  
no se lo des a nadie,  
cielito lindo, que a mí me toca.

iAy, ay, ay, ay!  
canta y no llores,  
porque cantando se alegran,  
cielito lindo, los corazones.

Siempre que te enamores  
mira primero, mira primero  
dónde pones los ojos,  
cielito lindo, no llores luego.

iAy, ay, ay, ay!  
canta y no llores,  
porque cantando se alegran,  
cielito lindo, los corazones.

De tu reja a la mía,  
cielito lindo, no hay más que un paso  
ora que estamos solos,  
cielito lindo, dame un abrazo.

iAy, ay, ay, ay!  
canta y no llores,  
porque cantando se alegran,  
cielito lindo, los corazones.

Letra de *Cielito lindo*

<sup>1</sup> Existen distintas versiones de *Cielito lindo*. La presentada en la actividad es una entre varias; a los efectos de la actividad, puede emplearse esta u otra.

Una vez que todos hayan concluido de transcribir la letra, les preguntamos: “¿Cuántas veces tuvieron que escribir el estribillo, es decir, la parte que dice: ‘Ay, ay, ay, ay! / canta y no llores...’? ¿Se les ocurre alguna forma más corta de hacerlo?”. De este modo, guiamos la discusión para que los estudiantes lleguen a la conclusión de que lo más conveniente es escribir el estribillo entero una sola vez y luego referenciarlo donde corresponda. Teniendo esto en cuenta, la letra podría escribirse de la siguiente manera:

### Cielito Lindo

De la Sierra Morena,  
cielito lindo, vienen bajando  
un par de ojitos negros,  
cielito lindo, de contrabando.

#### **Estríbillo:**

*iAy, ay, ay, ay!  
canta y no llores,  
porque cantando se alegran,  
cielito lindo, los corazones.*

Ese lunar que tienes,  
cielito lindo, junto a la boca  
no se lo des a nadie,  
cielito lindo, que a mí me toca.

#### [Estríbillo]

Siempre que te enamores  
mira primero, mira primero  
dónde pones los ojos,  
cielito lindo, no llores luego.

#### [Estríbillo]

De tu reja a la mía,  
cielito lindo, no hay más que un paso  
ora que estamos solos,  
cielito lindo, dame un abrazo.

#### [Estríbillo]

*Cielito lindo*, sin repeticiones explícitas del estribillo

### CIERRE

Reflexionamos con los estudiantes sobre lo tedioso que resulta repetir versos idénticos al escribir y observamos que es mucho más práctico escribir el estribillo solo una vez. Luego, mediante una simple referencia evitamos repetir un texto cuando tenemos que escribir la letra de una canción. Por supuesto, nunca cantamos la palabra *estribillo*, sino los versos que corresponden a la parte que este engloba. Terminamos contándoles a los estudiantes que algo parecido podemos hacer al programar con una herramienta llamada **procedimiento**.

## Actividad 2

### Filtros de colores

 DE A DOS

#### OBJETIVOS

- Definir procedimientos.
- Refactorizar programas.

#### MATERIALES

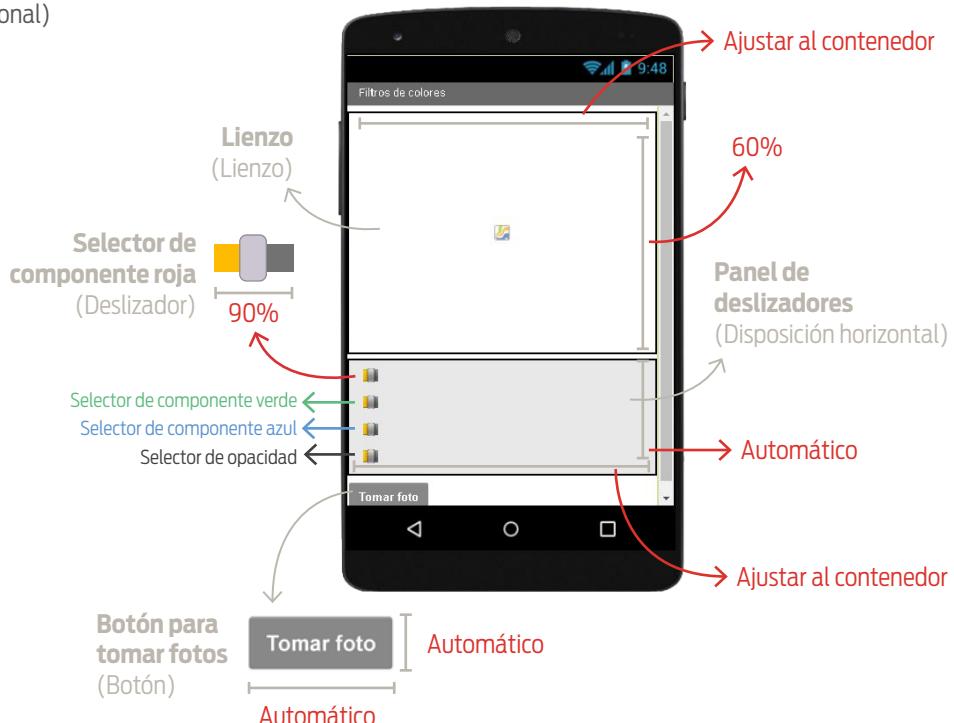
-  Computadora
-  Internet
-  MIT App Inventor 2
-  Ficha para estudiantes
-  Teléfono con Android (opcional)

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes construyan una aplicación usando procedimientos. A medida que desarrollen la propuesta, notarán que hay un conjunto de bloques que se repite en forma exacta en distintas partes del programa. Para evitar reiteraciones innecesarias, les resultará conveniente crear un único procedimiento que contenga tales bloques e invocarlo cada vez que sea necesario.<sup>1</sup>

#### Consigna 1: interfaz gráfica

Comenzamos repartiendo la ficha de la actividad a los estudiantes y les contamos que van a desarrollar una aplicación con la que podrán aplicar filtros de colores a fotos tomadas con la cámara de sus teléfonos. Les pedimos, entonces, que resuelvan la primera consigna, en la que se dan instrucciones para armar la interfaz gráfica de la aplicación. A continuación se presenta un esquema con las dimensiones de cada componente de la interfaz. Luego, hay una tabla que muestra los valores de las propiedades que hay que modificar en relación a los que App Inventor asigna por defecto, para que la aplicación se vea tal como se presenta en el desarrollo de la actividad.



Diseño de la interfaz gráfica

<sup>1</sup> Puede accederse a una versión completa de la aplicación buscando "programar2020" en la galería de aplicaciones de App Inventor.

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1 <sup>1</sup>	Pantalla	Título	Screen	Filtros de colores
Lienzo	Lienzo	Alto	Automático	60%
		Ancho	Automático	Ajustar al contenedor
PanelDeslizadores	Disposición horizontal	Disp. horizontal	Izquierda	Centro
		Disp. vertical	Arriba	Centro
		Ancho	Automático	Ajustar al contenedor
DeslizadorRojo	Deslizador	Color izquierda	Por defecto	Rojo
		Color derecha	Por defecto	Rojo
		Ancho	Automático	90%
		Valor máximo	50	255
		Valor mínimo	10	0
		Posición del pulgar	30	0
DeslizadorVerde <sup>2</sup>	Deslizador	Color izquierda	Por defecto	Verde
		Color derecha	Por defecto	Verde
DeslizadorAzul <sup>3</sup>	Deslizador	Color izquierda	Por defecto	Azul
		Color derecha	Por defecto	Azul
DeslizadorOpacidad <sup>4</sup>	Deslizador	Color izquierda	Por defecto	Gris oscuro
		Color derecha	Por defecto	Gris claro
		Posición pulgar	30	127
BotónTomarFoto	Botón	Color de fondo	Por defecto	Gris
		Negrita	(ninguno)	✓
		Texto	Texto para botón	Tomar foto
		Color de texto	Por defecto	Blanco

Valores diferentes a los asignados por defecto

<sup>1</sup> En App Inventor no es posible renombrar el componente que representa la pantalla de la aplicación.

<sup>2</sup> Los valores de los restantes atributos coinciden con los del deslizador rojo.

<sup>3</sup> Los valores de los restantes atributos coinciden con los del deslizador rojo.

<sup>4</sup> Los valores de los restantes atributos coinciden con los del deslizador rojo.

Les recordamos a los estudiantes que, antes de incorporar componentes, se puede ejecutar el emulador. Luego, a medida que agreguen elementos o cambien sus propiedades, las modificaciones se verán reflejadas en el emulador sin necesidad de interrumpirlo y volver a correrlo.

### Consigna 2: el círculo rojo

Una vez que todos hayan compuesto la interfaz gráfica, los invitamos a que resuelvan la segunda consigna. El objetivo es que, al mover el deslizador rojo, se imprima sobre la imagen un círculo rojo que cubra todo el lienzo.

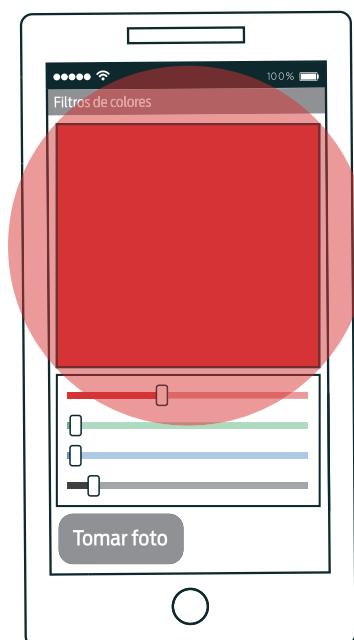
Lo primero que deben notar los estudiantes es que, al mover el deslizador rojo, se produce un evento que se puede manejar usando el bloque `cuando Deslizador.PosiciónCambiada [posiciónDelPulgar] / ejecutar { }`.

Para dibujar un círculo sobre el lienzo se utilizará el comando `Lienzo.DibujarCírculo centerX [ ] centerY [ ] radius [ ] fill [ ]`.

Probablemente, para entender qué valores son pertinentes para `centerX`, `centerY`, `radius` y `fill` los estudiantes requieran nuestra orientación.

```
cuando DeslizadorRojo .PosiciónCambiada
  posiciónDelPulgar
  ejecutar
```

Bloque para manejar evento de un deslizador



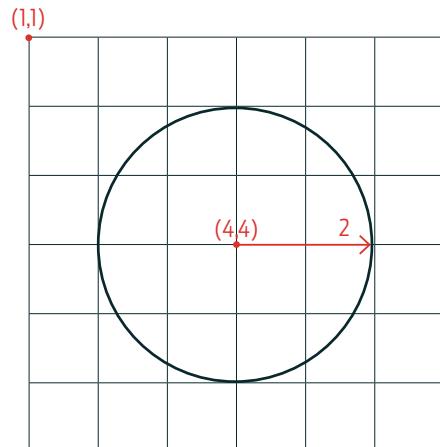
El círculo cubre un lienzo

```
llamar Lienzo .DibujarCírculo
  centerX
  centerY
  radius
  fill
```

Comando para dibujar un círculo en un lienzo

Un círculo en un plano puede definirse con un punto (que representa su centro) y un radio (la distancia entre el centro y todos los puntos del perímetro). Por ejemplo, en la imagen podemos observar un círculo en un lienzo cuyo centro es el punto (4,4) y su radio es 2.

En el problema propuesto, lo más natural es posicionar el centro del círculo en el centro del lienzo. Las coordenadas del centro del paño pueden calcularse dividiendo por 2 el alto y el ancho del lienzo, usando el bloque de división disponible en *Bloques > Integrados > Matemáticas*.

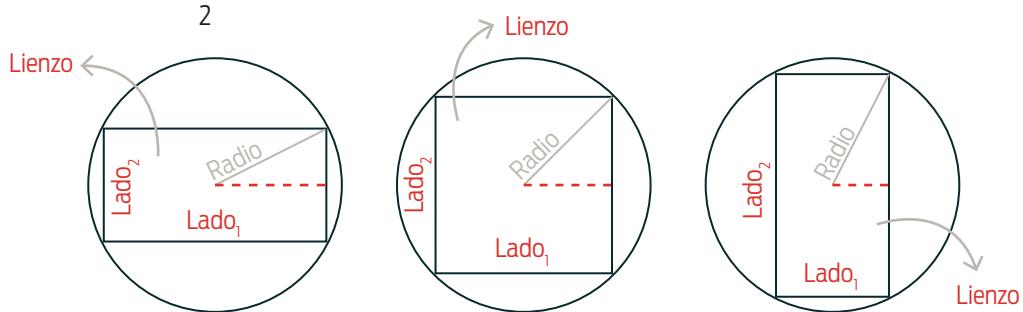


Círculo en un lienzo



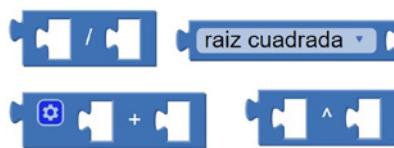
Valores del centro del lienzo

Para definir el valor adecuado del radio hay que usar un teorema que probablemente los estudiantes no conocen o no tengan presente: el teorema de Pitágoras. Este demuestra que en todo triángulo rectángulo –i.e., con un ángulo de  $90^\circ$ – la suma del cuadrado de los catetos es igual al cuadrado de la hipotenusa. Aun cuando en la ficha hay una breve descripción, es conveniente reproducir en el pizarrón el gráfico a continuación que, a partir del teorema, permite observar que, para cubrir completamente un rectángulo –en este caso, el lienzo– con un círculo, el radio del círculo debe ser igual a la raíz cuadrada de la suma de los cuadrados de sus lados divididos:

$$\text{radio} = \sqrt{\frac{\text{lado}_1^2 + \text{lado}_2^2}{2}}$$


Se cubren rectángulos con círculos

En *Bloques > Integrados > Matemáticas* hay un conjunto de bloques que permiten realizar las operaciones que hacen falta: dividir, calcular una raíz cuadrada, sumar y calcular una potencia.<sup>1</sup>



Bloques de *Bloques > Integrados > Matemáticas*

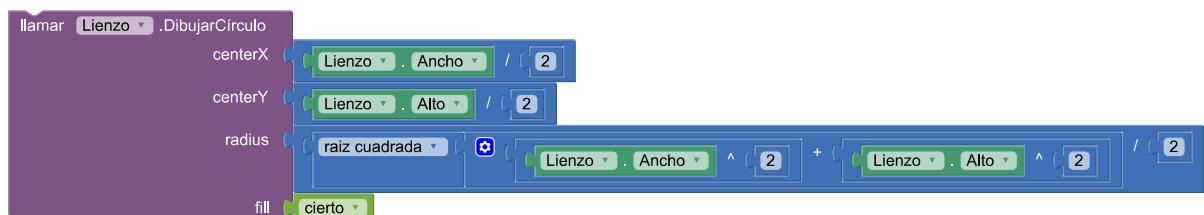
<sup>1</sup>El símbolo '^' representa la potencia. Por ejemplo, cuatro al cubo, se representa como  $4^3$ .

Combinando estos bloques adecuadamente se obtiene el valor del radio del círculo que hay que dibujar.



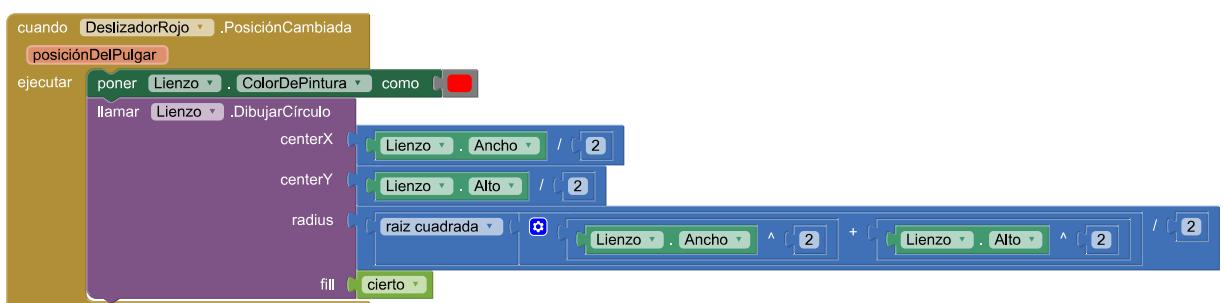
Cálculo del radio del círculo

App Inventor trabaja con distintos tipos de datos. Hasta ahora hemos usado números y colores, pero no son los únicos. Un tipo de datos particularmente relevante en computación es el de los **booleanos**, cuyos valores posibles son verdadero (en App Inventor llamado *cierto*) y falso, y sirven para realizar operaciones lógicas.<sup>1</sup> Al invocar al comando `Lienzo.DibujarCírculo centerX [ ] centerY [ ] radius [ ] fill [ ]`, el último argumento tiene que ser un booleano: en caso de usar `falso`, en el lienzo se dibuja únicamente el perímetro del círculo (que no sería visible porque el perímetro del círculo queda fuera de los límites del lienzo); en cambio, si se utiliza `cierto`, el interior del círculo aparecerá pintado. Los bloques que representan valores y operaciones booleanas son verdes y se encuentran en *Bloques > Integrados > Lógica*.



Bloques para dibujar el círculo

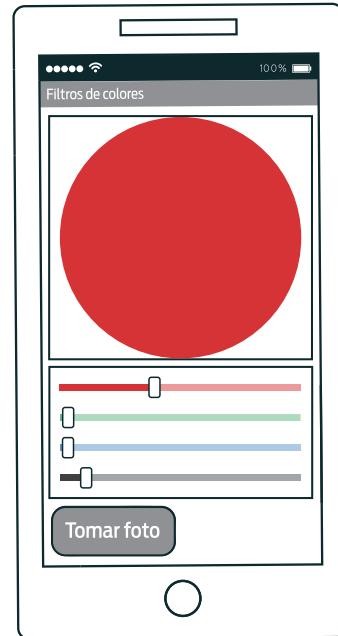
Finalmente, para que el círculo sea de color rojo –tal como pide la consigna– antes de dibujarlo hay que establecer `Lienzo.ColorDePintura` como `Rojo`. A continuación, se exhibe la solución del desafío:



Solución de la consigna 2

<sup>1</sup>En el capítulo 4, “Alternativas, repeticiones y variables”, se profundiza sobre el tipo booleano.

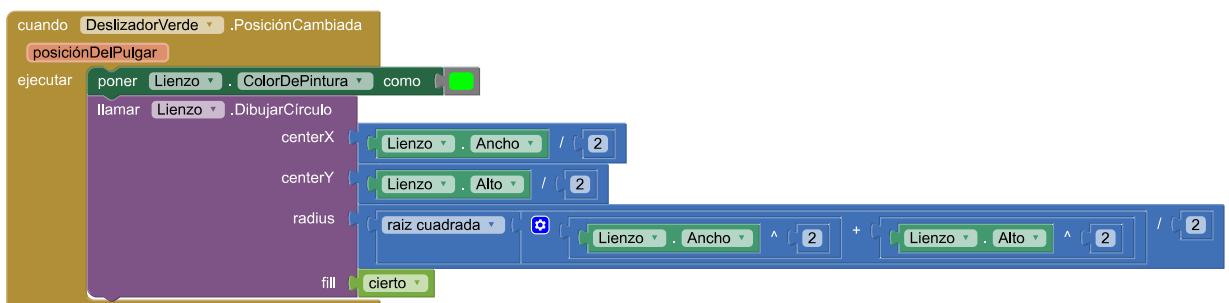
Podría ocurrir que algún estudiante proponga usar como radio el alto (o el ancho) del lienzo dividido 2. En este caso, el círculo no cubrirá completamente el paño.

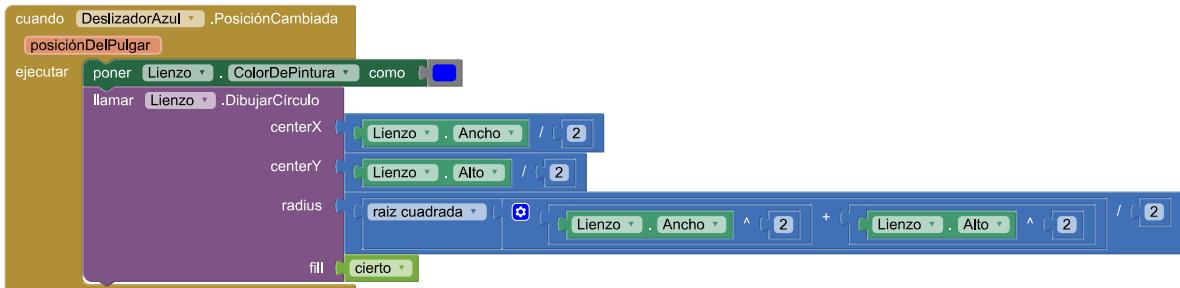


El círculo no cubre la totalidad del lienzo

### Consigna 3: el círculo verde y el círculo azul

Luego de que todos los estudiantes hayan conseguido dibujar un círculo rojo que cubra todo el lienzo, les indicamos que resuelvan la tercera consigna, muy similar a la anterior. En este caso, al mover el deslizador verde se tiene que dibujar un círculo verde y, al mover el azul, uno azul. Para resolver el desafío alcanza con incorporar al programa los bloques que se muestran a continuación. Mediante ellos, al manejar el evento que produce cada deslizador, se establece el color de pintura del lienzo que corresponde como paso previo a dibujar el círculo. De este modo, cada vez que se toque un deslizador diferente, también cambiará el color que observamos sobre el paño.



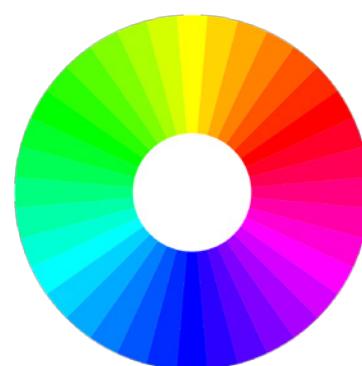


Solución de la consigna 3

#### Consigna 4: colores en RGB

Para introducir la cuarta consigna preguntamos: “¿Saben cómo se forman las imágenes en las pantallas de los dispositivos digitales?”. Escuchamos sus respuestas y, en caso de que no surja ninguna, les explicamos: “Las imágenes digitales están formadas por una cuadrícula de puntos muy pequeños, cada uno de un color. Debido a que se encuentran muy juntitos, al mirar una pantalla nuestro cerebro compone la imagen como si se tratara de un todo continuo. Técnicamente, cada punto se llama *píxel* y es la menor unidad homogénea en color que forma parte de una imagen digital. Si agrandamos mucho una imagen, notaremos cada uno de sus píxeles”.

Continuamos: “¿Alguna vez mezclaron témperas para crear nuevos colores? El rojo, el amarillo y el azul son los colores primarios; si los mezclamos, podemos formar otros colores. ¿Qué pasa si mezclamos azul y amarillo? Obtenemos el verde, ¿no? ¿Y rojo con amarillo?”. Prestamos atención a sus observaciones y seguimos: “El color que adquiere un píxel en una pantalla también se basa en la mezcla de tres colores. El punto de partida es el negro, y va adquiriendo color a través de tres luces minúsculas: una roja, una verde y una azul. A este conjunto de colores se los llama RGB por sus siglas en inglés: *Red* (rojo), *Green* (verde) y *Blue* (azul). Al aumentar y disminuir la cantidad de luz que emite cada una de estas pequeñas luces, se producen distintos colores. Como cada luz admite 256 intensidades diferentes, si hacen las cuentas van a ver que con este sistema se pueden formar más de 16 millones de colores!”.<sup>1</sup> Concluimos: “La elección de los valores 0 y 255 como mínimo y máximo que pusimos en los deslizadores de colores no fue arbitraria: cada posición representa una intensidad de luz diferente para cada color”.



Colores RGB

<sup>1</sup> La representación digital de imágenes se trabaja en profundidad en el capítulo 5, “Representación de la información”.

Invitamos a los estudiantes, entonces, a que resuelvan la cuarta consigna. El objetivo es que, cuando se dibujen círculos, el color se corresponda con el RGB codificado por las posiciones de los selectores de los deslizadores rojo, verde y azul.

Si observan el menú *Bloques > Integrados > Colores* notarán que, además de los bloques para algunos colores predeterminados, hay uno para definir un color a partir de tres números. El primero corresponde a la intensidad de la luz roja del píxel, el segundo a la intensidad de la luz verde y el tercero a la intensidad de la luz azul.



Bloquea para crear colores RGB

A continuación, puede observarse una posible propuesta de los estudiantes, en la que el manejo de los distintos eventos producidos por cambios en los deslizadores de los colores se realiza de idéntica forma: el color del círculo se establece usando la posición de los selectores, que puede obtenerse con la propiedad `Deslizador.PosiciónDelPulgar`.



Possible propuesta de los estudiantes a la consigna 4

Aun cuando un programa sea funcionalmente correcto –como es el caso del ejemplo anterior–, la presencia de porciones idénticas da la pauta de que podemos modificarlo para obtener otro equivalente de mejor calidad que evite la redundancia. Instamos a los estudiantes que lo hayan planteado de esta manera a que analicen sus propuestas, exploren el entorno buscando nuevas herramientas y piensen en otras alternativas.

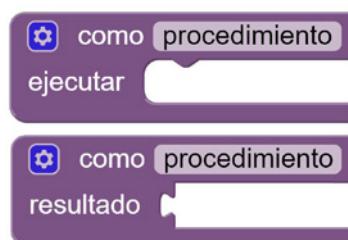
En esta instancia, se retomará el concepto de procedimiento, presentado en la actividad anterior. Así, se les podrá indicar a los estudiantes que, en lugar de copiar varias veces el mismo conjunto de bloques, se puede crear un procedimiento.

En programación, un procedimiento encapsula un conjunto de instrucciones para realizar una tarea específica. Permite, por lo tanto, descomponer un problema complejo en piezas más simples que, al combinarlas, resuelven el problema original. Además, al crearlo, se le asigna un nombre que describa su propósito. Luego, puede ser invocado en el programa como cualquier otra instrucción.

En el entorno de MIT App Inventor 2, los bloques para crear procedimientos están en *Bloques > Integrados > Procedimientos*. Allí hay dos disponibles:

`como (procedimiento) / ejecutar { }` y  
`como (procedimiento) / resultado [ ]`.

Mientras que el primero se utiliza para realizar una tarea, el segundo, habitualmente llamado *función*, se usa para calcular y retornar un valor.<sup>1</sup>

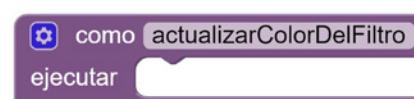


Bloques para crear procedimientos

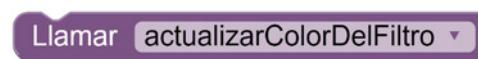
En el problema planteado no queremos calcular un valor; solo trasladaremos una secuencia de bloques repetida a un único lugar (*i.e.*, a un procedimiento). Por lo tanto, nos valdremos de `como (procedimiento) / ejecutar { }`.

Por defecto, el nombre es *procedimiento* y, para cambiarlo, alcanza con hacer clic allí y modificarlo. En esta actividad, un nombre adecuado podría ser *actualizarColorDelFiltro*.

Al crear un procedimiento, App Inventor genera automáticamente un bloque para que se lo pueda invocar cada vez que sea pertinente, disponible en *Bloques > Integrados > Procedimientos*.



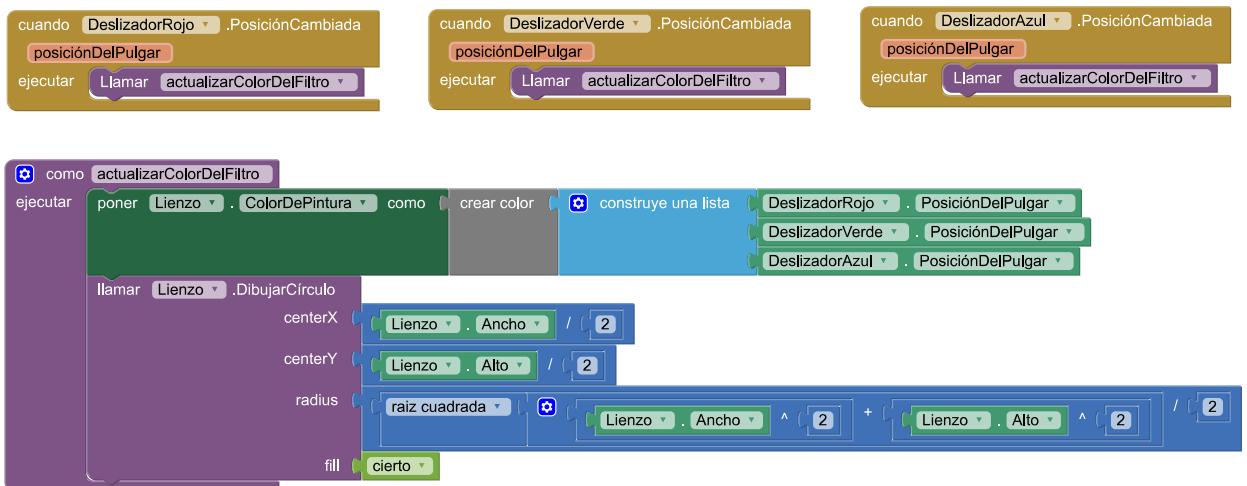
Cambio de nombre del procedimiento



Bloque para usar el procedimiento

<sup>1</sup> Las funciones se presentan en el capítulo 4, “Alternativas, repeticiones y variables”.

A continuación, se presenta una solución adecuada para la consigna. Ahora, sin porciones redundantes en el programa, al mover los deslizadores se generarán todos los colores RGB.



Solución con un procedimiento

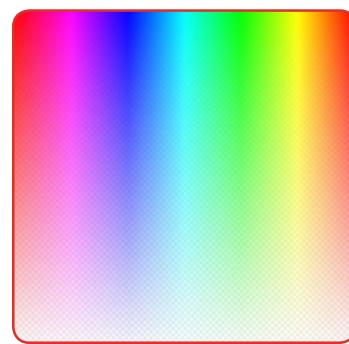
Una vez que los estudiantes hayan podido resolver la consigna definiendo un procedimiento, enfatizamos las ventajas de esta solución: “Recién hemos hecho lo que en programación se conoce como **refactorización**. En forma general, al refactorizar un programa transformamos (y mejoramos!) su estructura interna, sin modificar su comportamiento. En este caso, creamos un procedimiento para evitar repetir un mismo conjunto de instrucciones en distintas partes del programa. El procedimiento, al que le pusimos un nombre que describe su propósito, encapsula una tarea específica: dibujar el círculo con el color adecuado. Ahora, al leer nuestro programa, es más fácil interpretar qué es lo que hace”.

Luego, preguntamos: “¿Se les ocurre otra ventaja de usar procedimientos de este modo?”. Escuchamos las observaciones de los estudiantes y continuamos: “Imaginen que quisieramos modificar el programa para que, en lugar de que al desplazar los deslizadores cambie el color del círculo, usásemos los valores para modificar el color del botón. En el programa original, ¿en cuántos lugares deberíamos haber realizado cambios? ¿Y en la nueva versión?”. Mientras que en la versión original habría que haber cambiado el programa en tres lugares, en la nueva versión hay que hacerlo en uno solo: en el procedimiento. “¡Imagínense qué pasaría si en lugar de tres tuviésemos cien deslizadores!”.

### Consigna 5: transparencia y opacidad

Hasta aquí solo se hemos trabajado sobre el color del círculo que dibujamos en el lienzo. Sin embargo, el objetivo final del proyecto es que con la aplicación podamos aplicar filtros sobre imágenes para lograr un efecto parecido al que observamos al mirar a través de un papel celofán de color. Para conseguirlo, deberemos generar dos planos superpuestos: en el trasero estará la imagen y, encima, un círculo de color para teñirla. Sin embargo, hasta ahora hemos trabajado con colores completamente opacos, que no permiten ver qué hay detrás.

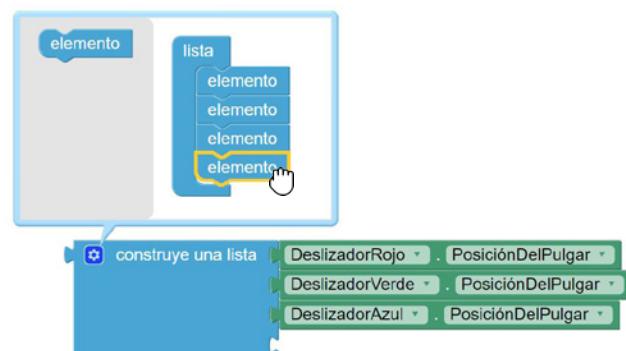
En computación gráfica, hay un proceso llamado **composición alpha** que consiste en combinar una imagen con un fondo para generar la apariencia de transparencia. En nuestra aplicación, la composición alpha nos permitirá graduar el nivel de opacidad –o, inversamente, de transparencia– del círculo que terminemos dibujando sobre la fotografía. El objetivo de esta consigna es incorporar niveles de transparencia a los círculos que dibujamos sobre el lienzo.



Composición alpha

En el programa, para establecer el color de pintura del lienzo utilizamos el bloque `crear color [ ]`, al que le proporcionamos tres valores que representan las componentes rojo, verde y azul del color con el que dibujamos el círculo. Sin embargo, a `crear color [ ]`, también se le pueden proveer cuatro valores, en los que el cuarto representa un valor *alpha*: un número entre 0 y 255 de una escala donde el 0 es la transparencia total y el 255 la completa opacidad.

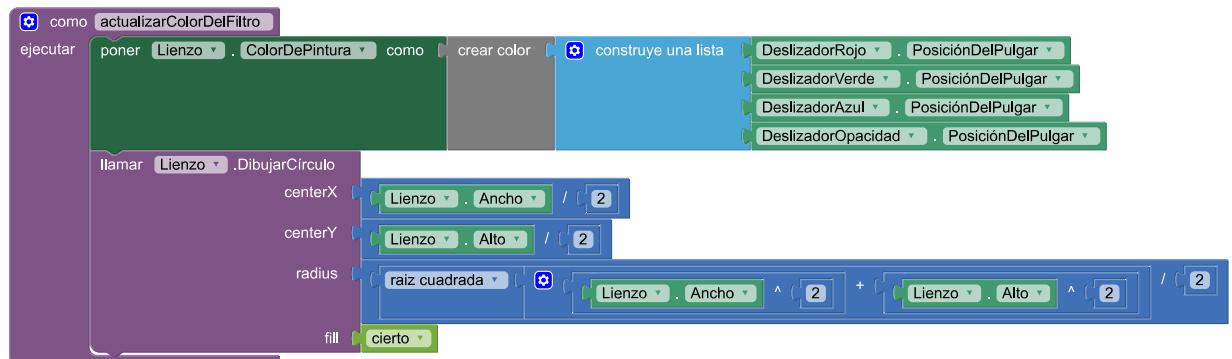
Para poder agregar un elemento a la lista hay que presionar la tuerca de la esquina superior izquierda del bloque `construye una lista [ ] [ ] [ ]` y encastrear un nuevo bloque `elemento` dentro de `lista { . . . }`. Entonces, aparecerá un espacio para agregar un nuevo valor.



Se agrega un elemento a una lista

De este modo, puede modificarse `actualizarColorDelFiltro` para incorporar el nivel de opacidad del color de la pintura del lienzo. Además, debe establecerse qué hace el programa frente al evento que se produce cuando se mueve el deslizador de opacidad; también, en este caso, actualizaremos el color del filtro.

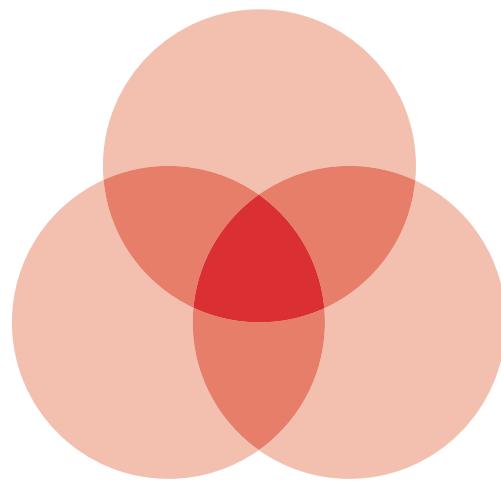
```
cuando [DeslizadorOpacidad .] .PosiciónCambiada
    posiciónDelPulgar
ejecutar Llamar [actualizarColorDelFiltro]
```



Incorporación al programa del nivel de opacidad de color

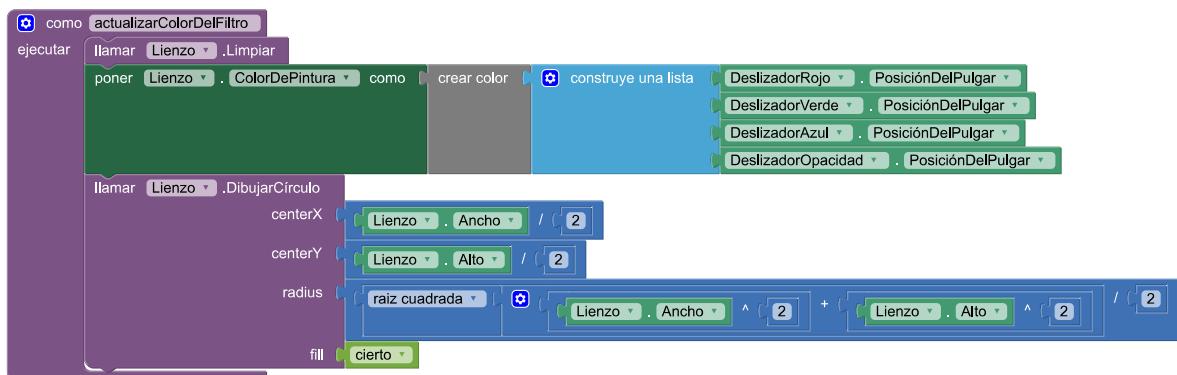
Al correr el programa, los estudiantes notarán que, si bien pueden generar los distintos colores al mover los deslizadores rojo, verde y azul, no podrán generar niveles altos de transparencia, aun ubicando la perilla de opacidad en el extremo izquierdo –que debería dejar el paño completamente blanco–. Lo que sucede es que cada vez que se ejecuta `Lienzo.`

```
DibujarCírculo (centerX) []
(centerY) [] (radius) [] (fill) []
se dibuja un nuevo círculo, sin borrar los que ya se habían dibujado. Las sucesivas superposiciones opacan el color del lienzo, como si diéramos varias manos de pintura sobre una pared, e impiden generar la ilusión de transparencia.
```



Círculos superpuestos

El problema puede resolverse limpiando el lienzo justo antes de dibujar un nuevo círculo. Utilizamos, para ello, `Lienzo.Limpiar`. A continuación, puede observarse corregido el procedimiento `actualizarColorDelFiltro`.



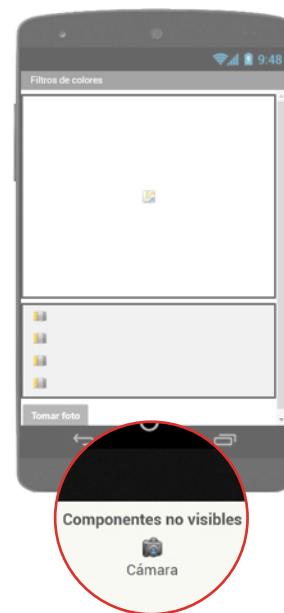
Se limpia el lienzo

#### Consigna 6: incorporación de la cámara

Llegados a este punto, les preguntamos a los estudiantes: “¿Qué sucede cuando presionamos el botón para tomar una foto?”. Es esperable que algún estudiante conteste que no sucede nada. Entonces, les indicamos que continúen con la sexta consigna. Allí se pide que, al apretar el botón, se active la cámara del teléfono y, luego de tomar una fotografía, esta aparezca como fondo del lienzo.

En primer lugar, hay que agregar un componente no visible: la cámara, que se encuentra disponible en *Paleta > Medios*. Por tratarse de un elemento que no se muestra en la pantalla, al agregarlo en el editor de diseño, aparecerá debajo del visor.

Luego, desde el editor de bloques, se puede establecer que, al hacer clic sobre el botón *Tomar foto*, se invoque al comando `Cámara.TomarFoto`. De este modo, se activará la cámara para poder sacar una fotografía.



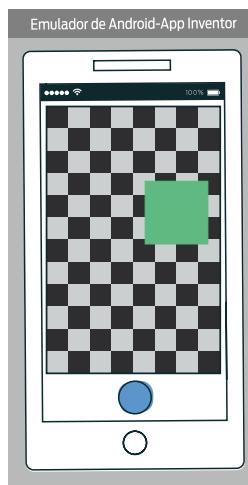
La cámara no es visible



Activación de la cámara

### LA CÁMARA Y EL EMULADOR

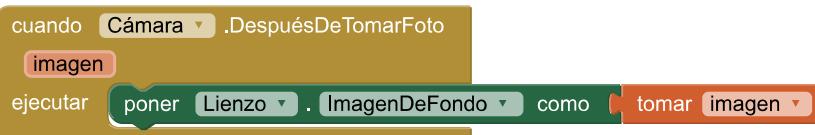
Mientras que con los teléfonos se pueden tomar fotografías, al usar el emulador no contamos con una cámara fotográfica. En este caso, veremos aparecer en la pantalla un cuadrado que se mueve sobre un damero gris y negro. Allí, al "realizar la toma", el programa usará como foto la imagen congelada de lo que se muestra en el visor al momento del "disparo". A efectos del desarrollo esto no es un problema, aunque si se quisiese ver la cámara en acción habría que ejecutar la aplicación en el teléfono.



Al tomar una fotografía, se produce un evento que podemos manejar con el bloque **cuando Cámara.**

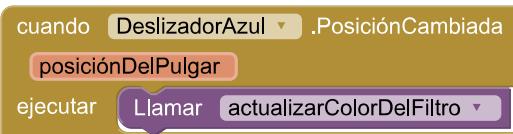
**DespuésDeTomarFoto (imagen) / ejecutar { }**, disponible en *Bloques > Screen1 > Cámara*.

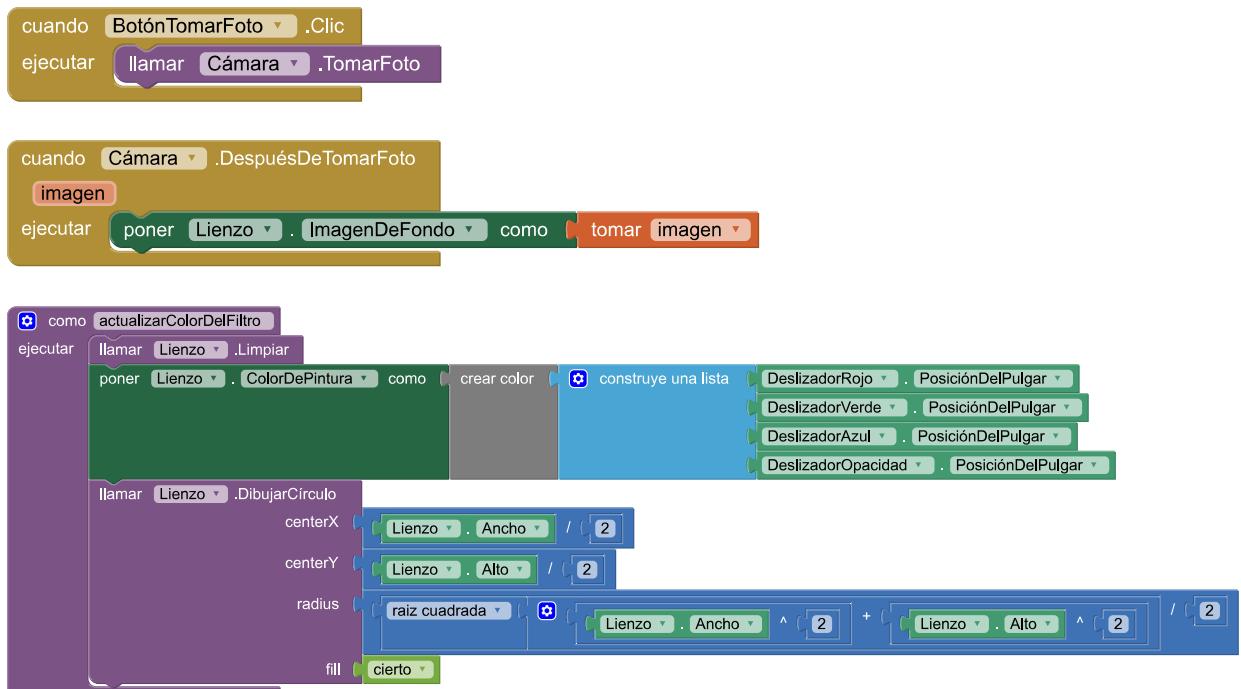
Una vez concluida la toma, podemos, entonces, incorporar la imagen como fondo del lienzo.



Incorporación de la imagen como fondo del lienzo

Se exhibe a continuación una solución completa de la actividad.





Solución completa de “Filtros de colores”

Luego de una puesta en común, invitamos a los estudiantes a que prueben la aplicación en sus teléfonos.

## CIERRE

Para finalizar, reflexionamos con los estudiantes sobre las posibilidades que nos brinda el uso de procedimientos. En primer lugar, resultan útiles para no repetir porciones idénticas de un programa. Actualizar el color del lienzo constituye una unidad de sentido dentro de los desafíos propuestos, y por eso resultó conveniente usar un procedimiento cuyo propósito fuera exclusivamente ese. Por otro lado, si hubiese que modificar algo de lo que hay que hacer cuando se mueven los deslizadores, localizamos los cambios en un único lugar en vez de desparramarlos por distintos lados del programa. La ventaja de hacerlo de este modo sería aún más evidente si, por ejemplo, tuviéramos que manejar los eventos de cien componentes en lugar de cuatro.

NOMBRE Y APELLIDO:

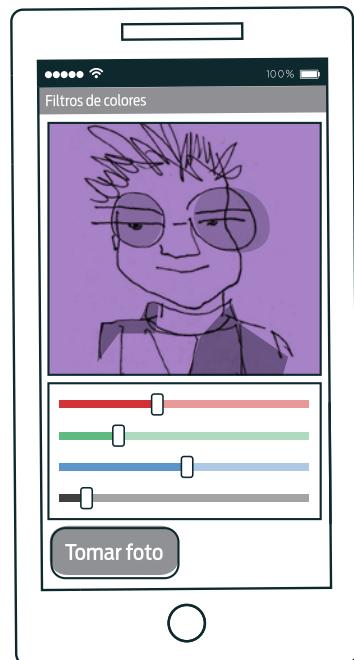
CURSO:

FECHA:

# FILTROS DE COLORES

Llegó la hora de que programes una aplicación con la que sacar fotos para aplicarles filtros de colores. Aunque parezca una broma, vas a poder aplicar filtros usando... ¡más de 16 millones de colores!

1. Empezá por la interfaz gráfica. En la imagen que sigue están los componentes y sus dimensiones. En la tabla, las propiedades que tenés que cambiar para que se vea igual a la que te mostramos. Seguí las indicaciones para componerla.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1 <sup>1</sup>	Pantalla	Título	Screen	Filtros de colores
Lienzo	Lienzo	Alto	Automático	60%
		Ancho	Automático	Ajustar al contenedor
PanelDeslizadores	Disposición horizontal	Disp. horizontal	Izquierda	Centro
		Disp. vertical	Arriba	Centro
		Ancho	Automático	Ajustar al contenedor
DeslizadorRojo	Deslizador	Color izquierda	Por defecto	Rojo
		Color derecha	Por defecto	Rojo
		Ancho	Automático	90%
		Valor máximo	50	255
		Valor mínimo	10	0
		Posición del pulgar	30	0
DeslizadorVerde <sup>2</sup>	Deslizador	Color izquierda	Por defecto	Verde
		Color derecha	Por defecto	Verde
DeslizadorAzul <sup>3</sup>	Deslizador	Color izquierda	Por defecto	Azul
		Color derecha	Por defecto	Azul
DeslizadorOpacidad <sup>4</sup>	Deslizador	Color izquierda	Por defecto	Gris oscuro
		Color derecha	Por defecto	Gris claro
		Posición pulgar	30	127
BotónTomarFoto	Botón	Color de fondo	Por defecto	Gris
		Negrita	(ninguno)	✓
		Texto	Texto para botón	Tomar foto
		Color de texto	Por defecto	Blanco

<sup>1</sup> En App Inventor no es posible renombrar el componente que representa la pantalla de la aplicación.

<sup>2</sup> Los valores de los restantes atributos coinciden con los del deslizador rojo.

<sup>3</sup> Los valores de los restantes atributos coinciden con los del deslizador rojo.

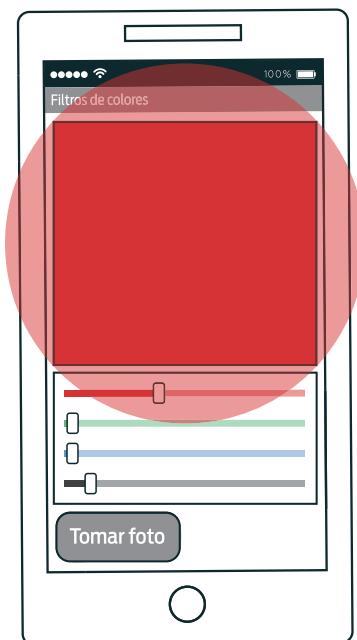
<sup>4</sup> Los valores de los restantes atributos coinciden con los del deslizador rojo.

NOMBRE Y APELLIDO:

CURSO:

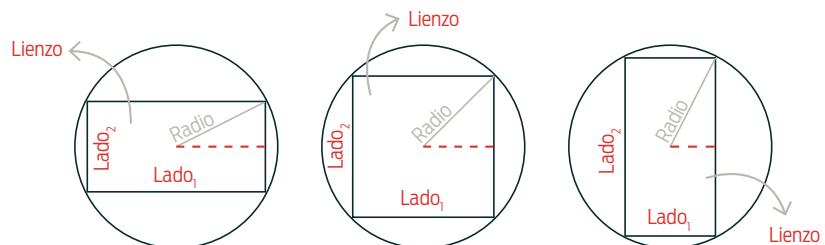
FECHA:

2. Ahora tenés que conseguir que cuando se mueva el deslizador rojo se imprima un círculo rojo que cubra todo el lienzo.

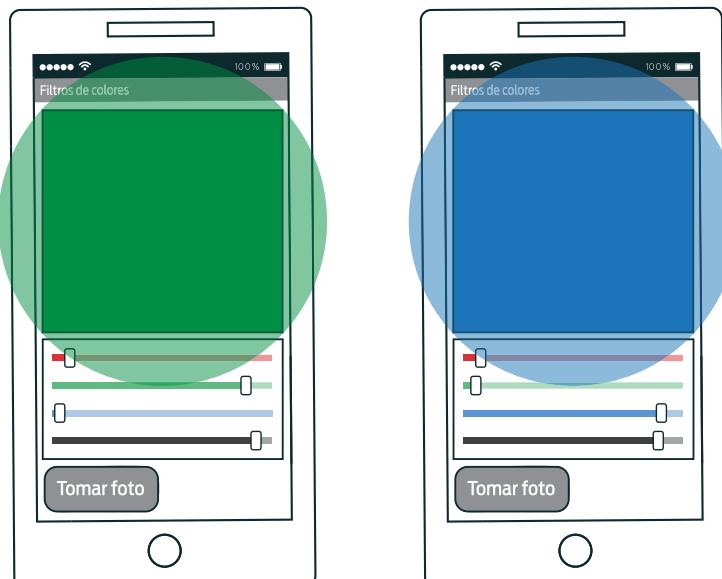
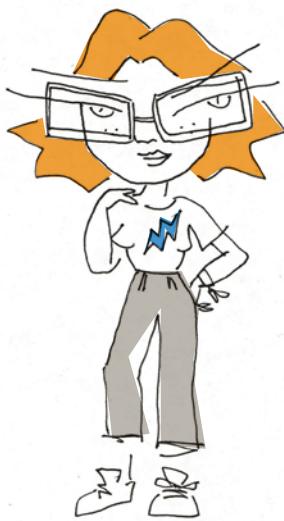


### ¡PITÁGORAS AL RESCATE!

Un lienzo es un rectángulo y, *a priori*, no conocemos sus dimensiones. ¿Cómo lo cubrimos con un círculo? Por suerte Pitágoras, matemático y filósofo griego, demostró en el siglo VI a.C. un teorema que nos da la respuesta. Para cubrir completamente un rectángulo con un círculo, si situamos el centro del círculo en el centro del rectángulo, su radio debe ser igual a la raíz cuadrada de la suma de los cuadrados de sus lados dividido 2:  $\text{radio} = \sqrt{\frac{\text{lado}_1^2 + \text{lado}_2^2}{2}}$ .



3. Seguí adelante. Ahora, al mover el deslizador verde, dibujá un círculo verde; y al mover el azul, uno azul.



3. Tres colores es muy poco. No te conformes con menos de ¡16 millones! Encárgate de que el color del círculo esté determinado por la posición de las perillas de los deslizadores rojo, verde y azul.

### PISTA

Investigá el bloque `crear color [ ]`.



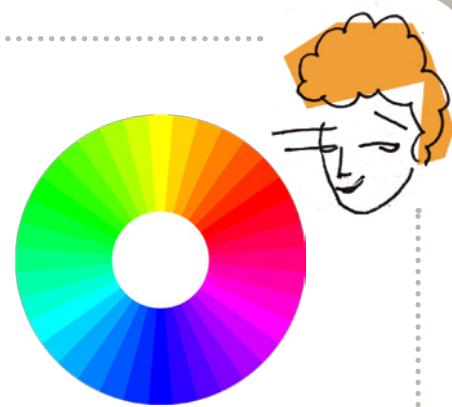
NOMBRE Y APELLIDO:

CURSO:

FECHA:

### COLORES EN RGB

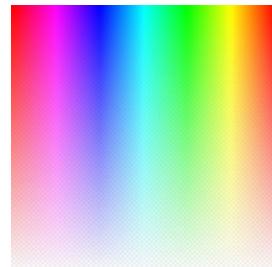
Las imágenes digitales están formadas por una cuadrícula de puntos muy pequeños llamados píxeles, cada uno de un color. El color que adquiere un píxel se basa en la mezcla de tres colores. El punto de partida es el negro, que va adquiriendo color a través de tres luces minúsculas: una roja, una verde y una azul. A este conjunto de colores se los llama RGB por sus siglas en inglés: Red (rojo), Green (verde) y Blue (azul). Al aumentar y disminuir la cantidad de luz que emite cada una de estas pequeñas luces se producen distintos colores. Como cada luz admite 256 intensidades diferentes, si hacés las cuentas vas a ver que con este sistema se pueden formar ¡más de 16 millones de colores!



5. Hasta acá solo trabajaste sobre el color del círculo que dibujaste en el lienzo. Sin embargo, el objetivo final es que podamos aplicar filtros sobre fotos para lograr un efecto parecido al que observamos al mirar a través de un papel celofán de color. Llegó el momento de trabajar sobre la transparencia.

### COMPOSICIÓN ALPHA

En computación gráfica, hay un proceso llamado **composición alpha** que consiste en combinar una imagen con un fondo para generar la apariencia de transparencia. En tu aplicación, la composición alpha te permitirá graduar el nivel de opacidad –o, inversamente, de transparencia– del círculo que termines dibujando sobre la fotografía.



En App Inventor, para crear un color se puede usar un cuarto parámetro. ¿Para vos de qué se trata? Fijate lo que podés hacer presionando la tuerca de arriba a la izquierda del bloque **construye una lista [ ] [ ] [ ]**.



¡Atención! Al dibujar un círculo, tenés que borrar el que había antes. ¿Por qué esto es importante?

---

---

---

---

6. Con lo hecho, ¿qué sucede cuando haces clic sobre el botón *Tomar foto*? Bueno, ocúpate de eso ahora.

---

---

---



## Secuencia Didáctica 2

# PARÁMETROS

Muchas veces los procedimientos que escribimos son muy parecidos y solo difieren en algún valor. Un parámetro es un “agujero” que se deja en un procedimiento que se completa con un valor cuando el procedimiento es invocado. De esta forma, podemos diseñar soluciones generales que, luego, resuelven problemas particulares.

A lo largo de esta secuencia didáctica se presentarán los motivos por los cuales se utilizan procedimientos con parámetros y en qué casos conviene hacerlo.

.....  
**OBJETIVOS**

- Introducir las nociones de parámetro y argumento.
  - Producir soluciones generales que resuelvan diversos problemas particulares.
- .....

## Actividad 1

### Elefantes que se balancean



TODA LA CLASE

#### OBJETIVO

- Presentar las nociones de parámetro y argumento

#### MATERIALES

- Papel
- Lápiz
- Reproductor de audio

## DESARROLLO

El objetivo de esta actividad es presentar las nociones de parámetro y argumento. Comenzamos escuchando con toda la clase la canción *Un elefante se balanceaba*.<sup>1</sup> A continuación, les pedimos a los estudiantes que transcriban en un papel las cuatro primeras estrofas de la manera más concisa posible.

Después de que ensayan posibles escrituras, hacemos una puesta en común. Preguntamos: “¿Cómo transcribieron la canción? ¿Escribieron cada estrofa por separado?”. Si bien son muy parecidas, las estrofas no son iguales. En cada una, la cantidad de elefantes que se balancean es distinta. Escribimos la canción en el pizarrón y subrayamos las diferencias de la segunda estrofa en adelante.<sup>2</sup>

 Un elefante se balanceaba sobre la tela de una araña. Como veía que resistía fue a llamar a otro elefante.	 Tres elefantes se balanceaban sobre la tela de una araña. Como veían que resistía fueron a llamar a otro elefante.
 Dos elefantes se balanceaban sobre la tela de una araña. Como veían que resistía fueron a llamar a otro elefante.	 Cuatro elefantes se balanceaban sobre la tela de una araña. Como veían que resistía fueron a llamar a otro elefante.

Estrofas de *Un elefante se balanceaba*

A continuación, borramos las palabras subrayadas para que noten que, sin ellas, las estrofas son exactamente iguales.

 Un elefante se balanceaba sobre la tela de una araña. Como veía que resistía fue a llamar a otro elefante.	 ___ elefantes se balanceaban sobre la tela de una araña. Como veían que resistía fueron a llamar a otro elefante.
 ___ elefantes se balanceaban sobre la tela de una araña. Como veían que resistía fueron a llamar a otro elefante.	 ___ elefantes se balanceaban sobre la tela de una araña. Como veían que resistía fueron a llamar a otro elefante.

Segunda, tercera y cuarta estrofa de la canción sin las cantidades de elefantes

<sup>1</sup> Un videoclip animado de la canción puede verse en <https://goo.gl/MhkGL>.

<sup>2</sup> Se deja fuera de la comparación entre estrofas la primera porque es la única con el sujeto y el predicado en singular (*elefante* en vez de *elefantes*, por ejemplo).

Luego, dejando de lado la primera estrofa, hacemos notar que, al eliminar la mención al número de elefantes, las estrofas siguientes funcionan como una suerte de estribillo. Por lo tanto, pueden quitarse dos de estas estrofas y transformar la restante en un estribillo con un “agujero” o “hueco” en el lugar donde aparecía antes el número de elefantes. A este agujero le ponemos un nombre que describa cuál es el dato faltante; en este caso, podría llamarse, *cantidad*. De este modo, queda definido un estribillo que admite distintas cantidades de elefantes. Por último, escribimos la canción, remitiendo el estribillo así formulado y completando el agujero con el valor que corresponda en cada caso. Señalamos que el agujero se denomina **parámetro** y el valor usado, **argumento**. En la canción, *cantidad* es el nombre del parámetro y *Dos*, *Tres* y *Cuatro*, argumentos.

The diagram illustrates the structure of a song. It features a green rectangular box containing two columns of text. The left column is labeled "Estríbillo (cantidad):" and contains the lyrics: "cantidad elefantes se balanceaban sobre la tela de una araña. Como veían que resistía fueron a llamar a otro elefante." The right column is labeled "Canción:" and contains the lyrics: "Un elefante se balanceaba sobre la tela de una araña. Como veía que resistía fue a llamar a otro elefante." Below the lyrics, three options are listed in brackets: "[ Estríbillo Dos ]", "[ Estríbillo Tres ]", and "[ Estríbillo Cuatro ]". Above the green box, there are two small circles with the numbers "10" and "11" respectively.

El estribillo se escribe solo una vez

## CIERRE

Reflexionamos con los estudiantes sobre la utilidad de los parámetros y los argumentos. En este caso, nos permitieron expresar la variación entre versos muy similares dejando un agujero (el parámetro), que luego completamos con un valor específico (el argumento). Al parámetro le pusimos un nombre descriptivo (*cantidad*) que se reemplaza por el valor del argumento correspondiente cada vez que se invoca el estribillo. En el ejemplo, *cantidad* se reemplaza por *Dos*, *Tres* y *Cuatro*. De este modo, el uso de parámetros y argumentos nos permitió escribir la canción de manera concisa.

## Actividad 2

### Tu arte pop

 DE A DOS

#### OBJETIVOS

- Descomponer problemas.
- Definir procedimientos con parámetros.
- Invocar procedimientos usando argumentos.

#### MATERIALES

- |   |                                 |
|---|---------------------------------|
|   | Computadora                     |
|  | Internet                        |
|  | MIT App Inventor 2              |
|  | Ficha para estudiantes          |
|  | Teléfono con Android (opcional) |

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes realicen una práctica de programación definiendo procedimientos que contengan parámetros. A medida que desarrollen el programa, aparecerán secuencias de instrucciones, si bien no idénticas, muy parecidas entre sí. Entonces, se les presentarán desafíos que los guiarán a construir soluciones más simples y generales mediante procedimientos con parámetros.<sup>1</sup>

#### Consigna 1: interfaz gráfica

Comenzamos preguntando: “¿Saben lo que es el arte pop?”. Escuchamos sus respuestas y les contamos: “El arte pop fue un movimiento artístico que surgió en el Reino Unido y los Estados Unidos en la década de 1950. Desafiando las tradiciones de las bellas artes, sus obras incorporaban elementos pertenecientes a la cultura popular, como publicidades de latas de tomates e historietas. Así como lo escuchan”. Continuamos: “Andy Warhol fue, probablemente, el referente más conocido del arte pop. ¿Nunca vieron el díptico de Marilyn Monroe?”.<sup>2</sup>

Les repartimos la ficha de la actividad a los estudiantes y les comentamos que van a desarrollar una aplicación con la que convertirán imágenes en obras pop. Les pedimos que resuelvan la primera consigna, en la que se dan instrucciones para armar la interfaz gráfica de la aplicación. La imagen que sigue muestra los componentes que hay que incorporar y sus dimensiones; y la tabla a continuación, los valores de las propiedades que hay que cambiar para que la aplicación se vea tal como en la imagen.

Es conveniente mencionarles a los estudiantes que el componente para seleccionar una imagen de la galería no es el botón utilizado en proyectos anteriores, sino de tipo *SelectorDeImagen*, que se encuentra disponible en *Paleta > Medios*.

<sup>1</sup> Puede accederse a una versión completa de la aplicación buscando “programar2020” en la galería de aplicaciones de App Inventor.

<sup>2</sup> Hay un museo exclusivamente dedicado a Warhol (<https://www.warhol.org/>), además de que sus obras forman parte de exposiciones de muchos de los museos más importantes del mundo.



Diseño de la interfaz gráfica

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen	Arte Pop
LienzoSuperior Izquierdo	Lienzo	Alto	Automático	40%
		Ancho	Automático	50%
LienzoSuperior Derecho	Lienzo	Alto	Automático	40%
		Ancho	Automático	50%
LienzoInferior Izquierdo	Lienzo	Alto	Automático	40%
		Ancho	Automático	50%

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
LienzoInferior Derecho	Lienzo	Alto	Automático	40%
		Ancho	Automático	50%
PanelDeBotones	Disposición horizontal	Disp. vertical	Arriba	Centro
		Alto	Automático	Ajustar al contenedor
		Ancho	Automático	Ajustar al contenedor
SelectorDelImagen DeGalería	Selector de imagen	Color de fondo	Automático	Gris
		Negrita	(ninguno)	✓
		Texto	Texto para botón	Galería
		Color de texto	Por defecto	Blanco
BotónTomarFoto	Botón	Color de fondo	Automático	Gris
		Negrita	(ninguno)	✓
		Texto	Texto para botón	Tomar foto
		Color de texto	Por defecto	Blanco

Valores diferentes a los asignados por defecto

### Consigna 2: Sacar la foto y ubicarla en los lienzos

Una vez que todos los estudiantes hayan dispuesto los componentes, los invitamos a que resuelvan la segunda consigna. Allí se pide que, al presionar el botón *Tomar foto*, se habilite la cámara del teléfono y, luego de tomar una foto, se coloque la imagen como fondo de cada uno de los cuatro lienzos.

Para resolver el desafío, en primer lugar hay que incluir la cámara como componente de la aplicación. Por tratarse de un elemento no visible, al agregarlo aparecerá debajo del visor. Es conveniente renombrarlo y llamarlo *Cámara*, en lugar de *Camara1*, el nombre que por defecto le asigna App Inventor. Si bien es un cambio sutil, la alternativa propuesta es más prolífica.



La cámara es un componente no visible

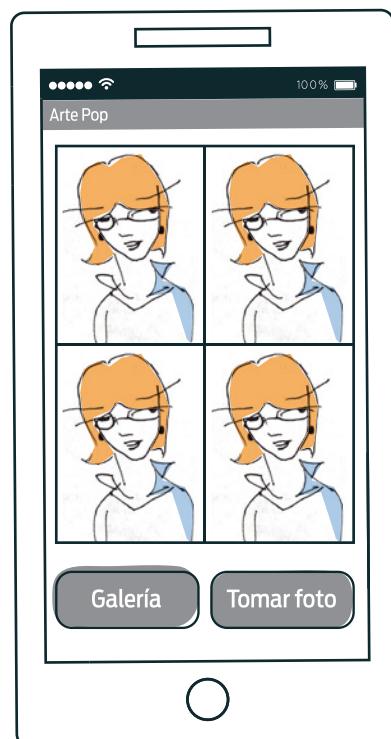
A grandes rasgos, la solución consiste en manejar dos eventos para establecer el comportamiento de la aplicación: (i) cuando se hace clic sobre **BotónTomarFoto**; y (ii) inmediatamente después de que se toma la fotografía. Para manejar el primero hay que usar el bloque **cuando Botón clic / ejecutar { }** y, allí, llamar al comando **Cámara.TomarFoto**.

```
cuando BotónTomarFoto .Clic
ejecutar llamar Cámara .TomarFoto
```

Activación de la cámara de fotos

Para manejar el evento que se produce luego de que se toma la fotografía hay que usar **cuando Cámara.DespuésDeTomarFoto (imagen) / ejecutar { }**. Una posible solución para ubicar las imágenes es establecer, para cada lienzo, la propiedad **Lienzo.ImagenDeFondo** usando el parámetro **imagen**.

```
cuando Cámara .DespuésDeTomarFoto
  imagen
ejecutar poner LienzoSuperiorIzquierdo . ImagenDeFondo como tomar imagen
  poner LienzoSuperiorDerecho . ImagenDeFondo como tomar imagen
  poner LienzoInferiorIzquierdo . ImagenDeFondo como tomar imagen
  poner LienzoInferiorDerecho . ImagenDeFondo como tomar imagen
```



Ubicación de la foto como fondo de cada lienzo

### Consigna 3: Seleccionar una foto de la galería y ubicarla en los lienzos

La tercera consigna propone que la imagen también pueda seleccionarse de la galería del teléfono.

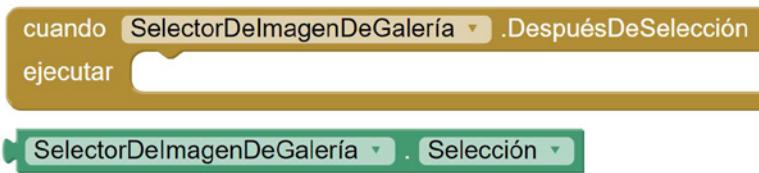
Como el aparente botón con la leyenda *Galería* es en realidad de tipo *SelectorDeImagen*, al presionarlo se abrirá la galería para seleccionar una imagen (esto es lo que hacen los componentes de este tipo).



Galería de fotos

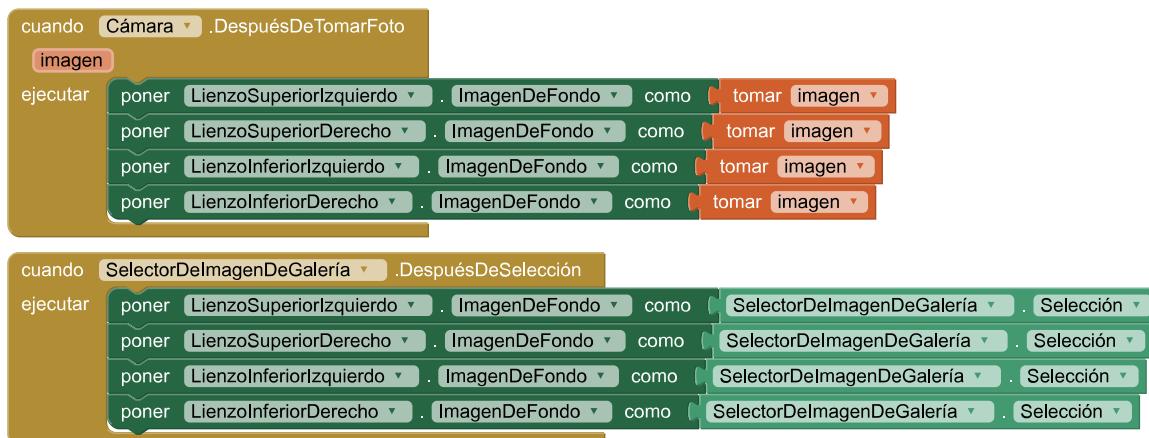
Una vez seleccionada una imagen, se produce un evento (se cierra la galería y vuelve a mostrarse la pantalla principal de la aplicación) que puede manejarse con el bloque **cuando**

**SelectorDeImagen.DespuésDeSelección / ejecutar { }**. Además, en la propiedad **SelectorDeImagen.selección** queda disponible la imagen escogida.



Bloques para manejar la selección de una imagen de la galería

Para ubicar la fotografía seleccionada de la galería en los lienzos hay que hacer esencialmente lo mismo que cuando se obtiene usando directamente la máquina de fotos: colocarla como imagen de fondo de los cuatro lienzos.

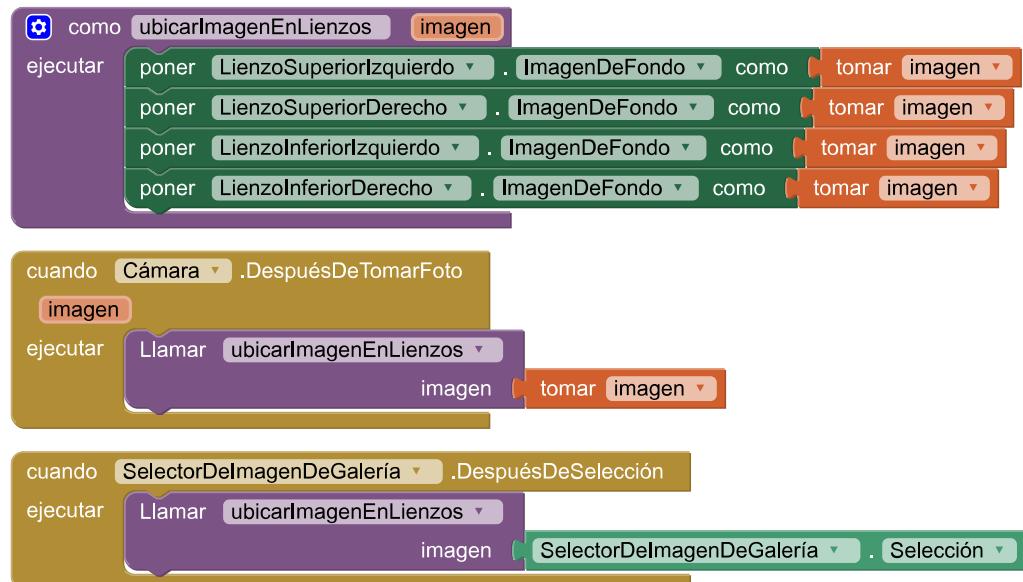


Manejo de los eventos que se producen luego de tomar una foto y luego de seleccionar una de la galería

Una vez que los estudiantes hayan alcanzado esta solución les preguntamos: “¿Cuál es la diferencia entre cómo manejan el evento que se produce al sacar una foto y el que se produce al seleccionar la imagen de la galería del teléfono?”. Guiamos el intercambio para concluir que en ambos casos se ubica la imagen como fondo de los cuatro lienzos; la única diferencia es que en `cuando Cámara .DespuésDeTomarFoto [imagen]` / `ejecutar { }` la foto se encuentra disponible en el parámetro `imagen`, mientras que en `cuando SelectorDeImagen .DespuésDeSelección / ejecutar { }` se la recupera desde la propiedad `SelectorDeImagen .Selección`.

Continuamos: “¿Se les ocurre cómo evitar hacer dos veces prácticamente lo mismo? Sería conveniente ocuparnos de colocar la imagen en los lienzos en un único punto del programa, ¿no?, independientemente de que la imagen la obtengamos desde la cámara o desde un archivo”. Alentamos entonces a los estudiantes a que piensen cómo conseguirlo.

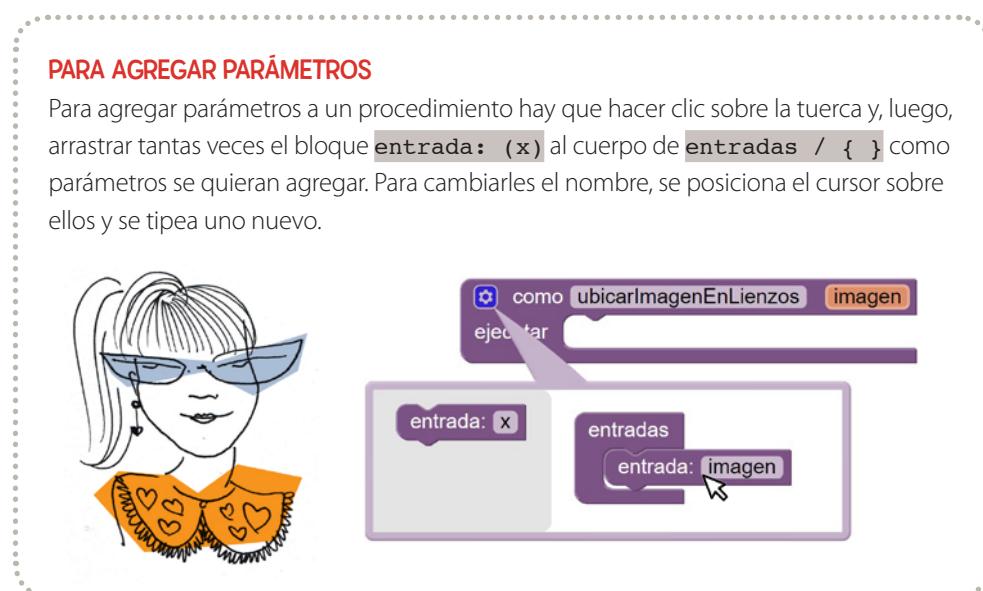
Para resolver el problema hay que crear un procedimiento con un parámetro que represente la imagen que se quiera colocar en los paños. Un nombre adecuado, en este caso, podría ser `ubicarImagenEnLienzos (imagen)`. Luego, desde `cuando Cámara .DespuésDeTomarFoto (imagen) / ejecutar { }` se lo invocará usando `imagen` como argumento; y desde `cuando SelectorDeImagen .DespuésDeSelección / ejecutar { }` usando `SelectorDeImagen .Selección`.



Solución de la segunda consigna

### PARA AGREGAR PARÁMETROS

Para agregar parámetros a un procedimiento hay que hacer clic sobre la tuerca y, luego, arrastrar tantas veces el bloque **entrada: (x)** al cuerpo de **entradas / { }** como parámetros se quieran agregar. Para cambiarles el nombre, se posiciona el cursor sobre ellos y se tipea uno nuevo.

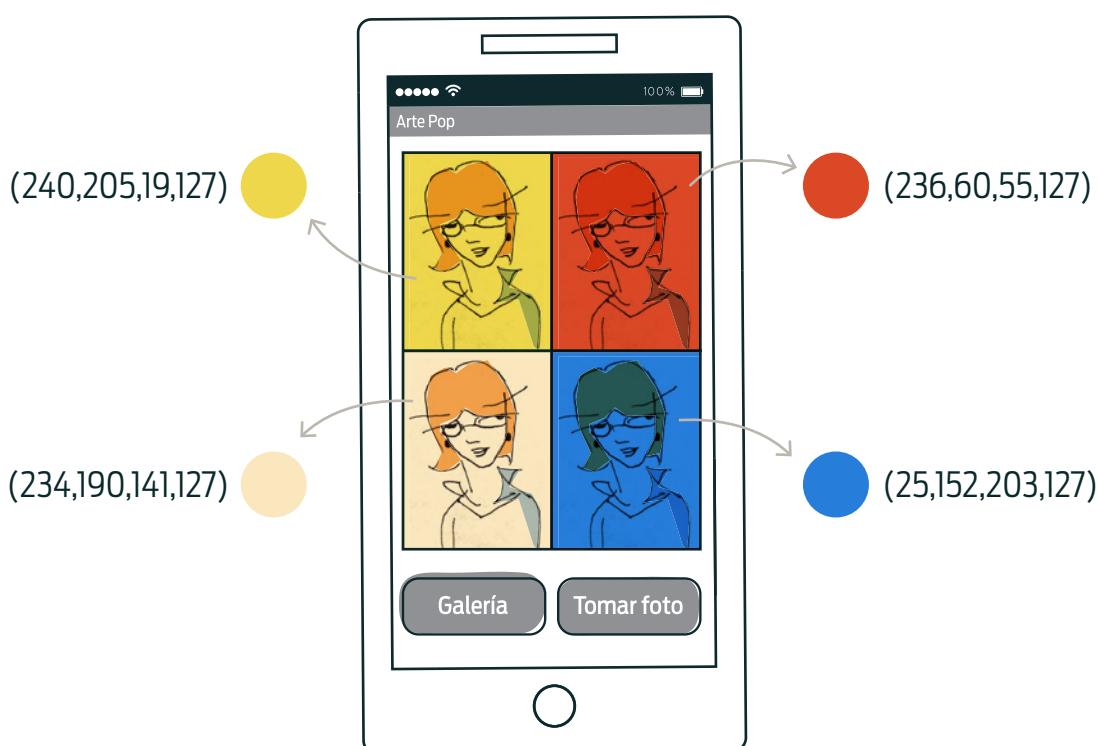


Antes de pasar a la siguiente consigna es importante señalar a los estudiantes que el alcance de los parámetros se limita al procedimiento del que forman parte: “Tanto en **Cámara**.

**DespuésDeTomarFoto (imagen)** como en **ubicarImagenEnLienzos (imagen)** hay un parámetro que se llama **imagen**. Sin embargo, tengan presente que se trata de dos parámetros distintos. En **Cámara.DespuésDeTomarFoto** representa la foto que se obtuvo con la cámara; en **ubicarImagenEnLienzos**, la imagen que se pretende ubicar en los lienzos. Recuerden que se trata solo de un nombre; los valores concretos surgen de los argumentos que se usan al invocar al procedimiento”.

#### Consigna 4: Aplicar filtros de colores

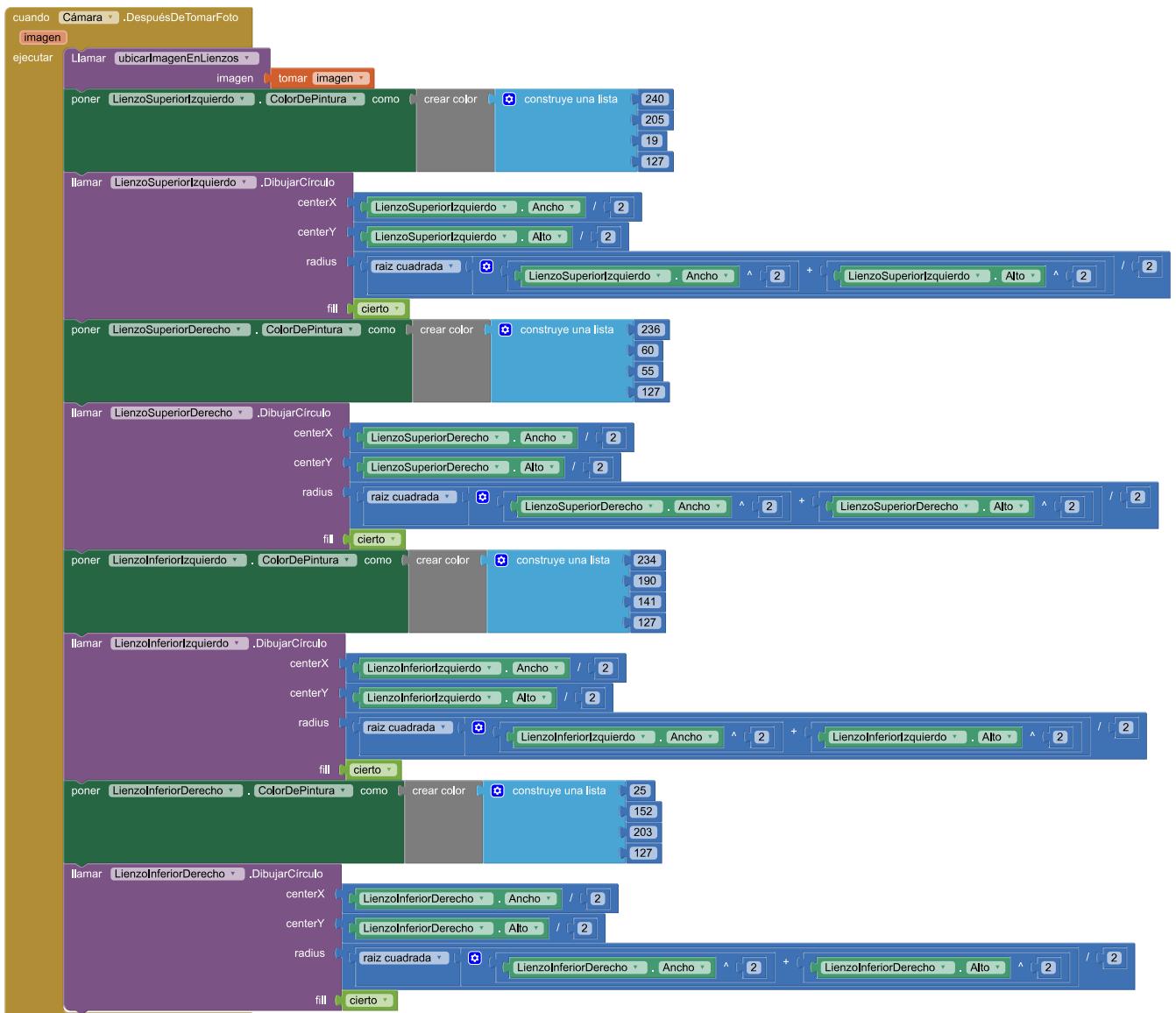
La cuarta consigna consiste en aplicar filtros de cuatro colores distintos, uno sobre cada foto, en forma similar a como lo hicieron en la actividad “Filtros de colores”. En la imagen se proponen cuatro colores, niveles de transparencia y disposiciones que, en conjunto, se asemejan a los usados en algunas obras de arte pop. Sin embargo, a lo largo del desarrollo de la actividad aconsejamos alentar a los estudiantes a que exploren distintas alternativas en busca de combinaciones cromáticas que les gusten, de modo de promover su creatividad.

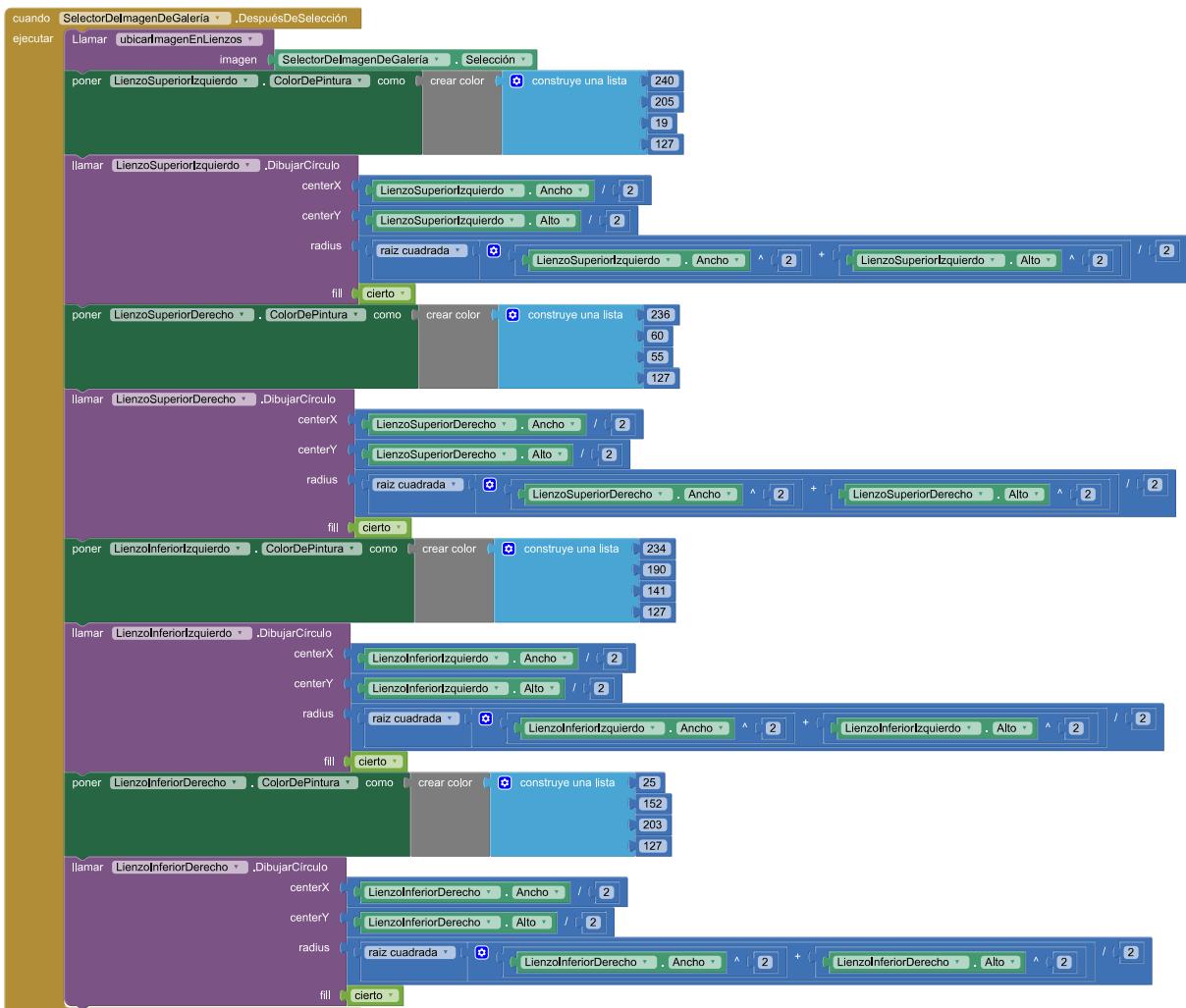


Propuesta de colores RGB + alpha

Es probable que los estudiantes esbozen distintas propuestas para completar la consigna. A continuación, analizamos una de ellas, observando ciertas debilidades que pueden aparecer y modos de corregirlas. Sin embargo, cualesquiera sean los programas que elaboren, es importante que usen procedimientos para descomponer problemas en subproblemas más simples y para evitar las redundancias, introduciendo parámetros en los casos que sean necesarios.

Una posible propuesta es que luego de posicionar la imagen sobre los lienzos se apliquen los filtros de colores, incluyendo los bloques correspondientes tanto en `Cámara.DespuésDeTomarFoto (imagen)` como en `cuando SelectorDeImagen. DespuesDeSelección / ejecutar { }`.



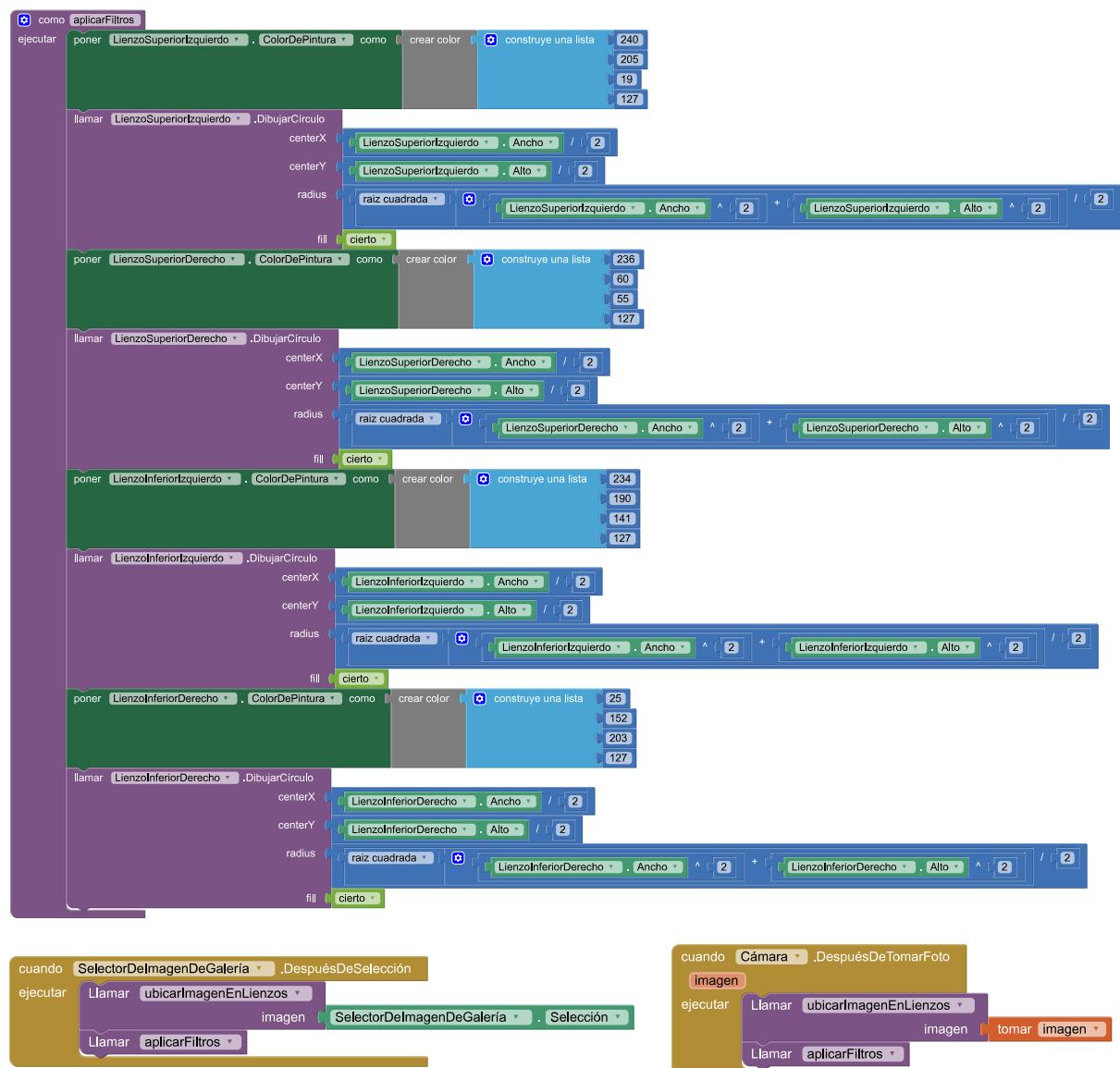


Possible resolution of the task 4

This program produces the expected effect –each canvas is painted with a color–, but it contains redundant application of color filters. Repeating the same group of instructions in different places of the program is a bad programming practice.

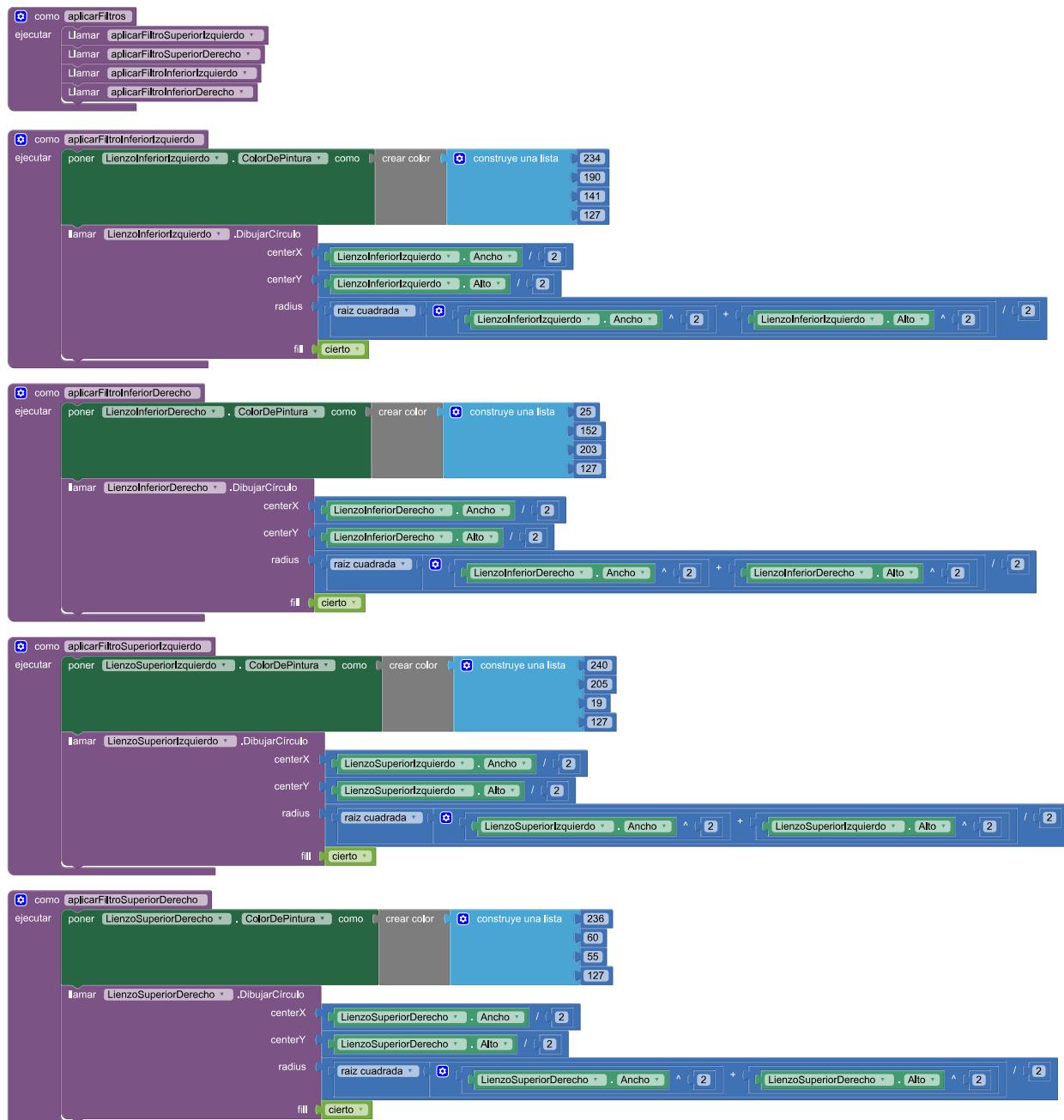
One of the most relevant uses of procedures is to divide a problem into subproblems. Then, the solutions of the subproblems can be combined to solve the original problem. In this sense, it is important to identify the different parts that make up the solution and solve each one separately.

Un rápido análisis permite darse cuenta de que en el programa siempre, luego de establecer la foto como fondo de los lienzos, se aplican filtros dibujando círculos. Por lo tanto, es lógico ubicar la aplicación de los filtros en un procedimiento aparte. Este procedimiento podría llamarse, por ejemplo, **aplicarFiltros**. Se obtiene de este modo un nuevo programa que funciona exactamente igual, pero con una estructura interna diferente, ahora sin porciones duplicadas.



Aplicación de filtros en un procedimiento separado

Como puede observarse, el cuerpo de `aplicarFiltros` es muy extenso,<sup>1</sup> debido a que se aplican los filtros sobre los cuatro lienzos. En general, la presencia de procedimientos largos es una señal de que la tarea encapsulada en ellos debe, a su vez, dividirse en subtareas más pequeñas. En este caso, una forma de descomponerlo consiste en crear otros cuatro procedimientos, uno para filtrar cada lienzo.



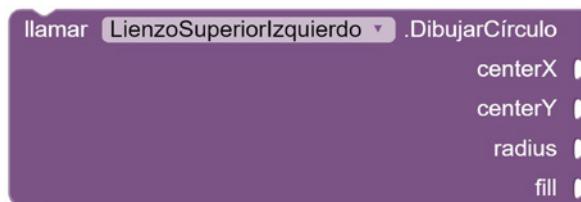
Aplicación de filtros en procedimientos separados

<sup>1</sup> En programación, a los módulos muy extensos de un programa se los llama *monolíticos*. Este tipo de módulos suelen aparecer cuando no se descomponen los problemas en subproblemas más chicos.

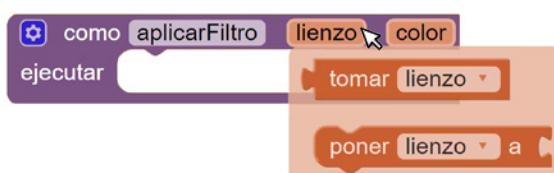
Llegado a este punto, compartimos con los estudiantes: "Si bien parecidos, los procedimientos que aplican los filtros no son idénticos. Difieren tanto en el color de la pintura como en el lienzo sobre el cual se aplica el filtro. ¿Se les ocurre cómo podríamos alcanzar una solución más sintética?". Es probable que surja la propuesta de crear un procedimiento con dos parámetros, uno que represente un lienzo y el otro un color. De este modo, se tiene una solución general que, después, puede invocarse con argumentos concretos para filtrar la imagen de cada lienzo con un color distinto. Un nombre adecuado para este nuevo procedimiento sería, por ejemplo, `aplicarFiltro (lienzo) (color)`.

Mientras contamos con un procedimiento por lienzo, pudimos referenciar en cada uno la propiedad `ColorDePintura` y el comando `DibujarCírculo` de un lienzo en particular. Por ejemplo, en `aplicarLienzoSuperiorIzquierdo` referenciamos a `LienzoSuperiorIzquierdo`. `ColorDePintura` e invocamos a `LienzoSuperiorIzquierdo.DibujarCírculo`. ¿Cómo se resuelve esto ahora, teniendo en cuenta que en `aplicarFiltro (lienzo) (color)` el lienzo es un parámetro *y, a priori*, no sabemos sobre qué lienzo hay que cambiar el color de la pintura y dibujar un círculo?

`poner LienzoSuperiorIzquierdo . ColorDePintura como`

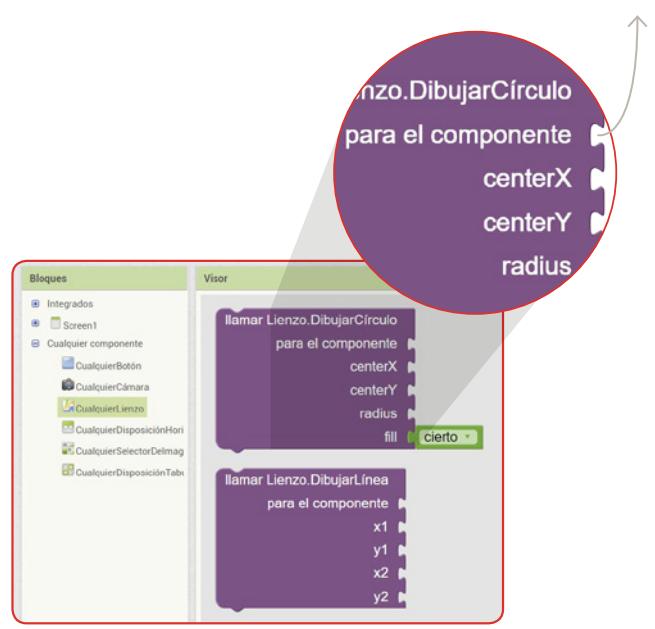


Al usar un componente concreto se accede a sus propiedades y comandos



El lienzo es un parámetro

Acá se encarta el lienzo sobre el que se quiere dibujar el círculo

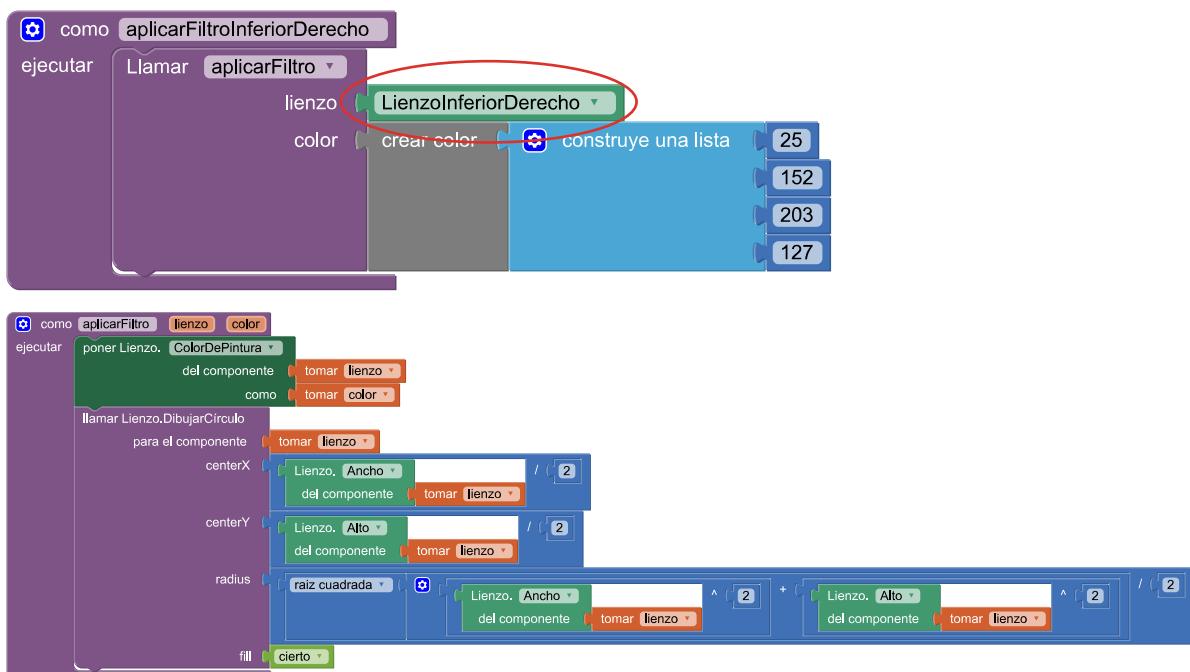


Bloques para cualquier lienzo

En *Bloques > Cualquier componente > CualquierLienzo* hay un conjunto de bloques que permiten acceder a las propiedades, los comandos y el manejo de los eventos de lienzos "genéricos", es decir, que puede usarse con cualquier lienzo.

Por ejemplo, en `Lienzo.DibujarCírculo` para el componente [ ]..., hay que indicar sobre qué lienzo se dibujará el círculo.

Usando los bloques para acceder a las propiedades `Lienzo.ColorDePintura`, `Lienzo.Ancho` y `Lienzo.Alto` y al comando `Lienzo.DibujarCírculo...` de un lienzo “genérico” se puede completar `aplicarFiltro (lienzo) (color)`. Este, a su vez, tiene que invocarse desde los procedimientos que filtran cada uno de los cuatro paños usando, en cada caso, los argumentos adecuados.



Procedimiento genérico (para cualquier lienzo y color) e invocación con argumentos para el lienzo inferior derecho

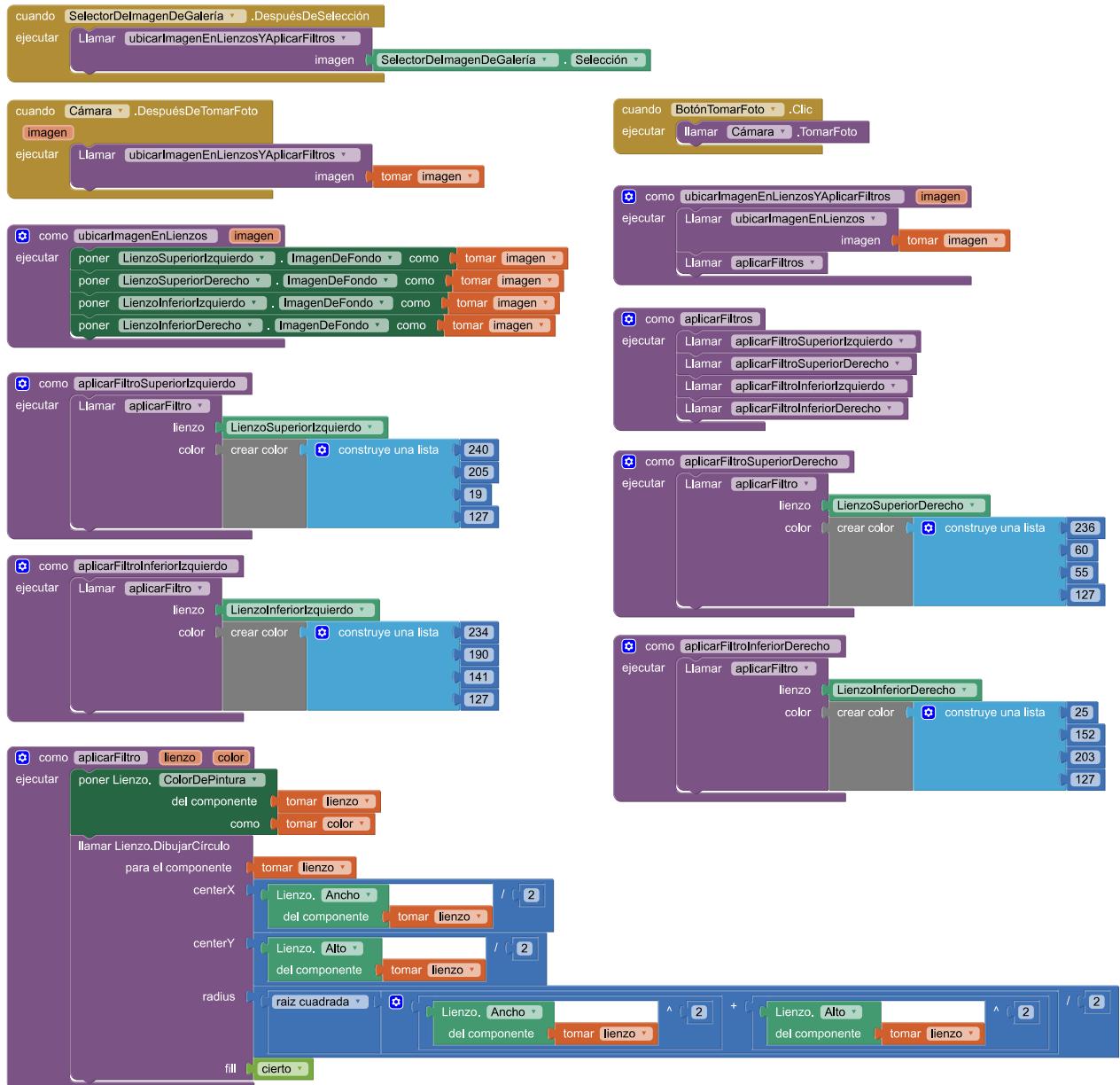
Para obtener el bloque que representa un lienzo hay que ir al menú del componente en cuestión.

Por ejemplo, `LienzoSuperiorIzquierdo` está disponible en *Bloques > Screen1 > Disposición Tabular > LienzoSuperiorIzquierdo*.

`LienzoSuperiorIzquierdo`

Bloque que representa un componente

Todavía se puede hacer un último cambio para conseguir un programa aún más modular. Los manejos de los eventos que se producen luego de tomar una foto o de seleccionar una imagen de la galería son muy parecidos: en ambos, primero se llama al procedimiento `ubicarImagenEnLienzos (imagen)` y, luego, a `aplicarFiltros`. La única diferencia está en el argumento usado al invocar al primero. Se puede, por lo tanto, crear un nuevo procedimiento con un parámetro que represente la imagen y, en cada caso, llamarlo con el argumento indicado. Un nombre adecuado sería `ubicarImagenEnLienzosYAplicarFiltros (imagen)`. A continuación, se muestra el programa completo.



Programa completo que resuelve el desafío

## CIERRE

Reflexionamos junto a los estudiantes sobre las posibilidades que brindan el uso de procedimientos y parámetros. Por un lado, crear procedimientos permite dividir un problema en varios problemas más pequeños. Por otro lado, los procedimientos pueden ser llamados tantas veces como haga falta, sin necesidad de escribir varias veces fragmentos del programa que comprenden las mismas instrucciones. Además, al usar parámetros, se consiguen soluciones más generales. En este caso, el comportamiento de los procedimientos se define una vez que se los llama y se les asigna valores concretos como argumentos.

NOMBRE Y APELLIDO:

CURSO:

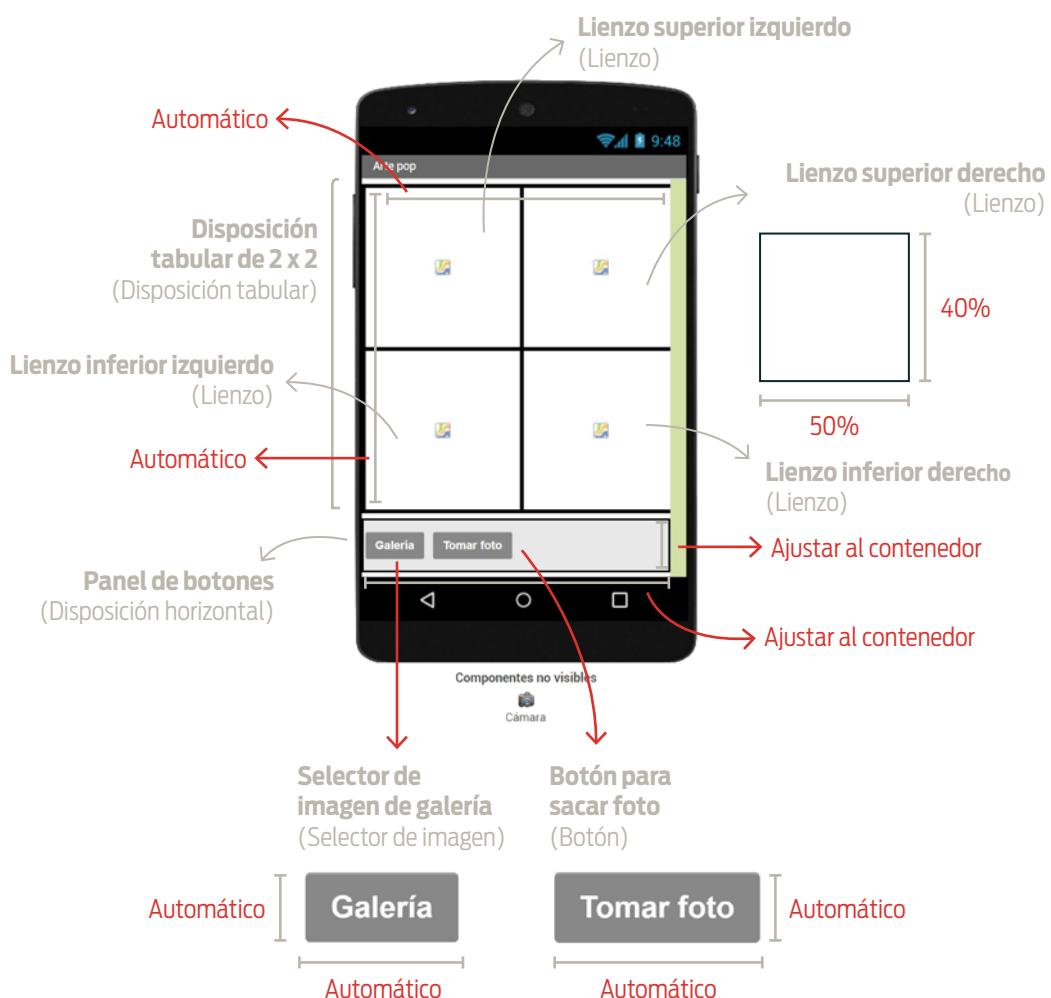
FECHA:

# TU ARTE POP

El arte pop fue un movimiento artístico que surgió en el Reino Unido y los Estados Unidos en la década de 1950. Desafiando las tradiciones de las bellas artes, sus obras incorporaban elementos pertenecientes a la cultura popular, como publicidades de latas de tomates e historietas. En esta actividad vas a crear una aplicación para transformar imágenes en ¡obras pop!



1. Arrancá por la interfaz gráfica. En la imagen vas a encontrar los componentes y sus dimensiones. En la tabla, las propiedades que tenés que modificar para que luzca como la imagen.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

### SELECTOR DE IMAGEN

App Inventor provee un componente que permite seleccionar una imagen de la galería. Se trata de *SelectorDelImagen*, que se encuentra disponible en Paleta > Medios.



NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen	Arte Pop
LienzoSuperior Izquierdo	Lienzo	Alto	Automático	40%
		Ancho	Automático	50%
LienzoSuperior Derecho	Lienzo	Alto	Automático	40%
		Ancho	Automático	50%
LienzoInferior Izquierdo	Lienzo	Alto	Automático	40%
		Ancho	Automático	50%
LienzoInferior Derecho	Lienzo	Alto	Automático	40%
		Ancho	Automático	50%
PanelDeBotones	Disposición horizontal	Disp. vertical	Arriba	Centro
		Alto	Automático	Ajustar al contenedor
		Ancho	Automático	Ajustar al contenedor
SelectorDelImagen DeGalería	Selector de imagen	Color de fondo	Automático	Gris
		Negrita	(ninguno)	✓
		Texto	Texto para botón	Galería
		Color de texto	Por defecto	Blanco
BotónTomarFoto	Botón	Color de fondo	Automático	Gris
		Negrita	(ninguno)	✓
		Texto	Texto para botón	Tomar foto
		Color de texto	Por defecto	Blanco

NOMBRE Y APELLIDO:

CURSO:

FECHA:

- 2.** Incorporá los bloques necesarios para sacar una foto y ubicarla como fondo de los cuatro lienzos. ¿Cómo lo hiciste? ¿Qué eventos manejaste para lograrlo?

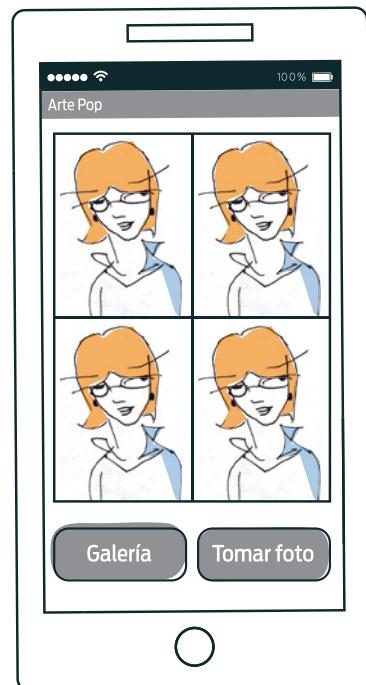
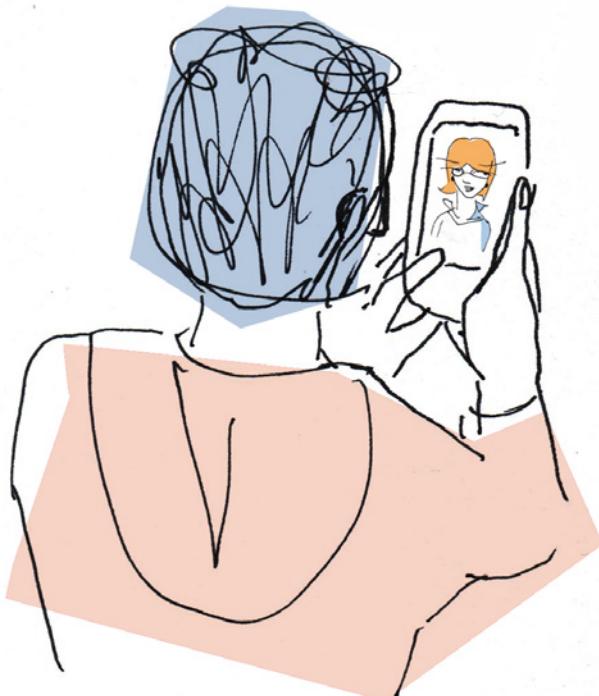
---

---

---

---

---



#### EL PADRE DEL ARTE POP

Uno de los exponentes más relevantes de este movimiento fue el artista Andy Warhol.

Hay un museo exclusivamente dedicado a este artista (<https://www.warhol.org/>), además de que sus obras forman parte de exposiciones de muchos de los museos más importantes del mundo.

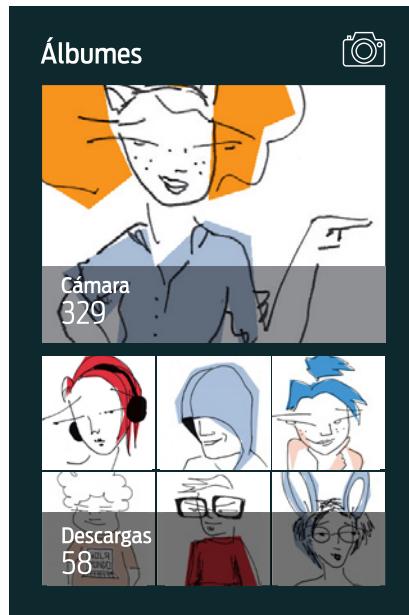


NOMBRE Y APELLIDO:

CURSO:

FECHA:

3. Ahora, ocúpate de que la imagen también pueda seleccionarse desde la galería de imágenes de tu teléfono.



¿Qué diferencia encontrás entre el bloque para manejar el evento que se produce al sacar una foto y el que se produce al seleccionar una imagen de la galería?

cuando SelectorDeImagenDeGalería .DespuésDeSelección  
ejecutar

cuando Cámara .DespuésDeTomarFoto  
imagen  
ejecutar

#### PARA QUE TENGAS EN CUENTA

La foto que selecciones en la galería  
está disponible en la propiedad  
SelectorDeImagen.Selección.

SelectorDeImagenDeGalería . Selección

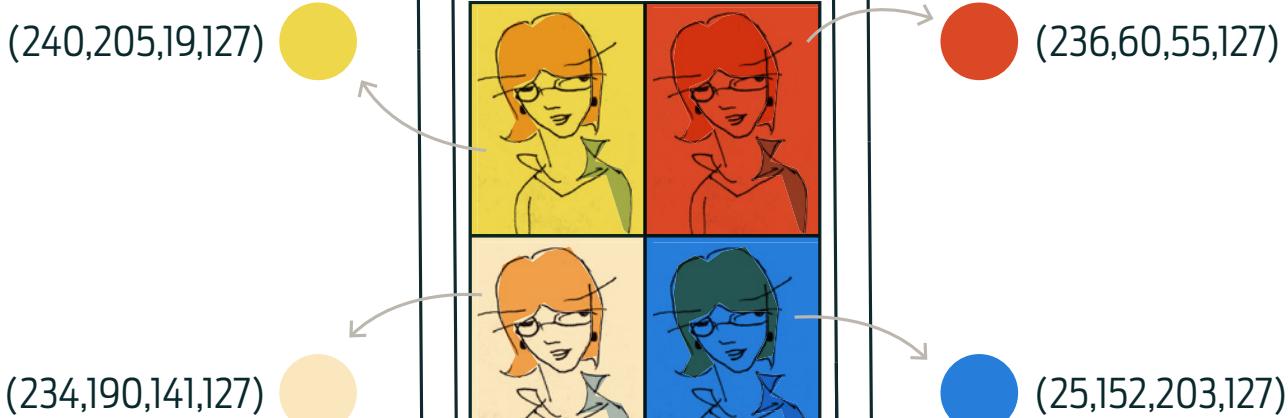


NOMBRE Y APELLIDO:

CURSO:

FECHA:

4. Llegó el momento de transformar la foto en una pieza pop. Aplicá filtros de cuatro colores sobre cada uno de los lienzos. Acá te pasamos una opción de colores que remite a algunas obras de este movimiento artístico, pero probá con otras hasta encontrar la combinación que más te guste.



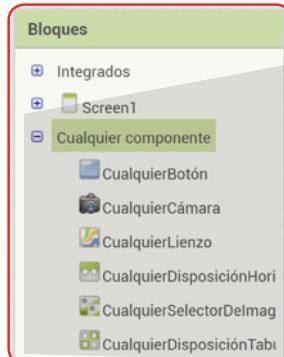
#### CONSEJOS

- Usá procedimientos para descomponer problemas en unidades más simples.
- Si en tu programa te encontrás con fragmentos muy parecidos entre sí, cuyo propósito es prácticamente el mismo, considerá definir un nuevo procedimiento con uno o varios parámetros.



#### CUALQUIER COMPONENTE

¡Investigá qué hacen los bloques que se encuentran en la categoría *CualquierComponente*!



# 04

# ALTERNATIVAS, REPETICIONES Y VARIABLES

Las condiciones en las cuales se ejecutan los programas suelen ser desconocidas a la hora de programar. Por lo tanto, debemos considerar distintos escenarios y las acciones adecuadas a realizar en cada caso. Los lenguajes de programación permiten expresar **alternativas condicionales**, que indican que algunas instrucciones solo tienen que ejecutarse en algunos casos, según se cumpla o no cierta condición.

## SECUENCIA DIDÁCTICA 1

ALTERNATIVAS CONDICIONALES  
Piedra, papel o tijera  
Cara o ceca

## SECUENCIA DIDÁCTICA 2

MENTE MAESTRA  
Mente maestra, desenchufada  
Mente maestra, parte I  
Mente maestra, parte II  
Mente maestra, parte III

Por otro lado, hay problemas cuya resolución requiere repetir, en forma consecutiva, una serie de acciones. En este capítulo mostraremos cómo construir programas usando **repeticiones**, que son comandos que permiten expresar la reiteración de acciones sin necesidad de escribirlas varias veces.

Por último, es frecuente que los programas requieran guardar información, recuperarla y modificarla. Para ello existen las **variables**, que abstraen un espacio de la memoria de la computadora para almacenar información.



## Secuencia Didáctica 1

# ALTERNATIVAS CONDICIONALES

Cuando una aplicación se ejecuta, acontecen circunstancias que no conocemos al momento de construir el programa. Por ejemplo, si pensamos en una máquina expendedora de golosinas, ¿sabemos con billetes de qué monto se hará una compra y, por lo tanto, cuánto vuelto debe darse? Los lenguajes de programación tienen un mecanismo para expresar **alternativas condicionales**, para que los programas realicen unas u otras acciones dependiendo de si ciertas condiciones se producen o no.

Esta secuencia didáctica tiene dos actividades. En la primera se introduce la noción de alternativa condicional sin usar computadoras. En la segunda, se propone un pequeño proyecto de programación cuya resolución requiere el uso de esta herramienta.

### ..... **OBJETIVOS**

- Introducir las alternativas condicionales.
  - Utilizar alternativas condicionales en un proyecto de programación.
- .....

## Actividad 1

### Piedra, papel o tijera



TODA LA CLASE

#### OBJETIVO

- Introducir las alternativas condicionales.

#### MATERIALES

Pizarrón

Tizas o fibrones

#### DESARROLLO

El objetivo de esta actividad es introducir las **alternativas condicionales**. Se propondrá a los estudiantes que jueguen al clásico juego *Piedra, papel o tijera* y que, luego, describan sus reglas. Debido a la naturaleza del juego, las reglas serán expresadas usando sentencias condicionales, lo que dará pie a analizar su morfología y conceptualizar sobre su estructura.

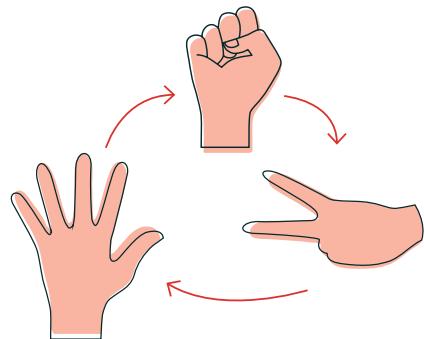
Comenzamos indicando a los estudiantes que, en parejas, jueguen al *Piedra, papel o tijera*. En caso de que algunos no conociesen las reglas, las recordamos.

#### Reglas del juego

*Piedra, papel o tijera* es un juego entre dos personas en el que, simultáneamente, cada jugador opta por formar con una mano una figura que denota uno de tres elementos: una piedra, una hoja de papel o una tijera. La piedra se representa con el puño cerrado; el papel con la mano extendida; y la tijera con los dedos índice y mayor extendidos, formando una V similar a la que se usa para simbolizar “victoria” o “paz”, con la diferencia de que apunta horizontalmente en lugar de verticalmente.

El juego comienza con los jugadores enfrentados, colocando sus manos detrás de sus espaldas y diciendo al unísono la frase “Piedra, papel o tijera”. Inmediatamente después, cada uno elige un elemento y le muestra la mano al contrincante, simbolizando con ella el elemento elegido.

El juego puede tener un ganador o resultar empatado. Si un jugador elige piedra y el otro papel, gana el que elige papel, pues el papel envuelve a la piedra; si uno elige piedra y el otro tijera, gana el que elige piedra, ya que la piedra rompe la tijera; si uno elige papel y el otro tijera, gana el que elige tijera, porque la tijera corta al papel; por último, si ambos eligen el mismo elemento, se produce un empate.



Representación de los elementos con las manos

Una vez que hayan jugado varias partidas, los invitamos a que grupalmente describan las reglas del juego. Es probable que algún estudiante exprese las reglas en forma similar a la descripción dada en el párrafo precedente. Entonces, anotamos en el pizarrón las condiciones que determinan un ganador o un empate.

Si un jugador elige piedra y el otro papel,  
entonces gana el que elige papel.

Si un jugador elige piedra y el otro tijera,  
entonces gana el que elige piedra.

Si un jugador elige papel y el otro tijera,  
entonces gana el que elige tijera.

Si ambos jugadores eligen el mismo  
elemento, entonces empatan.

#### Reglas del *Piedra, papel o tijera*

Les preguntamos: “Una vez que cada jugador hace su elección, ¿cómo hacemos para determinar si hay un empate o, alternativamente, quién gana?”. Guiamos el intercambio para concluir que observamos los elementos elegidos por los jugadores y, siguiendo las reglas, sabremos el resultado. Comentamos: “Muy bien, aquí es importante que nos detengamos en algo. Para describir las reglas consideramos todas las posibles combinaciones; sin embargo, al jugar una partida, solo una resulta ser cierta. Claro que de antemano no podríamos haber sabido cuál sería, ¿no?”.

Continuamos: “¿Notan alguna similitud entre las oraciones que describen las reglas del juego?”. Escuchamos con atención sus observaciones y concluimos que todas ellas tienen la misma estructura. En el pizarrón, subrayamos en cada una las palabras *Si* y *entonces* y agregamos la frase *Si condición entonces algo*.

Si condición, entonces algo.

Si un jugador elige piedra y el otro papel,  
entonces gana el que elige papel.

Si un jugador elige piedra y el otro tijera,  
entonces gana el que elige piedra.

Si un jugador elige papel y el otro tijera,  
entonces gana el que elige tijera.

Si ambos jugadores eligen el mismo  
elemento, entonces empatan.

#### Alternativas condicionales

Luego, comentamos: “Aquí hay una estructura que podemos identificar. Primero aparece la palabra *Si*, luego enunciamos una condición (también llamada *proposición*), que puede ser cierta o falsa; a continuación, la palabra *entonces* y, finalmente, lo que sucede en caso de que la condición sea verdadera. Se las presento: estas sentencias permiten expresar alternativas condicionales”.

### **CIERRE**

Les comentamos a los estudiantes que los lenguajes de programación proveen instrucciones que permiten incluir alternativas condicionales en los programas. De este modo, un programa se comporta de un modo u otro dependiendo de que ciertas condiciones sean verdaderas o falsas. Remarcamos que, al igual que lo que sucedió al describir las reglas, al programar hay que incluir todas las condiciones que resulten relevantes para la aplicación que estamos construyendo, ya que de antemano, antes de que el programa se encuentre en ejecución, no sabemos cuáles serán verdaderas y cuáles no.

---

## Actividad 2

### Cara o ceca



#### OBJETIVOS

- Utilizar alternativas condicionales en un programa.
- Presentar la noción de aleatoriedad.

#### MATERIALES

- Moneda
- Computadora
- Internet
- MIT App Inventor 2
- Ficha para estudiantes
- Teléfono con Android

#### DESARROLLO

El objetivo de esta actividad es utilizar alternativas condicionales en un pequeño proyecto de programación. Se propone desarrollar una aplicación que simule una tirada de una moneda. Además, para completar el proyecto, tendrán que usar números aleatorios, que son muy frecuentemente utilizados en distintos contextos de computación.<sup>1</sup>

Comenzamos preguntando: “¿Vieron que antes de comenzar un partido de fútbol el árbitro llama a los capitanes de ambos equipos? ¿Saben para qué?”. Escuchamos las respuestas y continuamos: “A uno de ellos le da la posibilidad de que escoja el lado en el que comienza jugando su equipo. ¿Tienen idea cómo se dirime cuál de los dos elige?”. Es probable que alguien conteste que se resuelve tirando una moneda. “Efectivamente, se tira una moneda y, de acuerdo al resultado, uno u otro escogerá el lado que prefiera. De algún modo, la posibilidad de elegir se libra al azar: al tirar una moneda saldrá o bien cara o bien ceca, pero no lo sabemos de antemano”.

Invitamos a dos estudiantes al frente y les decimos que, tirando una moneda, vamos a hacer un sorteo. Les comentamos: “El premio es secreto, ya verán de qué se trata...”. Les indicamos que, entre ellos, decidan quién gana si sale cara y quién si sale ceca. Entonces, tiramos la moneda y le decimos al ganador: “¡Has ganado! El premio consiste en... ¡más sinceras felicitaciones!”. Les pedimos a continuación que vuelvan a sus pupitres.

Continuamos: “Las monedas valen cada vez menos y, en general, no llevamos ninguna en el bolsillo. Ahora harán una aplicación que simule la tirada de una moneda. Así, si la vida los pone frente a una situación con dos posibles resultados que hay que definir mediante el azar, podrán resolverla con sus teléfonos, tengan o no una moneda encima”.

#### Consigna 1: incorporación de archivos

Esta actividad requiere no solo disponer componentes sobre la pantalla del teléfono sino que, además, incorpora imágenes y sonidos. Al ejecutar la aplicación, se mostrará como pantalla de inicio la imagen del archivo inicio.png; para representar los lados cara y ceca de la moneda usaremos otros dos archivos de imágenes: cara.png y ceca.png; y, para representar el sonido que produce una moneda al ser arrojada por el aire, el archivo lanzamiento-moneda.mp3.<sup>2</sup><sup>3</sup>

<sup>1</sup> Puede accederse a una versión completa de la aplicación buscando “programar2020” en la galería de aplicaciones de App Inventor.

<sup>2</sup> Estos archivos pueden reemplazarse por otros; los aquí propuestos son solo una opción.

<sup>3</sup> App Inventor admite otros formatos de archivos de imágenes y sonidos, como por ejemplo jpg o wav. Para más información, visitar la página <http://bit.ly/2XqMQ2u>.



Archivos cara.png y ceca.png

### CÓMO OBTENER LOS ARCHIVOS DE LA ACTIVIDAD

Los archivos inicio.png, cara.png, ceca.png y lanzamiento-moneda.mp3 pueden descargarse del proyecto de referencia (disponible en la galería de proyectos de "programar2020"). En el editor de diseño, dentro del panel *Medios* –ubicado bajo el panel *Componentes*–, se encuentran listados los archivos incorporados al proyecto. Al hacer clic sobre cada uno, se despliega un menú contextual en el que hay que seleccionar la opción *Descargar a mi ordenador*.



Repartimos la ficha y distribuimos los archivos de imágenes y el de sonido a los estudiantes. Podemos hacerlo con un *pendrive*, enviárselos por correo electrónico, etc. Les indicamos que creen un proyecto *Cara\_o\_ceca* y, a continuación, los incorporen, tal como pide la primera consigna.

Para agregar archivos a un proyecto, en el panel *Medios* tienen que presionar el botón *Subir archivo*. Entonces se abrirá una ventana *pop up* en la que, en primer lugar, tienen que hacer clic en *Seleccionar archivo* para seleccionar el archivo que se desea incorporar; y, en segundo, presionar *Aceptar* para incorporarlo al proyecto.<sup>1</sup> Finalmente, en *Medios* verán aparecer el archivo incorporado.

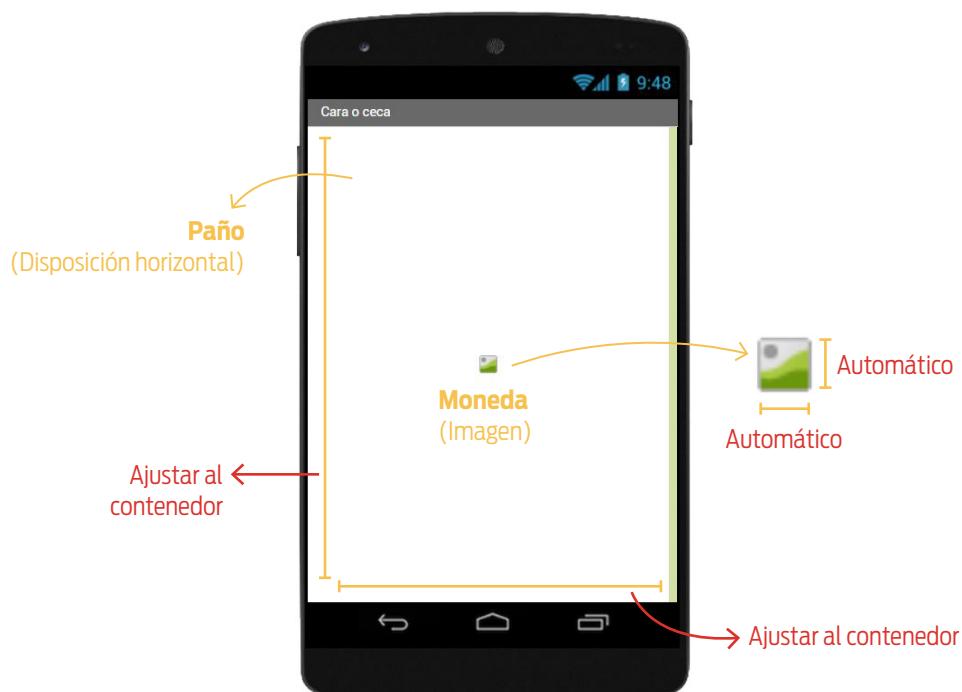


Se incorpora un archivo al proyecto

<sup>1</sup> Este proceso puede demorar unos segundos, porque el archivo se transfiere desde la computadora hacia el servidor de MIT donde reside el proyecto.

## Consigna 2: interfaz gráfica

Una vez incorporados los archivos, les pedimos que sigan las indicaciones de la segunda consigna para armar la interfaz de la aplicación. La tabla muestra los valores de las propiedades que hay que cambiar (en relación a los que App Inventor asigna por defecto) para que la aplicación se vea tal como en la imagen abajo.



Diseño de la interfaz gráfica

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen1	Cara o ceca
Paño	Disposición horizontal	Disposición horizontal	Izquierda	Centro
		Disposición vertical	Arriba	Centro
		Alto	Automático	Ajustar al contenedor
		Ancho	Automático	Ajustar al contenedor
Moneda	Imagen	Foto	Ninguno	inicio.png
		Escalar foto al tamaño máximo	(ninguno)	✓

Valores diferentes a los asignados por defecto

Cuando los estudiantes hayan terminado de componer la interfaz, les comentamos: "Como pueden observar, los componentes de tipo *Imagen* tienen una propiedad que se llama *Foto*. Ustedes, recién, la asociaron con el archivo cara.png, lo que significa que esa es la imagen que mostrará cuando pongamos a correr la aplicación. Sin embargo, es importante que sepan que ustedes, en sus programas, pueden hacer que cambie por otra mientras la aplicación se encuentra en ejecución".

### Consigna 3: se lanza la moneda

La tercera consigna plantea el desafío principal de la actividad: que al agitar el teléfono se simule la tirada de una moneda. Aquí, el programa tiene que llevar a cabo alguna operación en la que haya dos resultado posibles equiprobables.

Para empezar, les indicamos a los estudiantes que ejecuten la aplicación en sus teléfonos. Se encontrarán entonces que, al iniciar, en la pantalla se muestra la imagen de inicio. Lo primero que deben notar es que, con los componentes incorporados hasta aquí, no se puede detectar cuándo el teléfono es agitado. Para hacerlo hay que agregar un Acelerómetro que se encuentra en *Paleta > Sensores* del editor de diseño. Se trata de un sensor cuya función es, justamente, detectar zarandeados del dispositivo. Por tratarse de un componente no visible, al agregarlo aparecerá bajo el visor. Una vez incorporado podrá manejarse el evento que se produce al agitar el teléfono.



Pantalla de inicio de Cara o ceca

Una pregunta que naturalmente se harán los estudiantes es cómo simular un proceso en el que intervenga el azar. Aquí, hará falta una operación cuyo resultado pueda adquirir distintos valores cada vez que se la lleva a cabo.

Al explorar el entorno, descubrirán que en *Bloques > Integrados > Matemáticas* se encuentra `entero aleatorio entre [ ] y [ ]`. Se trata de un bloque que, dados dos números, arroja como resultado uno que se encuentre entre ellos (incluyéndolos). Por defecto utiliza los valores 1 y 100, pero estos pueden cambiarse por otros, como por ejemplo 1 y 2.

El lanzamiento de una moneda tiene dos resultados posibles mutuamente excluyentes: o bien sale cara o bien sale ceca. Entonces, si generamos un número al azar entre 1 y 2 podemos asociar a uno de ellos con el lado cara y, al otro, con el lado ceca. Por ejemplo, al 1 con cara y al 2 con ceca. En nuestro programa, el resultado determinará cuál foto mostrará el componente *Moneda*. Siguiendo con el ejemplo, si sale 1 mostrará *cara.png* y, si sale 2, *ceca.png*.

Para evaluar una condición y, de acuerdo a su valor de verdad –si es cierta o si es falsa–, hacer una u otra cosa, hay que agregar al programa una alternativa condicional. En *Bloques > Integrados > Control* se encuentra `si [ ] / entonces { }`. En el espacio que se encuentra en `si [ ] ...` hay que encastrear bloques que representen una expresión booleana –es decir, que el resultado de evaluarla sea verdadero o falso–; en `... entonces { }`, aquello que queramos que realice nuestro programa en caso de que la expresión sea cierta.



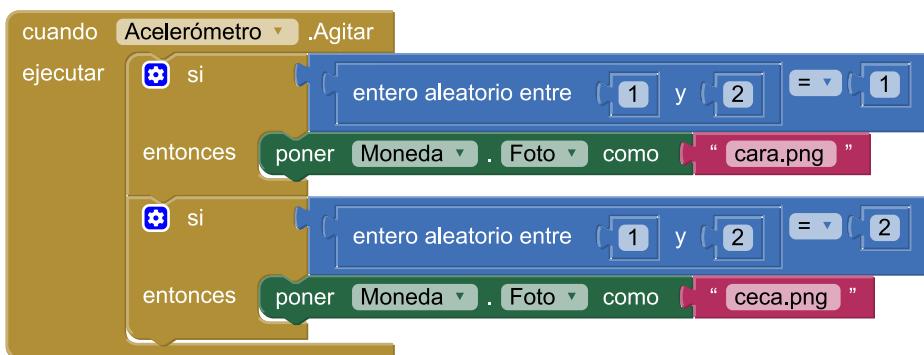
Bloque `si [ ] / entonces { }`

Combinando `entero aleatorio entre [ ] y [ ]` y la comparación entre números `[ ] = [ ]` de *Bloques > Integrados > Matemáticas*, se puede formar la expresión que indicará si el número aleatoriamente generado es 1. Además, para mostrar la imagen que corresponde, hay que establecer la propiedad *Imagen.Foto*: se debe indicar con un texto el nombre del archivo que tiene la imagen que hay que mostrar.



Muestra cara si el número es 1

Para completar la consigna aún resta considerar el caso en el que el número aleatorio sea 2. Algun estudiante podría sugerir imitar lo hecho previamente, modificando 1 por 2 y "cara.png" por "ceca.png", arribando al programa que muestra la figura.



Programa incorrecto que genera dos números al azar

Este programa no es correcto y produce resultados no deseados. En lugar de generar un número al azar para determinar si el resultado es cara o ceca, genera dos: uno en la condición del primer `si [ ... ]` y otro en la del segundo `si [ ... ]`. Una de las consecuencias es que las alternativas no resultan equiprobables; otra, que luego de agitar el teléfono podría mostrarse la pantalla de inicio en lugar de alguna de las dos caras de la moneda. Para verlo, alcanza con analizar lo que sucede la primera vez que se agita el teléfono.

Hay cuatro posibles combinaciones que pueden surgir al generar dos números aleatorios entre 1 y 2: [1,1], [1,2], [2,1] y [2,2].

**Caso [1,1].** La primera condición resulta verdadera, con lo que se establece que la foto del componente *Moneda* sea *cara.jpg*; luego, la segunda condición resulta falsa.

**Caso [1,2].** La primera condición resulta verdadera, con lo que en primer lugar se establece que la foto de *Moneda* sea *cara.jpg*. Luego, la segunda condición también resulta verdadera, por lo que se cambia la foto por *ceca.png*.

**Caso [2,1].** La primera condición resulta falsa y la segunda también. Por lo tanto, no se establece una nueva foto para *Moneda*, de modo que queda *inicio.png* –la que se muestra al cargar la aplicación–.

**Caso [2,2].** La primera condición resulta falsa y luego la segunda verdadera, estableciendo como foto de *Moneda* *ceca.png*.

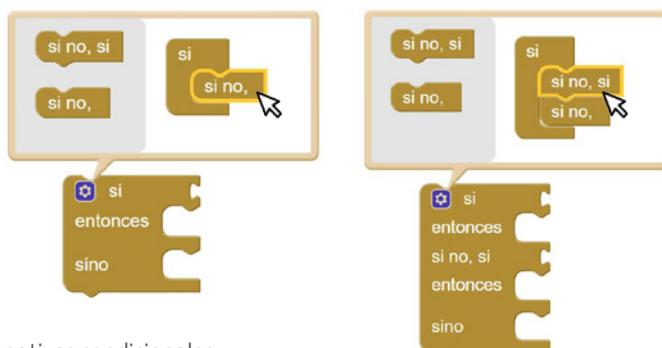
El siguiente cuadro resume los posibles resultados que pueden darse al agitar por primera vez el teléfono.

NÚMEROS OBTENIDOS AL AZAR	IMAGEN MOSTRADA EN EL TELÉFONO
[1,1]	cara.png
[1,2]	ceca.png
[2,1]	inicio.png
[2,2]	ceca.png

Programa incorrecto que genera dos números al azar

Como puede observarse, la probabilidad de mostrar cara.png es  $\frac{1}{4}$ , mostrar ceca.png  $\frac{2}{4} = \frac{1}{2}$  y que se siga mostrando la pantalla de inicio  $\frac{1}{4}$ .

El problema se origina al generar dos números aleatorios en lugar de uno. Si se generase uno solo, se podría establecer que si sale 1 el resultado sea cara y, si no, que sea ceca (si no es 1, la única posibilidad es que sea 2). Al presionar la tuerca que aparece en la esquina superior izquierda del bloque `si [ ] / entonces { }`, se despliegan opciones que podemos incorporar a la alternativa condicional. En este caso, como solo hay dos posibles resultados, alcanza con incorporar `si no`, –para lo cual hay que arrastrarlo y encastarlo dentro del bloque `si`–. Si las opciones fuesen más de 2, también habría que incorporar `si no, si` tantas veces como fuera necesario.



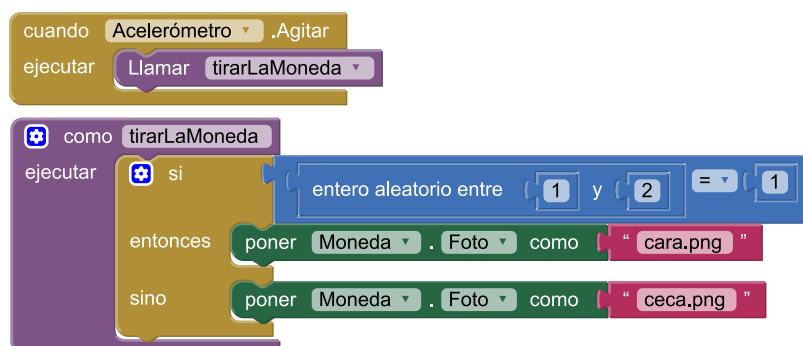
Opciones para alternativas condicionales

Utilizando `si [ ] / entonces { } / sino { }` se puede construir un programa que se comporte de acuerdo a lo esperado.



Programa correcto

A los estudiantes que alcancen esta solución, los instamos a que piensen otra alternativa, buscando que incorporen un procedimiento. Si escogen un buen nombre, conseguirán un programa más fácil de entender. Por ejemplo, en el programa de la imagen a continuación, resulta claro que al agitar el teléfono se tira la moneda y lo que está dentro del `tirarLaMoneda` comprende las acciones que se realizan en un lanzamiento. En el programa anterior, al no utilizar procedimientos con nombres descriptivos, la comprensión del programa requería un cierto grado de análisis.<sup>1</sup>

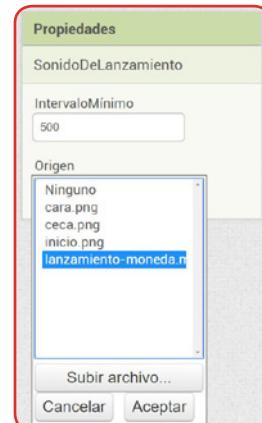


Programa con un procedimiento

#### Consigna 4: vibración y sonido

La cuarta consigna pide incorporar dos últimas características: al agitar el teléfono, hacerlo vibrar y reproducir el sonido de un lanzamiento de moneda. Para conseguirlo es necesario agregar un nuevo componente de tipo *Sonido* a la aplicación, disponible en *Paleta > Medios*. Una vez incorporado, hay que establecer la propiedad *Origen* como *lanzamiento-moneda.mp3*. De este modo, cada vez que el programa ejecute una instrucción para que suene el sonido, se reproducirá lo grabado en el archivo.

En el editor de bloques, al hacer clic sobre el nuevo componente, se desplegará una serie de bloques relacionados a los componentes de tipo *Sonido*. En particular, dos de ellos que permiten completar la actividad: `Sonido.Reproducir` y `Sonido.Vibrar milisegundos [ ]`. Este último, requiere un argumento numérico que indique el lapso de tiempo de vibración expresado en milisegundos.



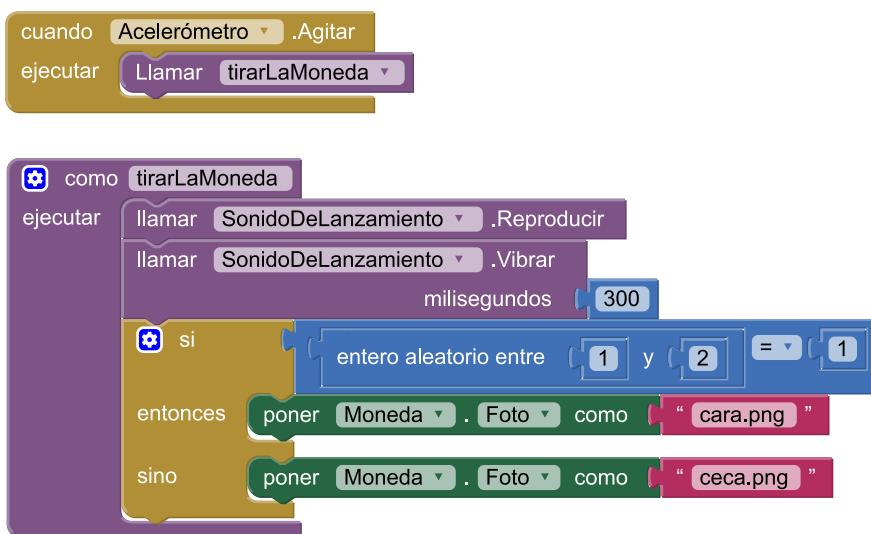
Propiedades de componentes de tipo *Sonido*



Bloques de componentes de tipo *Sonido*

<sup>1</sup> Esta observación resulta muy notable a medida que un programa crece en tamaño y complejidad.

A continuación, una solución completa de la actividad.



Programa que resuelve el desafío

### CIERRE

Reflexionamos con los estudiantes sobre cómo las alternativas condicionales permiten que los programas varíen su comportamiento de acuerdo a ciertas condiciones cuyo valor de verdad se determina recién cuando la aplicación se encuentra en ejecución y no mientras el programa se construye. Comentamos, además, que el uso de números al azar es muy común en el universo de la computación. Muchas veces se tiene que simular un proceso con un final incierto, para lo cual este tipo de números es muy útil.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# CARA O CECA

Muchas veces enfrentamos situaciones que se tienen que resolver mediante el azar, como decidir qué materia estudiar cuando falta solo un día para los exámenes de dos materias y no sabemos nada de ninguna; o si será tu amigo o vos quien irá a la pizzería de la otra cuadra a comprar unas empanadas y volver. Ahora vas a hacer una aplicación para poder resolverlas, aun cuando no tengas una moneda en el bolsillo.



1. Creá un proyecto de App Inventor e incorporá cuatro archivos multimedia: inicio.png, cara.png, ceca.png y lanzamiento-moneda.mp3.



## PARA AGREGAR ARCHIVOS A UN PROYECTO

Para agregar archivos a un proyecto, tenés que presionar el botón Subir archivo del panel Medios. Entonces,

se desplegará una ventana en la que, en primer lugar, tenés que hacer clic en Seleccionar archivo (para seleccionar el archivo que querés incorporar); y, en segundo, presionar Aceptar para incorporarlo al proyecto. Finalmente, en Medios verás aparecer el archivo incorporado.

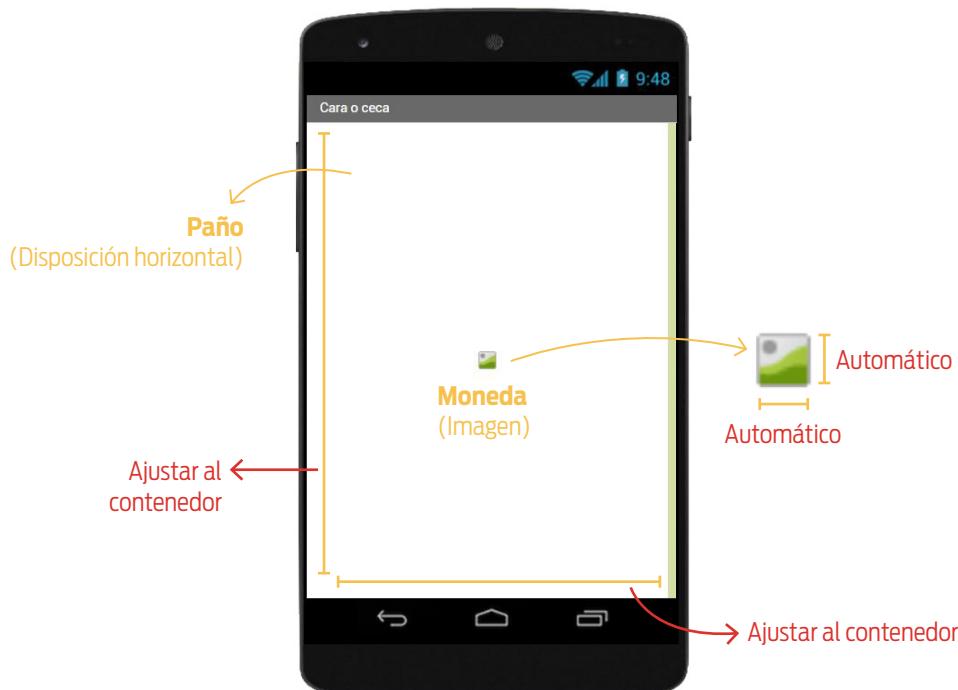


NOMBRE Y APELLIDO:

CURSO:

FECHA:

2. Ahora armá la interfaz gráfica. En la imagen están los componentes y sus dimensiones. En la tabla, las propiedades que tenés que cambiar para que se vea igual a la que te mostramos. Seguí las indicaciones para componerla.



NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen1	Cara o ceca
Paño	Disposición horizontal	Disposición horizontal	Izquierda	Centro
		Disposición vertical	Arriba	Centro
		Alto	Automático	Ajustar al contenedor
		Ancho	Automático	Ajustar al contenedor
Moneda	Imagen	Foto	Ninguno	inicio.png
		Escalar foto al tamaño máximo	(ninguno)	✓

Ejecutá la aplicación en tu teléfono. ¿Qué aparece en la pantalla?

NOMBRE Y APELLIDO:

CURSO:

FECHA:

### EXPRESIONES BOOLEANAS

En Ciencias de la Computación, una expresión booleana es una construcción sintáctica que, cuando se evalúa, o bien es verdadera o bien es falsa.

Por ejemplo,  $5 > 2$  es (verdadera) y  $3 = 4$  (falsa).



- 3.** ¡Llegó el momento del azar! Al agitar el teléfono, en la pantalla tiene que aparecer con idéntica probabilidad, o bien el lado cara de la moneda o bien el lado ceca. ¿Cómo lo conseguiste?

---

---

---

### AYUDA

Explorá el entorno en busca de algún bloque que te permita generar números aleatorios. Tené en cuenta que vas a necesitar una instrucción que te permita hacer una cosa si una cierta condición es verdadera y, otra, si es falsa.



- 4.** Por último tenés que lograr que, al zarandearlo, el teléfono vibre y emita el sonido que se produce al lanzar una moneda. ¿Agregaste algún componente no visible? ¿Para qué?

---

---

---

### ¿SABÍAS QUE...

... el uso de números al azar es muy común en el universo de la computación? Muchas veces se tiene que simular un proceso con un final incierto, para lo cual este tipo de números es muy útil. Por ejemplo, programas para jugar a la generala o al truco.





## Secuencia Didáctica 2

# MENTE MAESTRA

Es habitual que un programa tenga que llevar a cabo muchas veces seguidas una misma serie de acciones. Repetir en forma explícita una secuencia de instrucciones resulta incómodo y, lo que es más grave, aumenta considerablemente el riesgo de introducir errores en los programas. Para evitar estas complicaciones, los lenguajes de programación proveen comandos que denotan **repeticiones** de instrucciones sin que haga falta reiterarlas en forma explícita.

Además, en muchas ocasiones hace falta recordar y modificar valores a lo largo de un programa; por ejemplo, en los juegos, hay que llevar registro de los puntos alcanzados por un jugador. En esta secuencia didáctica, también, se presentan las **variables**, que permiten almacenar valores en la memoria de la computadora, leerlos y modificarlos.

Con el objetivo de introducir repeticiones y variables, se proponen actividades para desarrollar una versión simplificada de Mente maestra, un juego de dos jugadores en el que uno arma un código secreto con fichas de colores que su oponente buscará descubrir.

.....  
**OBJETIVOS**

- Introducir el uso de variables.
  - Presentar las listas como estructura de datos.
  - Utilizar repeticiones para resolver problemas.
- .....

## Actividad 1

### Mente maestra, desenchufada



TODA LA CLASE

#### OBJETIVOS

- Ejercitarse en el razonamiento lógico deductivo.
- Presentar procesos repetitivos.

#### MATERIALES

Pizarrón

Hojas

Fibras rojas, verdes, azules y amarillas

Anexo “Mente maestra, la aplicación”

#### DESARROLLO

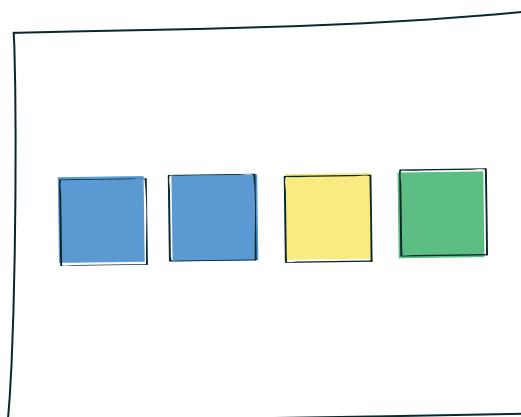
Los objetivos de esta actividad son: (i) poner en práctica mecanismos de deducción lógica y (ii) presentar procesos repetitivos. En primer lugar, se propone jugar grupalmente a una versión simplificada del juego Mente maestra (habitualmente conocido como *Mastermind*, por su nombre en inglés) en modo desenchufado –sin dispositivos electrónicos-. Este juego permite ejercitarse en el razonamiento lógico deductivo. A continuación, se indagará sobre los mecanismos que gobiernan la dinámica del juego, lo que dará pie para presentar nociones de repetición.

#### Parte 1: Mente maestra desenchufada

Para que comprendan la dinámica del juego, les propondremos a los estudiantes jugar al Mente maestra grupalmente. Se trata de un juego de mesa de dos jugadores en el que uno arma un código secreto con cuatro fichas de colores y su oponente busca descubrirlo. Nosotros elegiremos el código y los estudiantes asumirán el rol del contrincante, con el fin de descifrarlo en la menor cantidad de intentos posible.

#### Primer paso: selección del código secreto

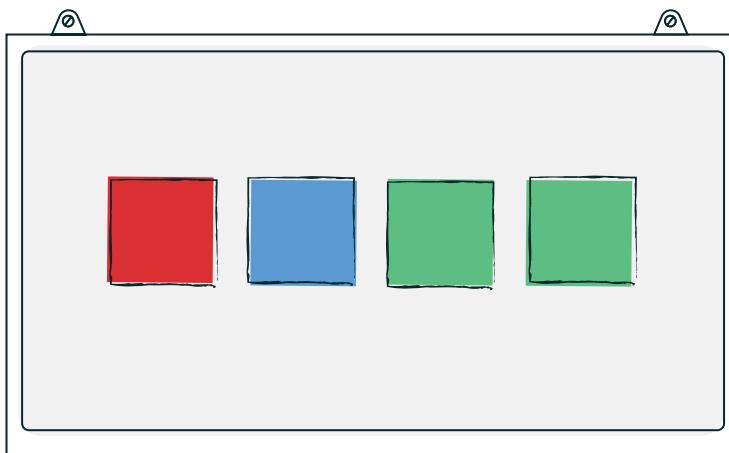
Para dar comienzo al juego elegimos un código de longitud cuatro, al que a cada posición le asignamos un color entre azul, rojo, amarillo y verde, y lo anotamos en una hoja sin que ningún estudiante pueda observarlo. Algunos ejemplos son: [verde, verde, rojo, azul], [amarillo, rojo, azul, verde] y [rojo, rojo, rojo, rojo]. Como el código tiene 4 posiciones y para cada una hay cuatro opciones de color, existen 256 códigos posibles ( $4^4 = 256$ ). A modo de ejemplo, supondremos en la explicación que hemos elegido [azul, azul, amarillo, verde].



Hoja con el código secreto de cuatro colores

### Segundo paso: intento de quebrar el código

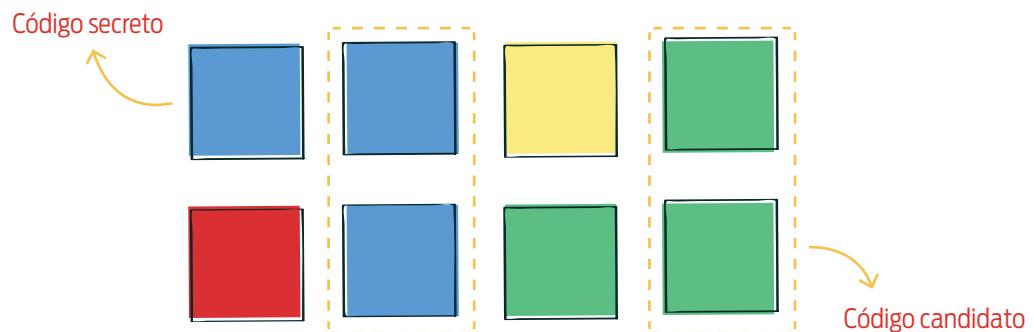
Le pedimos a algún estudiante que intente descubrir el código. Como a esta altura no cuenta con ninguna información sobre nuestra elección, simplemente intentará adivinarlo eligiendo una clave al azar. Una vez que la haya dicho, la copiamos en el pizarrón. Siguiendo con el ejemplo, supondremos que su tentativa es [rojo, azul, verde, verde].



Código de cuatro colores propuesto por un estudiante

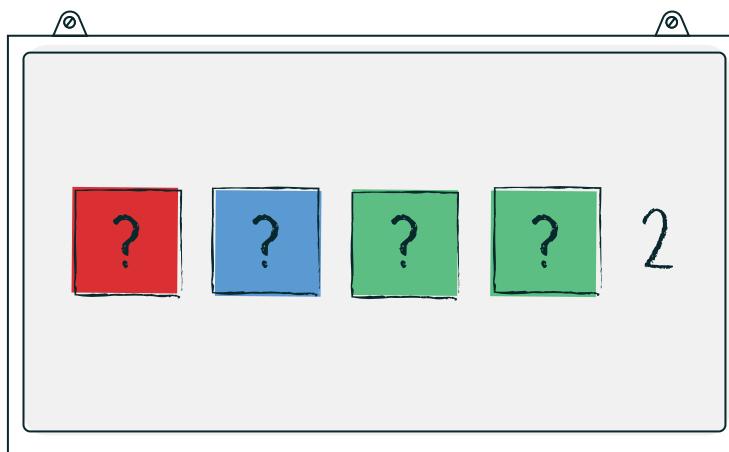
### Tercer paso: proveemos feedback

En esta instancia les informamos a los estudiantes la cantidad de posiciones de la clave propuesta que coinciden con el código secreto elegido por nosotros, sin revelar cuáles son los aciertos –en el ejemplo, el azul de la segunda posición y el verde de la cuarta–: “Hay dos coincidencias”.



Dos coincidencias

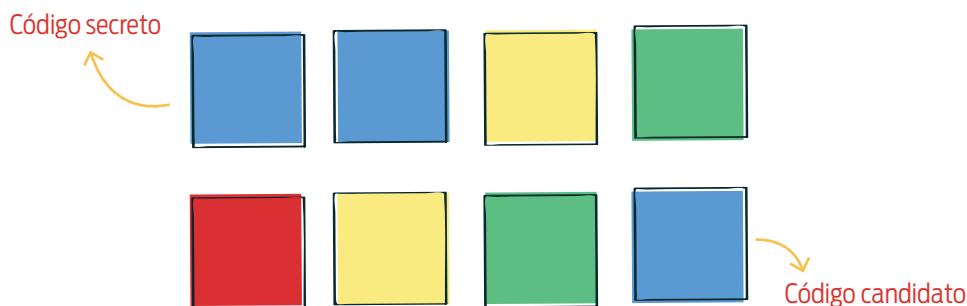
Seguidamente, anotamos el número de coincidencias en el pizarrón. Además, como no hay certeza sobre cuáles son las posiciones adivinadas, colocamos sobre todas un signo de interrogación.



Se copia en el pizarrón la cantidad de coincidencias

De aquí en adelante, se repiten los pasos dos y tres: los estudiantes arriesgan códigos y nosotros informamos la cantidad de aciertos. La diferencia con lo anteriormente descrito es que, en los siguientes intentos, pueden usar la información recabada en las manos previas. El objetivo es descubrir el código en la menor cantidad de intentos posibles.

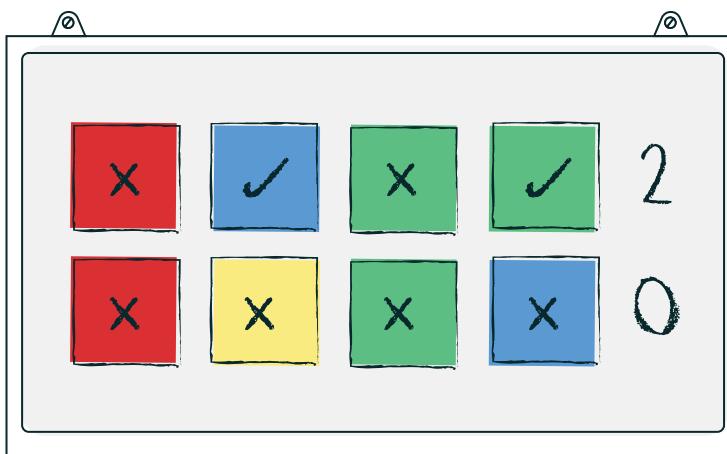
Continuando el ejemplo, los estudiantes podrían especular (erróneamente) que los dos aciertos corresponden a las posiciones 1 y 3 y arriesgar el código [rojo, amarillo, verde, azul] (solo modifican los colores de las posiciones 2 y 4 respecto de su intento anterior). Entonces, informamos que allí no hay ninguna coincidencia.



No hay coincidencias

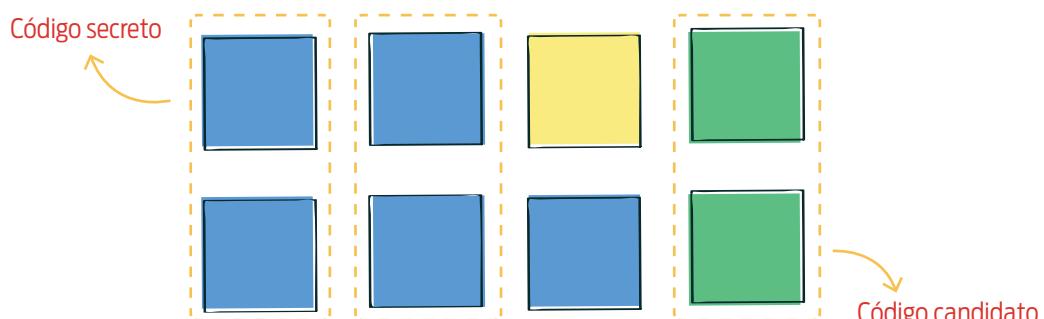
En esta instancia los estudiantes pueden empezar a usar el *feedback* recibido para deducir algunos colores del código secreto y descartar otros. Razonamos con ellos: "En primer lugar, como en el primer intento hubo dos coincidencias y en el segundo ninguna, los aciertos de la primera propuesta corresponden necesariamente las dos posiciones que modificaron en la segunda. Es decir, la segunda posición del código secreto es azul y la cuarta verde. En segundo lugar, también saben que la primera posición no es roja y la tercera no es verde".

Al copiar nuevos códigos en el pizarrón, borramos los signos de interrogación y marcamos las posiciones de las que se tiene certeza sobre su color y las opciones descartadas, de forma que los estudiantes fijen su atención en lo que falta descubrir.



Se informa la cantidad de coincidencias

A continuación podrían arriesgar, por ejemplo, el código [azul, azul, azul, verde]. En este caso, hay tres coincidencias. Sin embargo, no pueden aún concluir si la tercera coincidencia es el azul de la primera posición o el azul de la tercera.

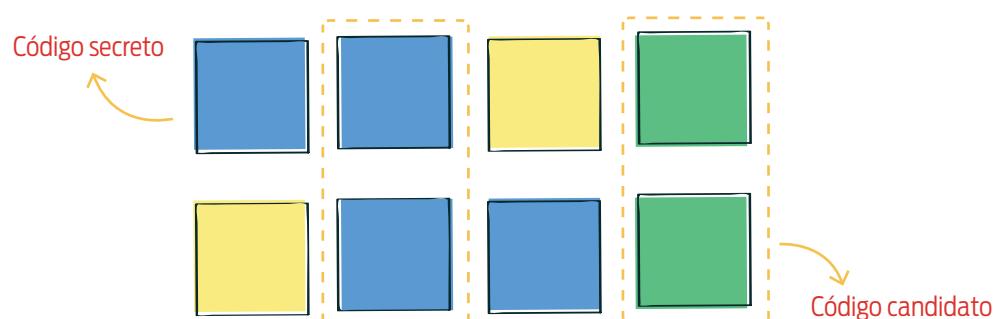


Tres coincidencias

				2
				0
				3

Se informa la cantidad de coincidencias

A partir de la información disponible podrían especular con que la tercera posición es azul y proponer [amarillo, azul, azul, verde], obteniendo dos coincidencias.



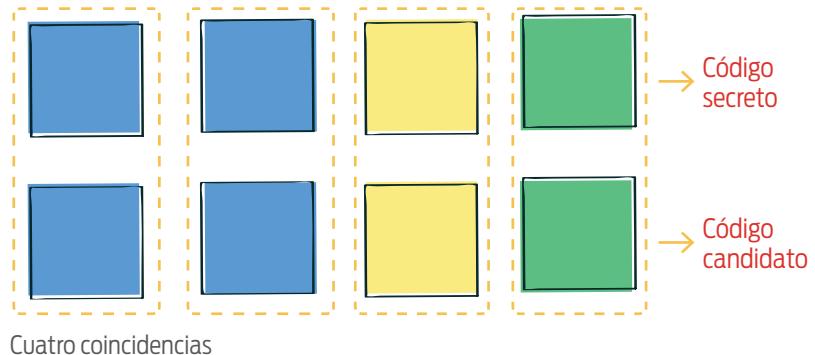
Dos coincidencias

Razonamos con ellos: "Como en este intento hubo dos coincidencias y son las que ya sabían de antemano –la posición dos azul y la cuatro verde–, la tercera coincidencia del intento previo es necesariamente el azul de la posición uno –el único que cambió entre las últimas dos propuestas–. Además, la tercera posición no puede ser ni verde –ya lo sabían– ni azul".

				2
				0
				3
				2

Se informa la cantidad de coincidencias

Solo resta conocer el color de la tercera posición. Las únicas dos opciones son que sea roja o amarilla. Con un poco de fortuna, en esta ocasión podrían proponer el código [azul, azul, amarillo, verde], logrando descifrar el código en el quinto intento.



<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	3
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	2
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	4

El código es descubierto

Los invitamos a que, en parejas, jueguen algunas partidas de Mente maestra. Después, les entregamos el anexo “Mente maestra, la aplicación” y les contamos que allí se describe una aplicación que ellos construirán luego de completar una serie de actividades.

### Parte 2: Repeticiones

Comenzamos la segunda parte de la actividad simulando rememorar una anécdota: “Tengo un amigo que tiene cinco hijos con un año de diferencia entre el más grande y el segundo, un año entre el segundo y el tercero y así también con los otros. ¡Cinco años seguidos tuvo un hijo! No recuerdo muy bien a qué venía, pero el otro día me comentó que cuando sus hijos eran chiquitos, hacía lo siguiente: **para cada hijo**, seleccionaba la ropa, lo despertaba, lo vestía y lo mandaba a lavarse los dientes. Repetía esto 5 veces, una vez por hijo. Qué trabajo, ¿eh?“.

Continuamos: “Volvamos a lo nuestro”. Les proponemos a los estudiantes que imaginen que en un partido de Mente maestra ellos son los que escriben el código secreto y preguntamos: “Una vez que su

ponente arriesga un código, ¿cómo harían para proporcionarle feedback?". Les damos unos instantes para que lo piensen y escuchamos sus ideas. Es posible que aparezcan respuestas tales como "les decimos cuántas son iguales", "contamos la cantidad de coincidencias" o similares. "Es cierto, contamos la cantidad de coincidencias. Detengámonos aquí y hagamos el ejercicio de desmenuzar el proceso de conteo. ¿Qué hacemos primero?". Guiamos el intercambio para concluir que lo primero que hacemos, es mirar si el color de la primera posición del código propuesto coincide con el color de la primera posición de nuestro código y que, de ser así, la consideramos para el conteo. A continuación, copiamos en el pizarrón el mecanismo enunciado.

Si el color de la primera posición coincide con el color de la primera posición de nuestro código, entonces la tenemos en cuenta en el conteo.

Análisis de la primera posición

Luego, comentamos: "Aquí hay una estructura que podemos identificar. Primero aparece la palabra *Si*, luego enunciamos una condición que puede ser cierta o falsa, a continuación la palabra *entonces* y, finalmente, lo que hacemos en caso de que la condición sea verdadera. Como ya saben, esta es una alternativa condicional". Agregamos en el pizarrón la descripción para las tres posiciones restantes.

Si el color de la primera posición coincide con el de la primera posición de nuestro código, entonces la tenemos en cuenta en el conteo.

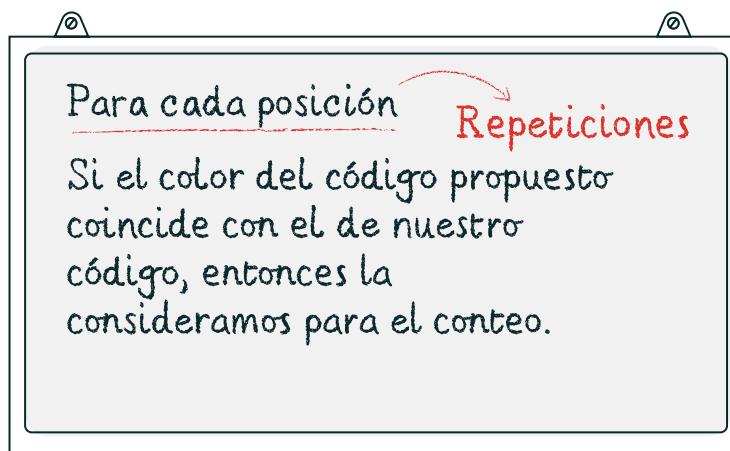
Si el color de la segunda posición coincide con el color de la segunda posición de nuestro código, entonces la tenemos en cuenta en el conteo.

Si el color de la tercera posición coincide con el color de la tercera posición de nuestro código, entonces la tenemos en cuenta en el conteo.

Si el color de la cuarta posición coincide con el color de la cuarta posición de nuestro código, entonces la tenemos en cuenta en el conteo.

Alternativas condicionales

A continuación indagamos a los estudiantes: “Como pueden observar, estamos haciendo cuatro veces cosas análogas, ¿no es cierto? ¿Se les ocurre cómo podrían escribir lo mismo de forma más sintética?”. Escuchamos sus respuestas y, a medida que las enuncian, las analizamos grupalmente. En caso de que no haya surgido, copiamos en el pizarrón la siguiente:



Repeticiones

Preguntamos: “¿Qué diferencias y similitudes encuentran entre lo que escribí antes y lo que escribí ahora?”. Guiamos la discusión para concluir que, aunque se usaron expresiones distintas, ambas connotan que para hacer el conteo solo se consideran las posiciones que comparten el color. Nos aseguramos de que todos hayan comprendido la equivalencia y redondeamos: “Así, obtenemos una forma más concisa de describir el proceso de conteo de coincidencias, ¿no? ¡Imagínense lo que sería escribirlo de la forma anterior si los códigos tuvieran mil posiciones!”.

Finalmente les decimos: “En definitiva, inspeccionamos una colección de elementos –las posiciones de los códigos– y con cada uno de ellos hicimos lo mismo –preguntarnos si había que considerarlos para el conteo de coincidencias–. A este proceso en el que se hace varias veces lo mismo se lo conoce como **repetición**”.

### CIERRE

Les comentamos a los estudiantes que, en general, los lenguajes de programación proveen instrucciones que permiten incorporar en nuestros programas repeticiones. Estas resultan muy útiles cuando hay que ejecutar muchas veces una misma secuencia de instrucciones.

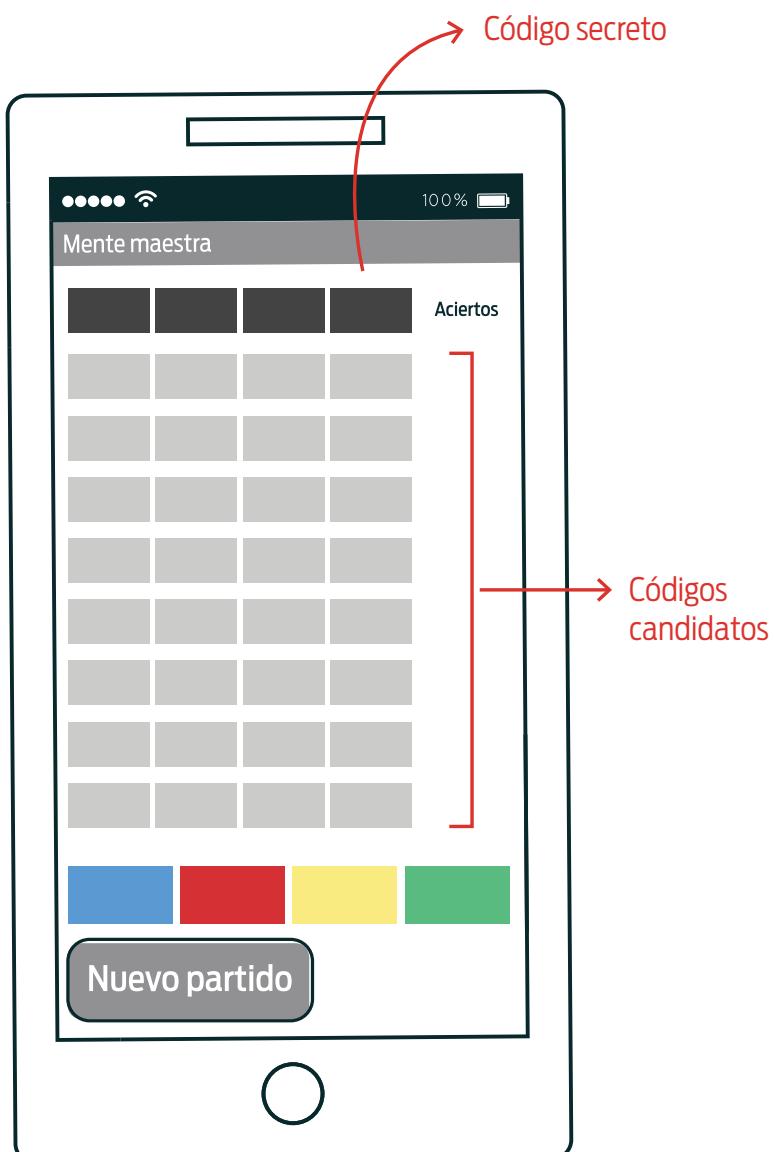
# MENTE MAESTRA, LA APLICACIÓN

¡Mirá esta aplicación del juego Mente maestra! En esta versión, el jugador tiene que estar muy atento y concentrado porque, para adivinar el código secreto, cuenta solo con ocho intentos.

ANEXO

## ¡Comienza el juego!

Esto es lo que se ve en la pantalla apenas la aplicación comienza su ejecución. Los rectángulos en gris oscuro de arriba ocultan el código secreto; y cada una de las ocho líneas siguientes mostrará cada uno de los intentos del jugador por descubrirlo.



### Probando, probando, probando...

Para seleccionar los colores de los códigos candidatos, el usuario presionará los botones azul, rojo, amarillo y verde. A medida que lo haga, irán apareciendo los colores elegidos en los rectángulos reservados para los ocho ensayos. Además, cuando completan un intento, a la derecha se mostrará la cantidad de coincidencias entre el código ingresado y el código secreto.

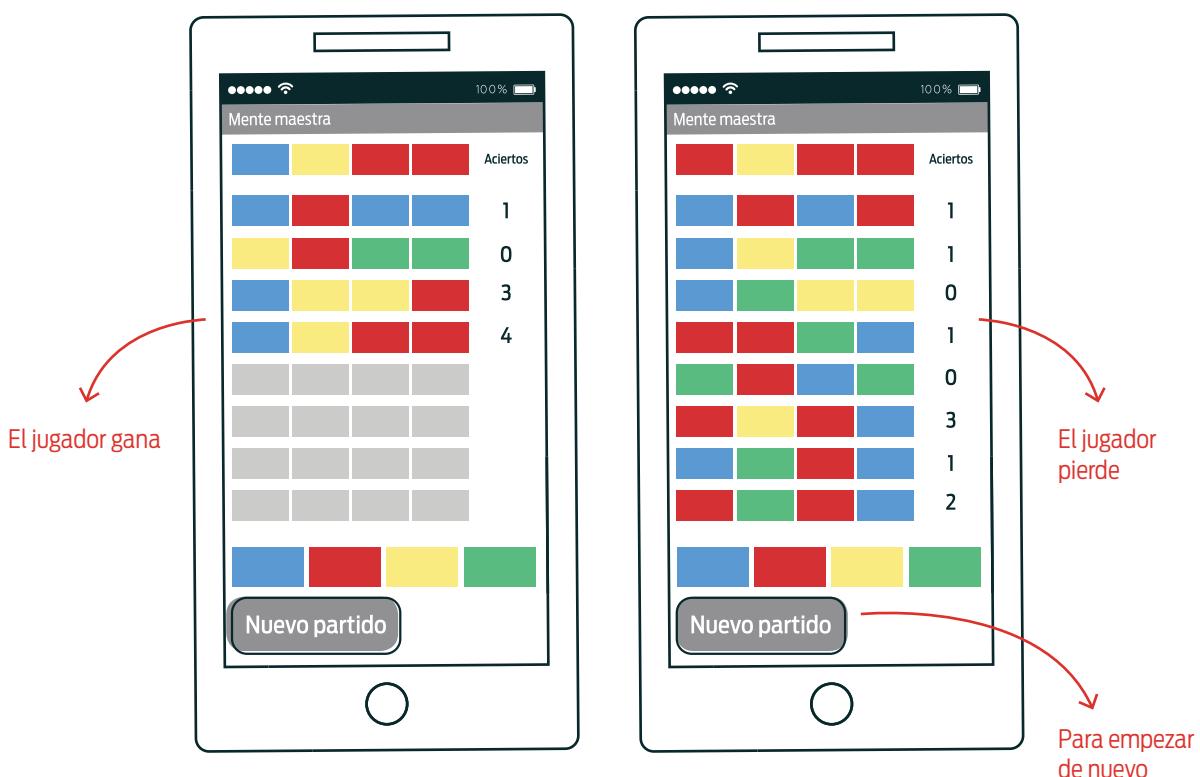
Botones para ingresar códigos candidatos

Número de coincidencias entre el código secreto y el código candidato



### Y en un momento se acaba

El juego finaliza tanto cuando el código secreto es descubierto como cuando se agotan las ocho posibilidades. Por supuesto, en el primer caso el jugador gana y, en el segundo, pierde. Igual, este juego da revancha: presionando el botón Nuevo partido todo comienza otra vez.



## Actividad 2

### Mente maestra, parte I

#### DE A DOS

#### OBJETIVOS

- Introducir el uso de variables.
- Diferenciar variables locales y globales.

#### MATERIALES

 Computadora

 Internet

 MIT App Inventor 2

 Ficha para estudiantes

 Teléfono con Android (opcional)

#### DESARROLLO

Esta es la primera de una serie de actividades para desarrollar una aplicación para jugar al juego Mente maestra. La división en actividades propuesta se debe a que, para completar el programa, hay que incorporar varios temas importantes de programación y, en cada actividad, se hace foco en algunos de ellos.

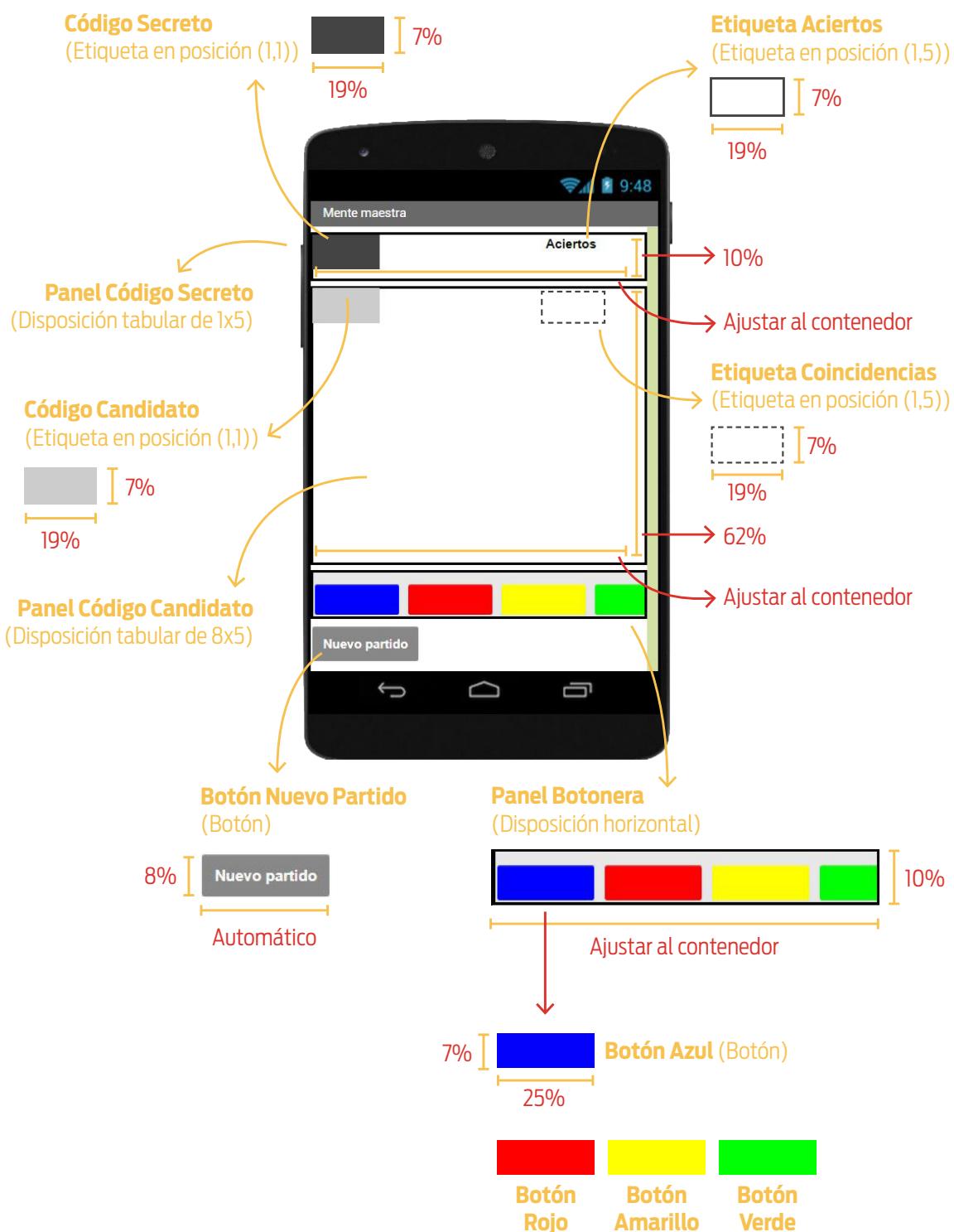
El objetivo de este primer ejercicio es poner de manifiesto la necesidad de usar **variables** al programar. Para ello, los estudiantes construirán un programa para jugar a una versión muy simplificada del Mente maestra: los códigos son de una sola posición –en lugar de cuatro– y los jugadores cuentan con un único intento para descubrirlos. Se trata, en definitiva, de adivinar un color que el programa elige al azar al comenzar su ejecución. Al completar la propuesta, los estudiantes diferenciarán las variables **locales** de las **globales**, advirtiendo cuándo es conveniente usar unas y cuándo las otras.

Es probable que los estudiantes esbozen distintas propuestas para completar la consigna. En el desarrollo de la actividad presentamos y analizamos una de ellas. Sin embargo, cualesquiera sean los programas que elaboren, es importante que *(i)* usen procedimientos para descomponer problemas en subproblemas más simples y para evitar las redundancias, introduciendo parámetros en los casos que sean necesarios, y *(ii)* que incorporen el uso de variables locales y globales.<sup>1</sup>

#### Consigna 1: interfaz gráfica

Comenzamos contándoles a los estudiantes que esta es la primera de una serie de actividades para desarrollar una aplicación para jugar al juego Mente maestra. Luego, les repartimos la ficha y los invitamos a que resuelvan la primera consigna. Allí se dan instrucciones para armar la interfaz gráfica de la aplicación. La tabla a continuación muestra los valores de las propiedades que hay que cambiar (en relación a los que App Inventor asigna por defecto) para que la aplicación se vea tal como en la siguiente imagen.

<sup>1</sup>Una versión completa de la aplicación se encuentra disponible en la galería de proyectos de App Inventor del usuario “programar2020”.



Diseño de la interfaz

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen1	Mente maestra
Panel Código Secreto	Disposición tabular	Columnas	2	5
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	1
Código Secreto	Etiqueta	Color de fondo	Ninguno	Gris oscuro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Aciertos	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	Aciertos
		Posición del texto	Izquierda	Centro
Panel Código Candidato	Disposición tabular	Columnas	2	5
		Alto	Automático	62%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	8
Código Candidato	Etiqueta	Color de fondo	Ninguno	Gris Claro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Coincidencias	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
		Posición del texto	Izquierda	Centro

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Panel Botonera	Disposición horizontal	Disp. horizontal	Izquierda	Centro
		Disp. vertical	Arriba	Abajo
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
Botón Azul	Botón	Color de fondo	Por defecto	Azul
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Rojo	Botón	Color de fondo	Por defecto	Rojo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Amarillo	Botón	Color de fondo	Por defecto	Amarillo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Verde	Botón	Color de fondo	Por defecto	Verde
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Nuevo Partido	Botón	Color de fondo	Por defecto	Gris
		Negrita	(ninguno)	✓
		Alto	Automático	8%
		Texto	Texto para botón	Nuevo partido
		Color de texto	Por defecto	Blanco

Valores diferentes a los asignados por defecto

## Consigna 2: generar el código secreto

La segunda consigna propone que, al iniciar la aplicación, se genere el código secreto que luego se buscará descubrir. Por tratarse de un código de una posición, el problema se reduce a seleccionar al azar uno de los cuatro colores.

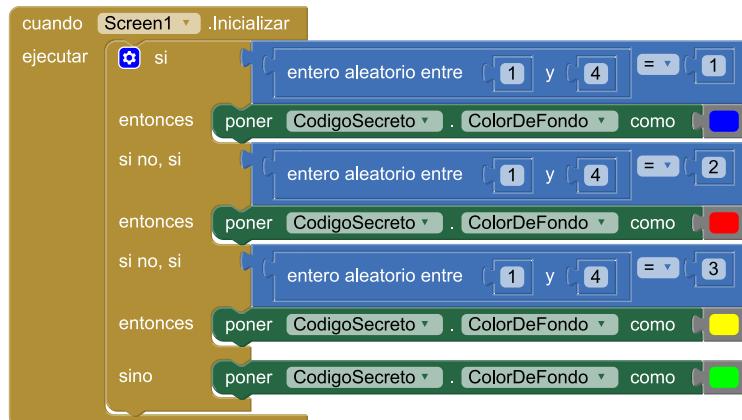
Lo primero que deben reconocer los estudiantes es que tienen que usar el bloque **cuando Pantalla.Inicializar / ejecutar { }** para manejar el evento que se produce cuando la aplicación comienza a ejecutarse. Además, como solo hay cuatro opciones de color –azul, rojo, amarillo y verde–, es esperable que intuyan que para resolverlo tendrán que generar un número al azar entre 1 y 4.

cuando Screen1 .Inicializar  
ejecutar

entero aleatorio entre [1] y [4]

Bloques para resolver la consigna

Algunos estudiantes podrían proponer un programa como el de la figura (o similar, asociando de otra forma los números con los colores). La propuesta contiene dos problemas fundamentales: el modo de selección de color y lo que hace una vez que este fue seleccionado.<sup>1</sup>



Programa con errores

Como puede observarse, para seleccionar uno de los cuatro colores se generan 3 números al azar. De este modo, no todos los colores tienen la misma probabilidad de ser elegidos –*i.e.*, con probabilidad  $\frac{1}{4}$ .<sup>2</sup> Les comentamos: “En realidad, lo que hay que hacer es generar un único número y, de acuerdo al resultado, definir el color secreto. Por ejemplo, si se genera el 1 el color es azul, si se genera el 2 es rojo, etc. Pero

<sup>1</sup> Aunque aquí se presentan juntos, podría ocurrir que haya propuestas que tengan solo uno de los problemas descritos; en tal caso, el programa se puede corregir con las modificaciones propuestas para el error en cuestión.

<sup>2</sup> Si bien aconsejamos no realizar un análisis probabilístico con los estudiantes, se les puede mencionar que de este modo solo el color azul tiene un 25% de probabilidades de ser escogido; la probabilidad de escoger el rojo es de 19% (aprox.), el amarillo 14% (aprox.) y el verde 42% (aprox.).

recuerden: generando un único número". Les damos tiempo para que, autónomamente, exploren el entorno y ensayan alternativas para resolver el problema al que se enfrentan.

La solución al problema planteado requiere el uso de una variable. Por tratarse de un concepto complejo, es esperable que haya estudiantes que no comprendan por sí solos qué son las variables ni cómo se utilizan. En ese caso se puede hacer una revisión grupal, buscando en todo momento que los estudiantes infieran, a partir de los obstáculos que presenta el problema, la necesidad de que un programa "recuerde" valores que luego necesitará recuperar. A continuación, un apartado sobre las variables y, más adelante, la continuación del desarrollo de la actividad.

### Sobre las variables

Una variable es un nombre –que elegimos nosotros al programar– que denota un espacio de la memoria de la computadora en la que se puede guardar un valor para, luego, recuperarlo o modificarlo. Las variables permiten, por ejemplo, que los juegos registren la cantidad de puntos que alcanza un jugador a medida que el juego avanza. Como siempre, es conveniente que el nombre que escogamos refleje su propósito. Siguiendo con el ejemplo de los puntos que obtiene un jugador, podría llamarse *puntos*. En App Inventor, los bloques para manipular variables se encuentran en *Bloques > Integrados > Variables*.



Bloques para manipular variables

Existen dos tipos de variables: locales y globales. Una variable global puede ser referenciada en cualquier punto de un programa –por ejemplo, en distintos procedimientos, en los bloques para manejar eventos, etc.–. Para crear una hay que usar el bloque `inicializar global (nombre) como [ ]` y, allí, asignarle un valor. Este valor es el que adquiere la variable ni bien la aplicación comienza a ejecutarse. Luego, para leer y modificar su contenido hay que usar, respectivamente, los bloques `tomar (nombre)` y `poner (nombre) a [ ]`. Para cambiarle el nombre, alcanza con hacer clic sobre *nombre* y tipear uno nuevo.

En la imagen siguiente se observa un ejemplo en el que se crea una variable `usuario` cuyo valor inicial es un texto vacío, un procedimiento –`iniciarSesión (nombreDeUsuario)`– que modifica su valor; y, por último, otro –`mostrarNombreDeUsuario`– que lee su contenido y lo muestra como texto de una etiqueta.



Definición, escritura y lectura de una variable global

A diferencia de las variables globales, las variables locales solo pueden referenciarse en un espacio restringido del programa. Para inicializar una variable de este tipo se usa el bloque **inicializar local (nombre) como [ ] / en { }**; luego, solo puede actualizarse o leerse su valor dentro del espacio delimitado por **{ }**. A continuación, se muestra un ejemplo con un procedimiento que simula una tirada de una ruleta y, luego de obtener un valor, usa el sintetizador de voz del teléfono para que la aplicación lo anuncie y, además, lo muestra como texto de una etiqueta.



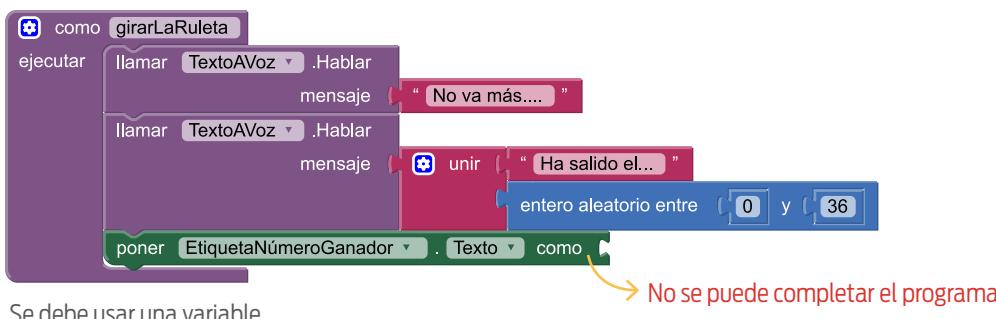
Definición y lectura de una variable local

Al intentar referenciar una variable local fuera de su campo de aplicación, App Inventor muestra que se está en presencia de un error.



Variables locales fuera de su campo de aplicación

A modo de ejemplo, se presenta un análisis de por qué hace falta una variable para producir el efecto buscado en el caso de la simulación de una tirada de la ruleta. Suponiendo que no se use una, hay que generar el número aleatorio en el primer momento en el que se lo utiliza –en este caso, cuando el número es anunciado por el sintetizador de voz–. Sin embargo, como el valor no puede recuperarse de ningún lado, no es posible mostrarlo en la etiqueta. El problema surge al querer utilizar más de una vez un valor sin previamente haberlo conservado.



### SIEMPRE ES MÁS SEGURO UTILIZAR VARIABLES LOCALES QUE GLOBALES

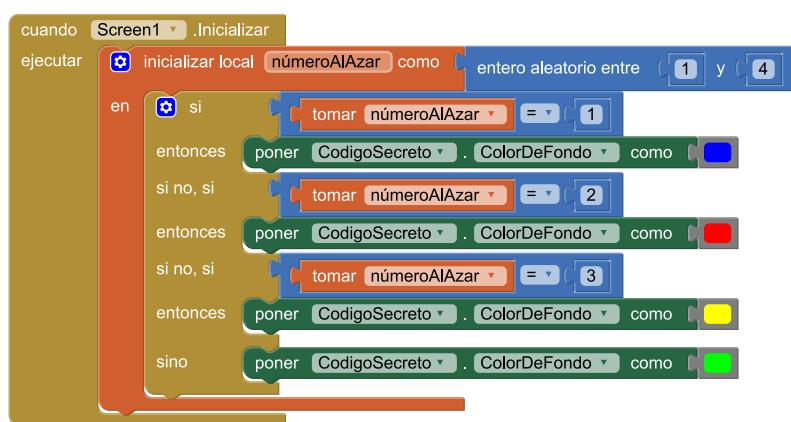
Al ser accesibles desde cualquier punto del programa, es difícil tener control sobre lo que pasa con las variables globales y poder hacer asunciones sobre su valor. Esto se potencia cuando la ejecución de distintas porciones de un programa se dispara por acciones del usuario –como sucede con App Inventor– que, *a priori*, no se sabe cuándo van a ocurrir. Las variables globales hay que usarlas únicamente cuando sea necesario acceder a ellas en distintas partes del programa.



Por el contrario, las variables locales tienen un ámbito de aplicación restringido a una pequeña porción del programa, lo que permite que, al momento de programar, sepamos cómo evolucionará su valor cuando la aplicación se encuentre en ejecución.

### Consigna 2: generar el código secreto (continuación)

El problema que se produce al generar tres números aleatorios distintos para la selección de un color puede resolverse usando una variable local: se genera un único número y, luego, se consulta su valor. De este modo, los cuatro colores tienen la misma probabilidad de ser seleccionados.

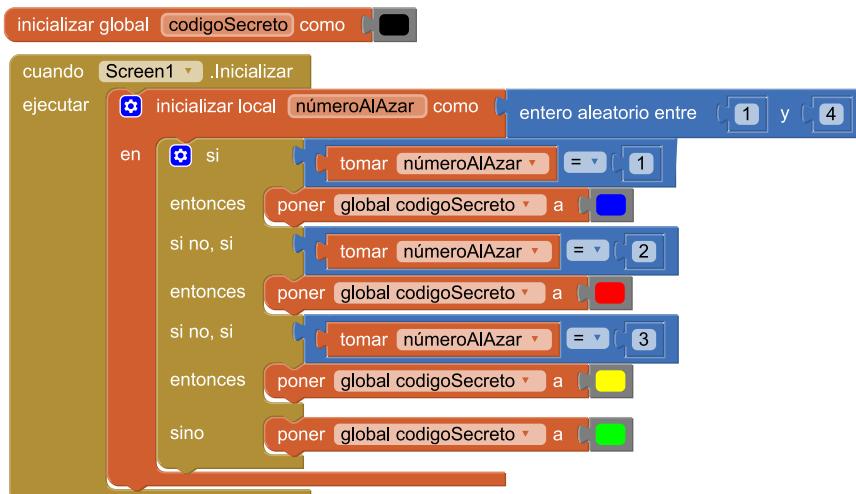


Se genera un único número al azar

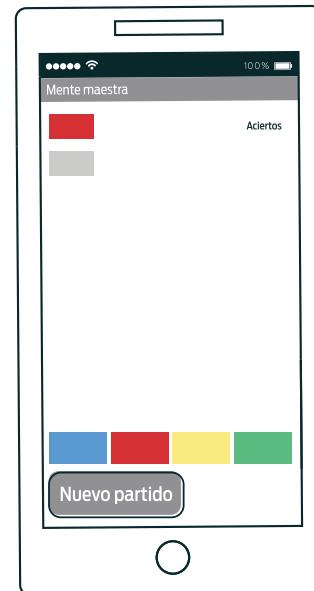
El segundo problema de la propuesta es que el color secreto, una vez definido, se lo establece como color de fondo de la etiqueta que representa la incógnita. De este modo, en lugar de conservarse oculto, el color queda expuesto ni bien la aplicación comience a ejecutarse. Si algún estudiante propone un programa con esta característica, le sugerimos que lo corra; así, podrá observar cómo se manifiesta el error y pensar algún modo de corregirlo.

En esta primera aproximación al Mente maestra, la interacción entre la aplicación y el usuario involucra tres pasos: (i) el programa selecciona un color al azar, (ii) el usuario presiona uno de los cuatro botones intentando adivinar el color secreto, y (iii) el programa compara ambos colores y muestra el resultado. Por lo tanto, el color seleccionado aleatoriamente cuando comienza la ejecución de la aplicación debe ser recordado para evaluar, tiempo más tarde, cómo le fue al jugador. En este caso, sí, hace falta una variable global: hay que conservar un valor generado en un lugar –donde se maneja el evento que se produce cuando comienza la ejecución– y usarlo en otro –donde se maneja el evento que se produce cuando el usuario selecciona un color–.

En la imagen se exhibe una posible solución del desafío, que incluye una variable global: `codigoSecreto`. Si bien se inicializa con el color negro, podría inicializarse con cualquier otro valor, pues apenas la aplicación comienza su ejecución el negro se reemplaza por el color seleccionado al azar.

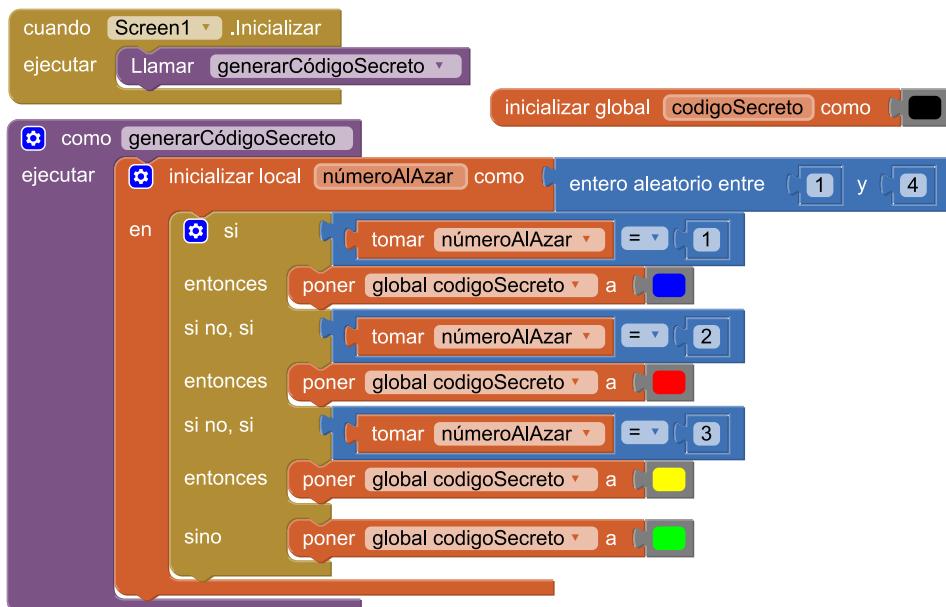


Solución sin procedimientos



Al iniciar la aplicación el código secreto queda expuesto

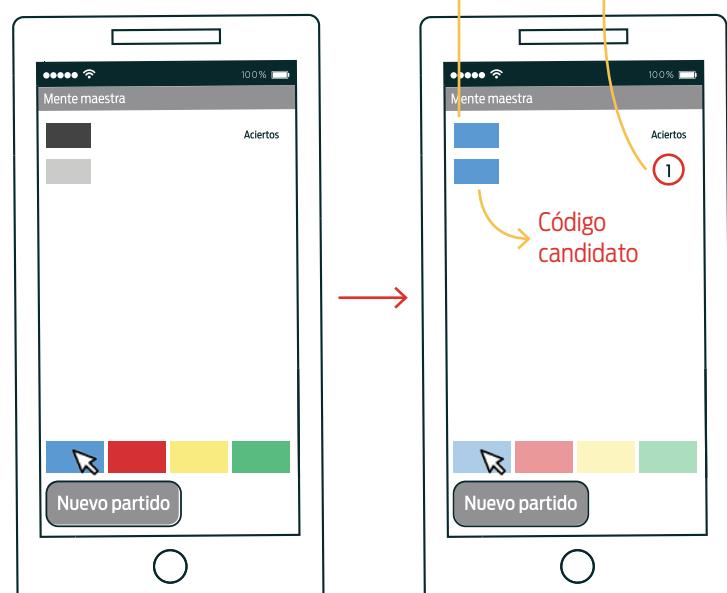
Es importante insistirles a los estudiantes sobre la ventaja de usar procedimientos (y escoger cuidadosamente sus nombres) para construir programas claros y fácilmente entendibles. A continuación se muestra una solución alternativa que define un procedimiento, en la que se aprecia a simple vista que, al iniciar la ejecución, la aplicación genera el código secreto; además, cómo hace para generararlo, queda encapsulado en un procedimiento separado.



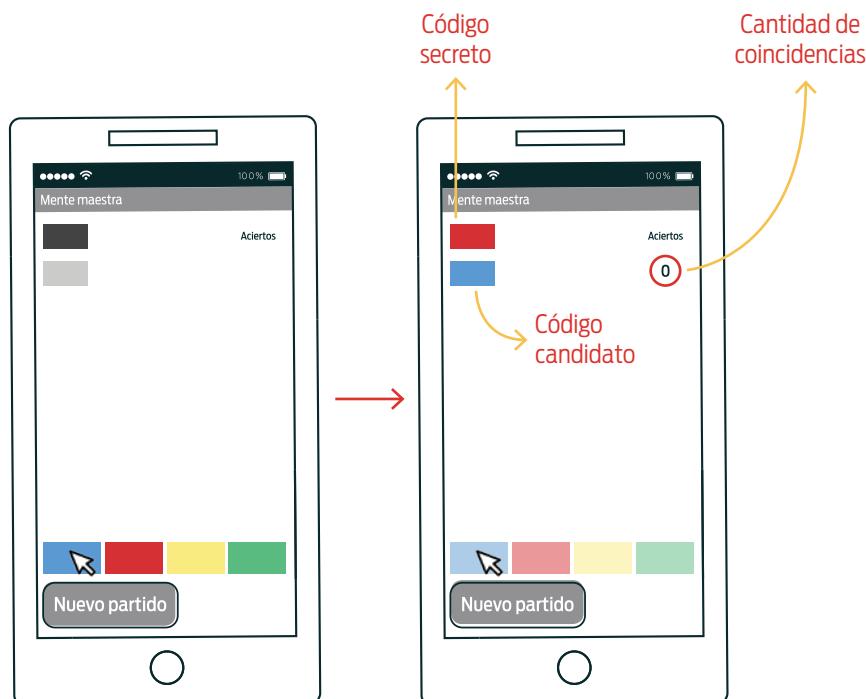
Solución con un procedimiento

### Consigna 3: manejar el evento del botón azul

La tercera consigna pide que, luego de que el usuario presione el botón azul, se muestre si hay o no una coincidencia entre la elección del jugador y el color secreto. A continuación se muestra la respuesta que se espera del programa, tanto si hay coincidencia como si no, en un caso asumiendo que el color secreto es azul y en el otro que es rojo.



Hay coincidencia



No hay coincidencia

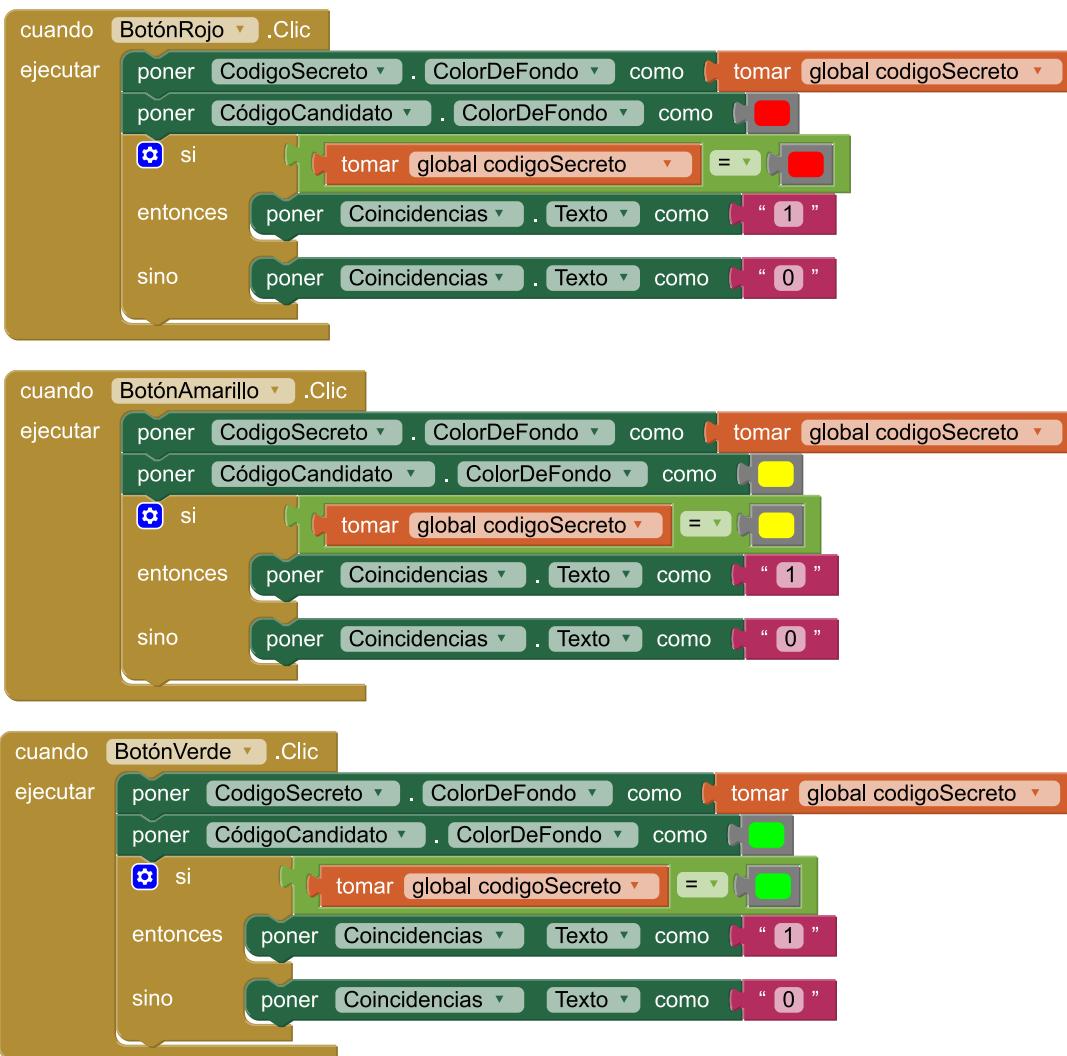
Cuando se presiona el botón azul hay que: (i) mostrar el color del código secreto, (ii) mostrar el color elegido por el usuario –azul–, y (iii) indicar la cantidad de coincidencias entre ambos códigos –1 si coinciden y 0 si no–. La figura muestra una posible solución.



Possible solution of the third challenge

#### Consigna 4: manejar el evento de los botones rojo, amarillo y verde

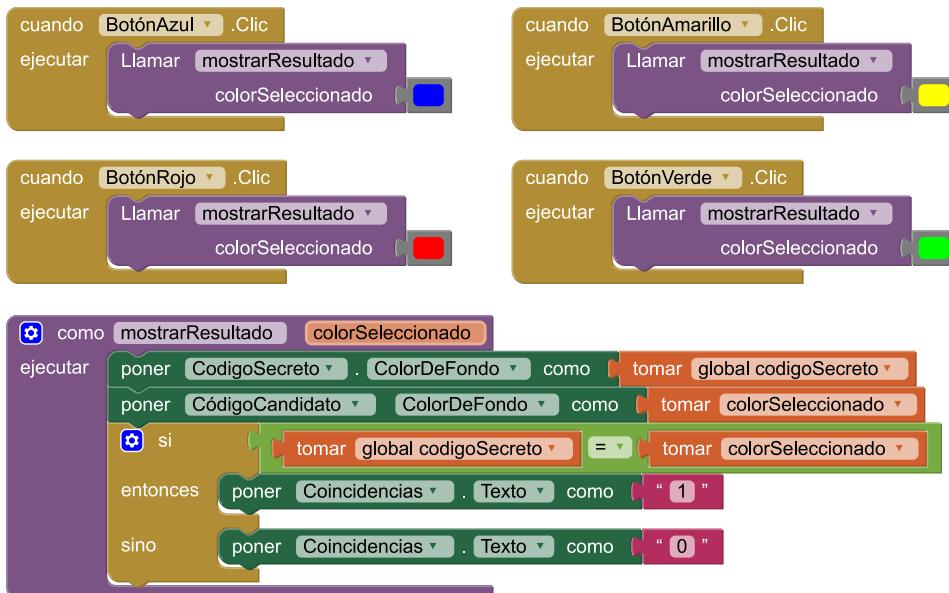
The fourth challenge asks to incorporate the handling of events that occur when pressing the red, yellow and green buttons. It could happen that some students propose a program where the handling of these three events is very similar to what was programmed for the blue button, changing each one the considered color.



Fragmentos similares del programa

Si bien el programa es funcionalmente correcto, no es una solución cualitativamente buena. A los estudiantes que optasen por un programa con esta característica, les sugerimos que piensen algún programa alternativo.

La presencia de fragmentos muy parecidos sugiere que es conveniente incorporar un procedimiento en el que el color seleccionado por el usuario sea un parámetro. A continuación, se da una solución de la consigna que incorpora el procedimiento `mostrarResultado (colorSeleccionado)`.



Solución de la cuarta consigna

### Consigna 5: deshabilitación de la botonera de colores

La cuarta consigna propone que, una vez que el usuario arriesga una alternativa y se devela el color secreto, se deshabiliten los cuatro botones de colores. De esta forma, se da por finalizada la jugada.

Con este propósito hay que establecer la propiedad `Botón.Habilitado` con el valor `falso` (disponible en *Bloques > Integrados > Lógica*). Nuevamente, es conveniente definir un procedimiento que se encargue específicamente de esta tarea.



Procedimiento `deshabilitarBotonesDeColores`

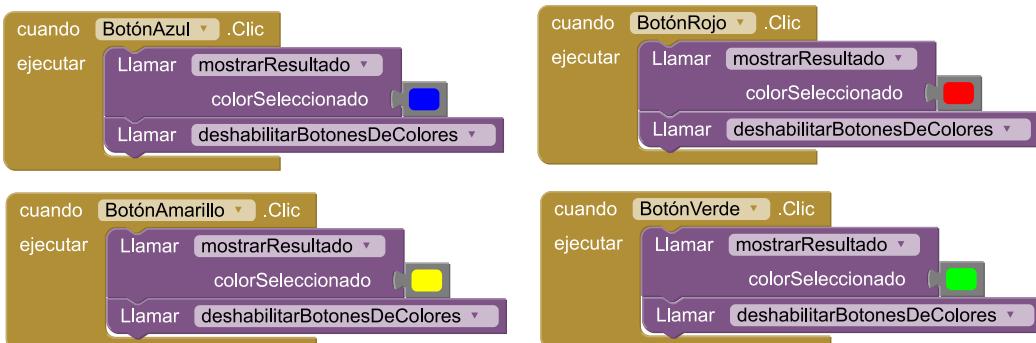
Una pregunta válida es: ¿desde dónde hay que invocar al procedimiento `deshabilitarBotonesDeColores`? Existen distintas alternativas que alcanzan el objetivo propuesto. A continuación se muestran tres, ordenadas por calidad, de menor a mayor.

La primera es hacerlo luego de mostrar el resultado, como última instrucción del procedimiento `mostrarResultado (colorSeleccionado)`.



Invocación a `deshabilitarBotones` dentro de `mostrarResultado`

El punto débil de esta propuesta es que la deshabilitación de los botones, como unidad de sentido, comprende acciones que no forman parte de mostrar el resultado. Otra opción, más adecuada, es invocar a `deshabilitarBotones` desde los bloques de manejo de los eventos que producen los botones.



Invocación a `deshabilitarBotones` dentro de `mostrarResultado`

Esta propuesta maneja los eventos de los cuatro botones de eventos de forma muy similar. Aunque en este caso la similitud es sutil –cambia el color con el que se invoca `mostrarResultado [colorSeleccionado]`–, una alternativa superadora consiste en crear un nuevo procedimiento que tenga un parámetro que represente el color y se encargue tanto de mostrar el resultado como de deshabilitar los botones. Podría llamarse, por ejemplo, `procesarSelecciónDeColor [colorSeleccionado]`.





Definición de un nuevo procedimiento

### Consigna 6: volver a empezar

La sexta y última consigna solicita que, cuando se presione el botón *Nuevo partido*, se pueda volver a jugar una mano; es decir, que todo quede como al comienzo. Por lo tanto hay que (i) generar un nuevo código secreto, (ii) habilitar los botones de colores, (iii) agrisar las etiquetas que representan ambos códigos –la que simboliza el color secreto y la que muestra la elección del jugador–, y (iv) borrar el texto que muestra la cantidad de coincidencias.

Dado que ya se cuenta con el procedimiento `generarCódigoSecreto`, el desafío puede completarse generando otros dos: uno para habilitar los botones de colores –que podría llamarse `habilitarBotonesDeColores`– y otro para ocultar los colores de las etiquetas que representan los códigos –que podría llamarse `agrisarEtiquetasDeLosCódigos`–.

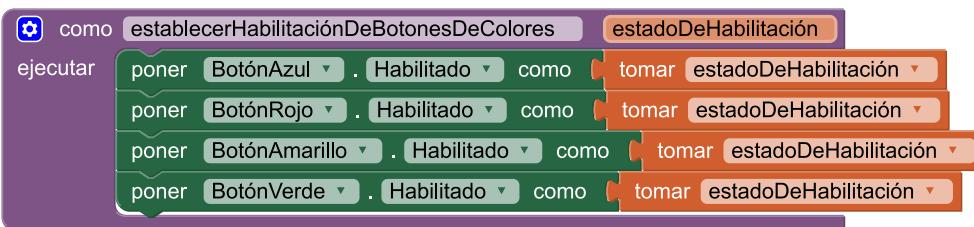
Es probable que algunos estudiantes propongan habilitar los botones de colores de forma similar a como los deshabilitaron en la consigna anterior.



Dos procedimientos muy similares

Nuevamente, por ser ambos procedimientos muy similares, lo conveniente es crear un nuevo procedimiento en el que el estado de habilitación de los botones sea un parámetro. Podría llamarse, por ejemplo, `establecerHabilitaciónDeBotonesDeColores (estadoDeHabilitación)`.





Un procedimiento para habilitar y deshabilitar botones

Para agrisar las etiquetas que representan los códigos basta con establecer la propiedad `Etiqueta.colorDeFondo` de `CódigoSecreto` y `CódigoCandidato` con gris oscuro y gris claro respectivamente.

Finalmente, al hacer clic sobre `BotónNuevoPartido`, los procedimientos son invocados y se borra el texto que muestra la cantidad de aciertos.

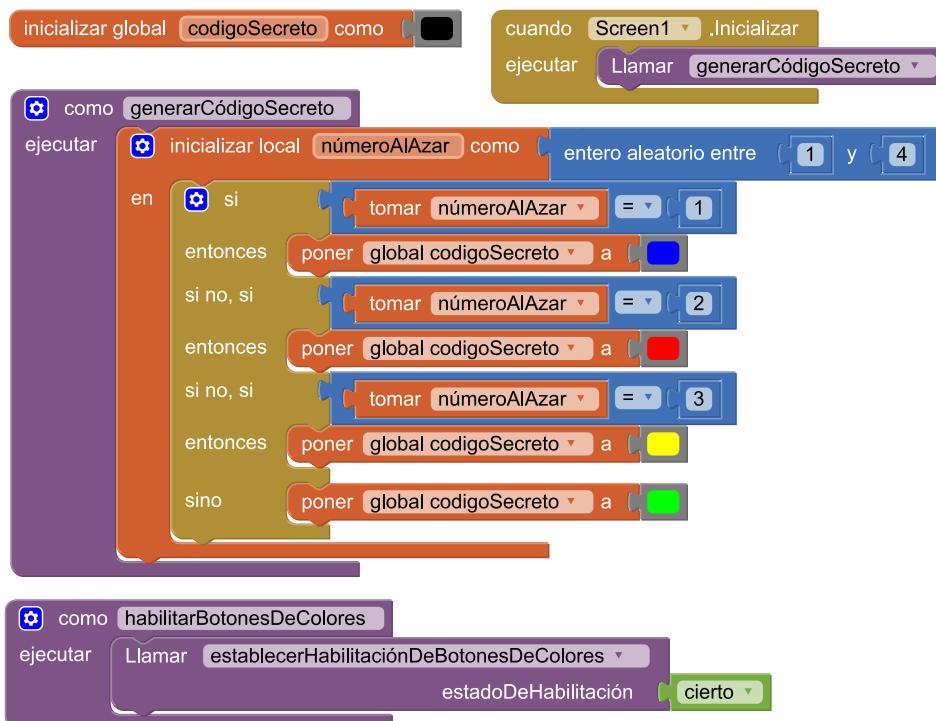


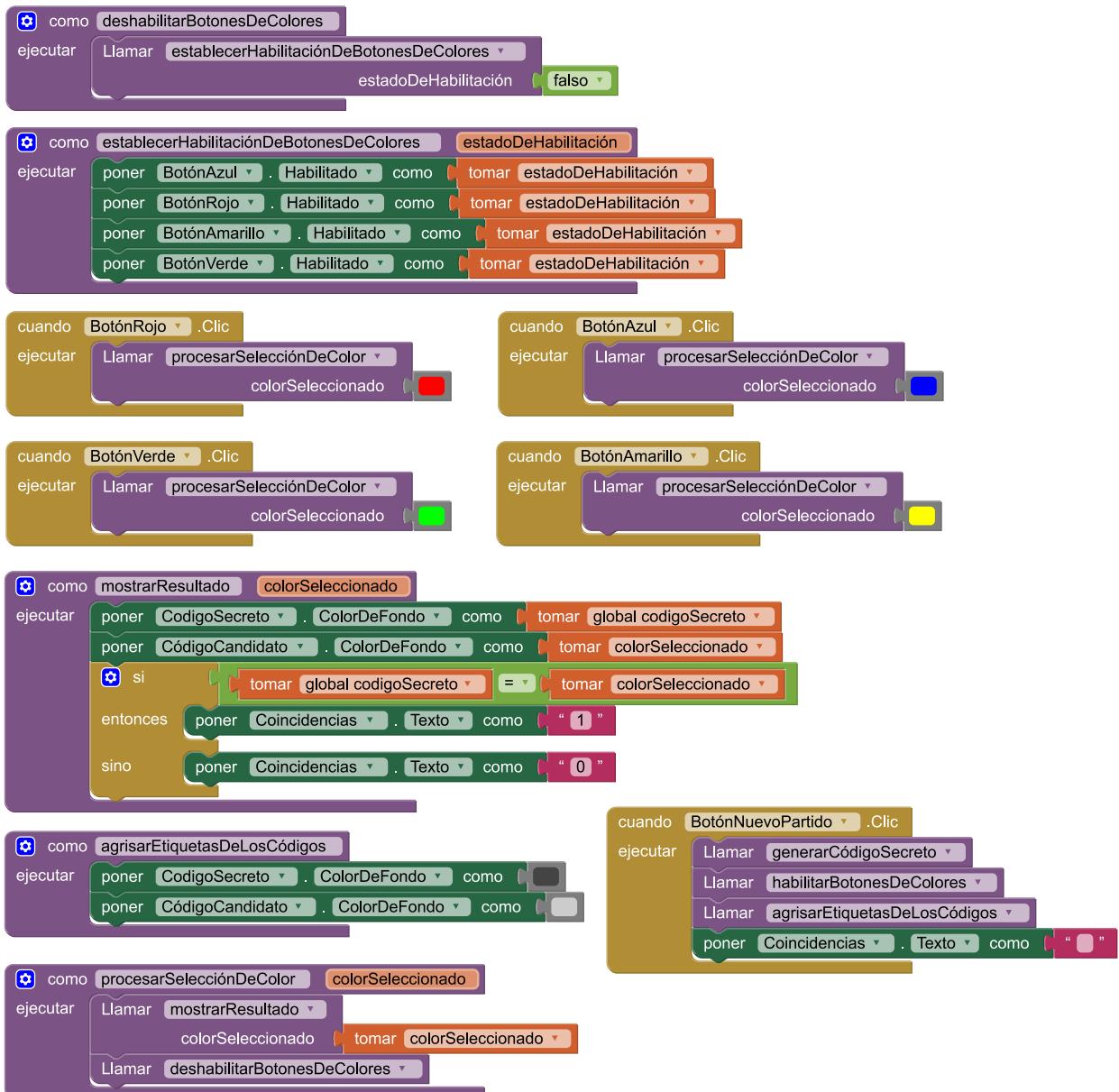
Procedimiento `agrisarEtiquetasDeLosCódigos`



Manejo de evento de `BotónNuevoPartido`

A continuación se muestra una solución completa de la actividad.





Solución completa de la actividad

### Cierre

Repasamos con los estudiantes las posibilidades que dan las variables. Reiteramos que una variable nos sirve para recordar un valor y que, una vez definida, puede leerse y modificarse más adelante. Además, hacemos hincapié en la diferencia entre las globales y las locales: las globales pueden referenciarse en cualquier lugar del programa, mientras que las locales existen en espacios reducidos. Insistimos, por último, en que hay que ser muy cuidadosos al usar variables globales, pues fácilmente podemos perder noción de su evolución a medida que un programa se ejecuta.

NOMBRE Y APELLIDO:

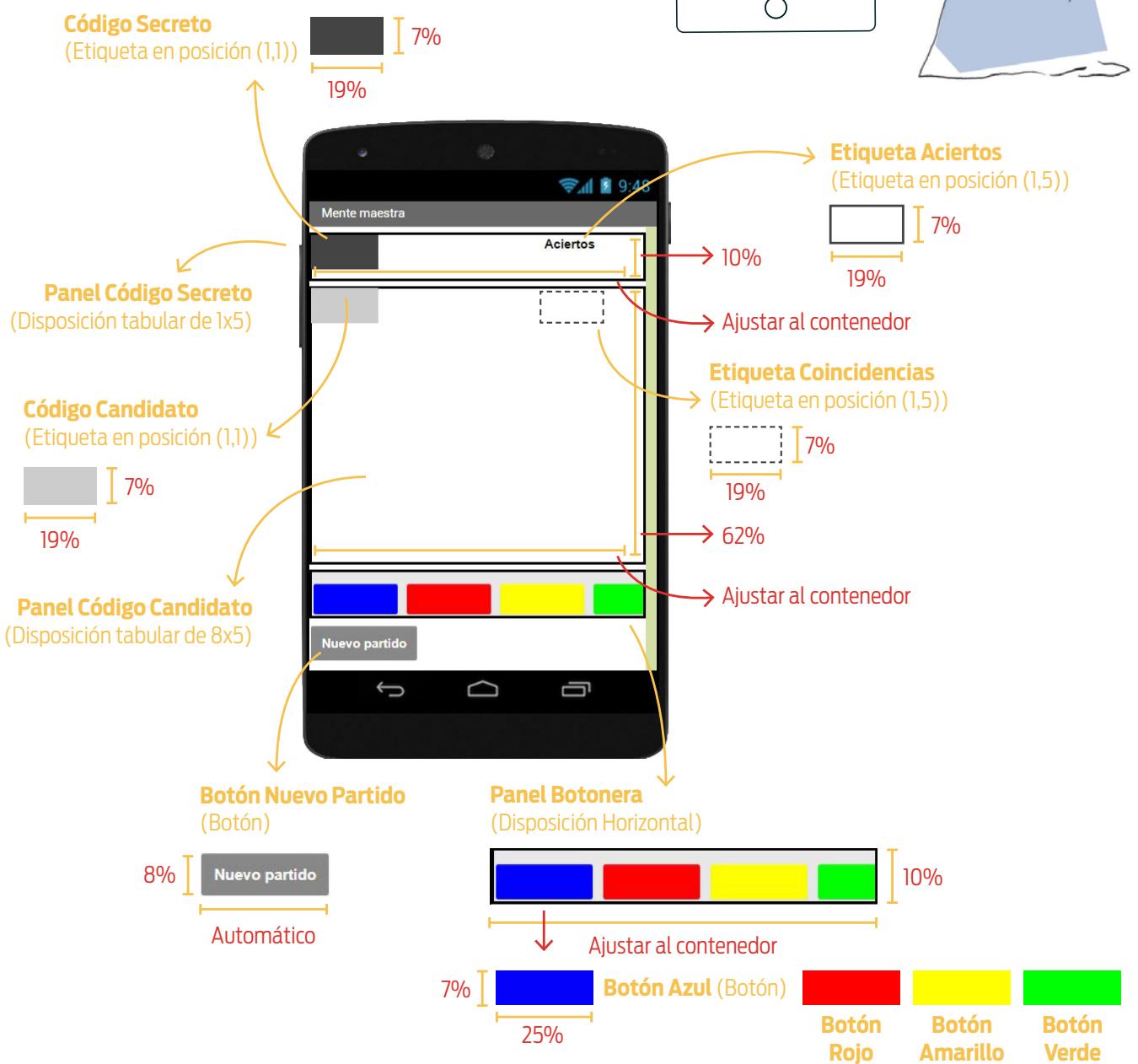
CURSO:

FECHA:

# MENTE MAESTRA, PARTE I

¡Primera versión del juego! Acá, el código secreto es solo un color. Además, hay un solo intento. Para ganar, ¡hay que pegarle de una!

1. Comenzá armando la interfaz gráfica de esta primera aproximación al Mente maestra. Acá abajo podés ver los componentes y sus dimensiones. En la tabla, las propiedades que tenés que cambiar para que se vea igual a la que te mostramos. Seguí las indicaciones para componerla.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen1	Mente maestra
Panel Código Secreto	Disposición tabular	Columnas	2	5
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	1
Código Secreto	Etiqueta	Color de fondo	Ninguno	Gris oscuro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Aciertos	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	Aciertos
		Posición del texto	Izquierda	Centro
Panel Código Candidato	Disposición tabular	Columnas	2	5
		Alto	Automático	62%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	8
Código Candidato	Etiqueta	Color de fondo	Ninguno	Gris Claro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Coincidencias	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
		Posición del texto	Izquierda	Centro

NOMBRE Y APELLIDO:

CURSO:

FECHA:

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Panel Botonera	Disposición horizontal	Disp. horizontal	Izquierda	Centro
		Disp. vertical	Arriba	Abajo
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
Botón Azul	Botón	Color de fondo	Por defecto	Azul
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Rojo	Botón	Color de fondo	Por defecto	Rojo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Amarillo	Botón	Color de fondo	Por defecto	Amarillo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Verde	Botón	Color de fondo	Por defecto	Verde
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Nuevo Partido	Botón	Color de fondo	Por defecto	Gris
		Negrita	(ninguno)	✓
		Alto	Automático	8%
		Texto	Texto para botón	Nuevo partido
		Color de texto	Por defecto	Blanco

NOMBRE Y APELLIDO:

CURSO:

FECHA:

2. Ni bien comienza su ejecución, el juego tiene que generar al azar un código secreto de longitud 1. Se trata, en definitiva, de librarse a la suerte la elección de un color. Tené en cuenta que, más tarde, quien juegue buscará adivinarlo. ¿Qué herramienta usaste para resolver el desafío? ¿Para qué?

---

---

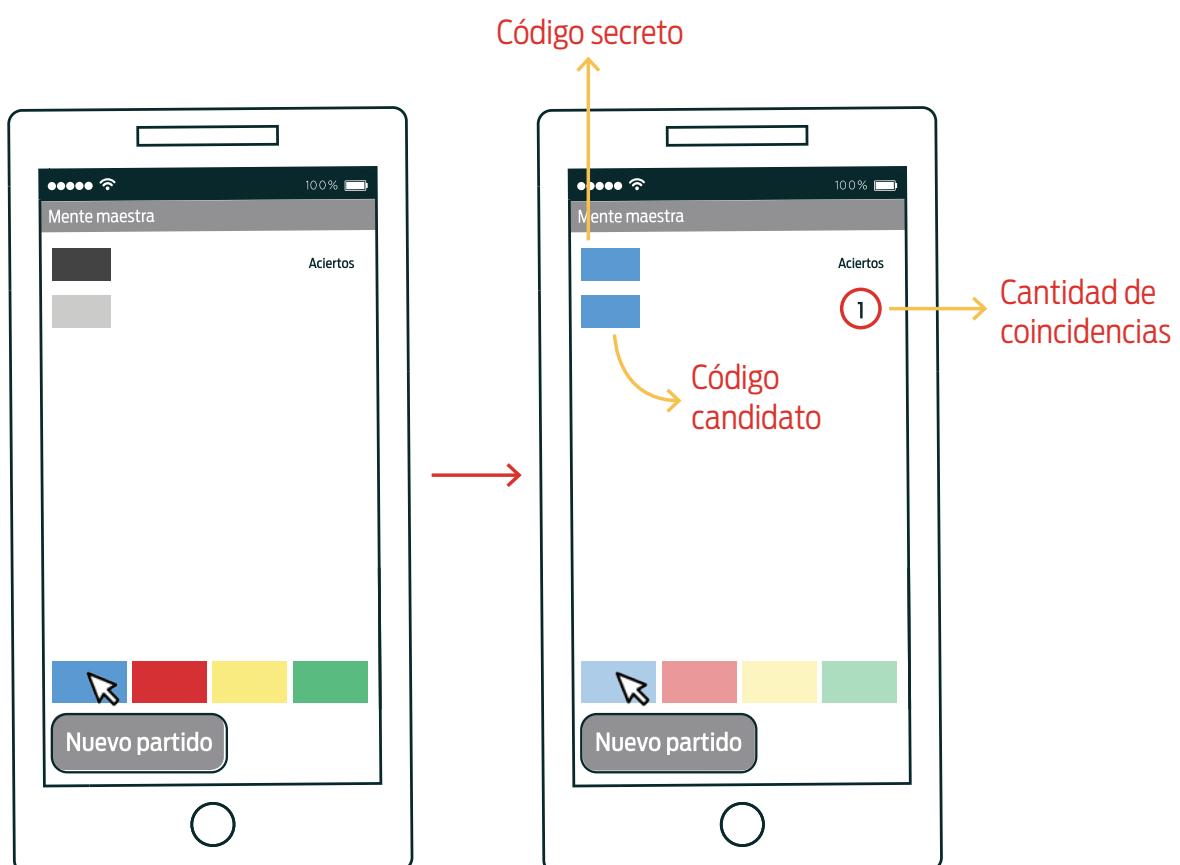
---

**PARA NO OLVIDAR...**

Una variable es un nombre –que elegimos nosotros al programar– que denota un espacio de la memoria de la computadora donde se puede guardar un valor y, luego, recuperarlo o modificarlo. Las variables permiten, por ejemplo, que los juegos registren la cantidad de puntos que alcanza un jugador a medida que el juego avanza.



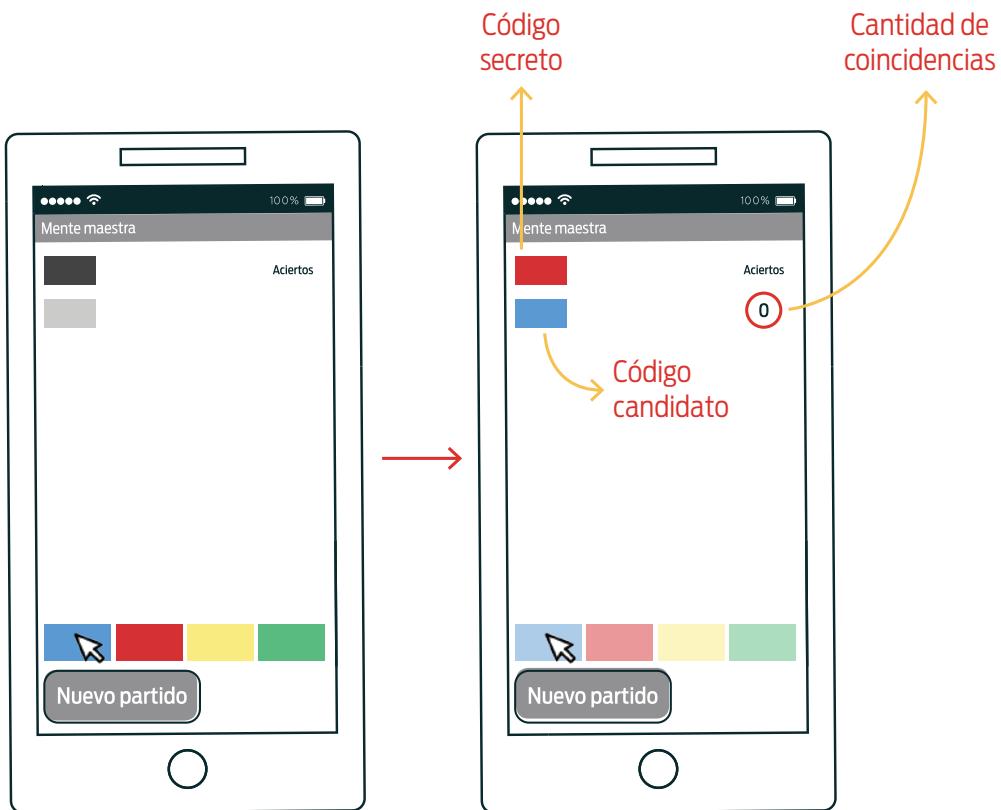
3. Ahora tenés que encargarte de que, una vez que el usuario presione el botón de azul, se muestre si hay o no una coincidencia entre la elección del jugador y el color secreto. Acá podés ver lo que tiene que pasar tanto si hay coincidencia como si no (en la página siguiente).



NOMBRE Y APELLIDO:

CURSO:

FECHA:



4. No solo de apretar el botón azul se trata este juego. También hay que resolver lo que sucede cuando se hace clic sobre el rojo, el amarillo y el verde. Resolvelo evitando que en tu programa haya redundancia. ¿Qué hay que hacer, en general, si cuando programamos observamos que hay distintas partes del programa que son iguales o muy parecidas?
- 
- 

#### PROCEDIMIENTOS PARA SIMPLIFICAR LO COMPLEJO

Tené siempre presente que, al resolver un problema, es conveniente comenzar pensando las partes que componen una solución. En tu programa, cada una de estas partes las podés resolver en un procedimiento separado.

5. A los jugadores vamos a darles una sola oportunidad para que adivinen el color secreto. Una vez hecha la elección, los botones tienen que quedar deshabilitados. ¿Cómo lo conseguiste?
- 
6. Y ahora, todo otra vez. Con un clic sobre el botón *Nuevo partido* hay que dejar todo como al principio, listo para que el jugador pueda intentar otra vez descubrir un nuevo color secreto.

## Actividad 3

### Mente maestra, parte II

 DE A DOS

#### OBJETIVOS

- Construir un programa con repeticiones.
- Introducir el uso de listas.

#### MATERIALES

 Computadora

 Internet

 MIT App Inventor 2

 Ficha para estudiantes

 Teléfono con Android (opcional)

#### DESARROLLO

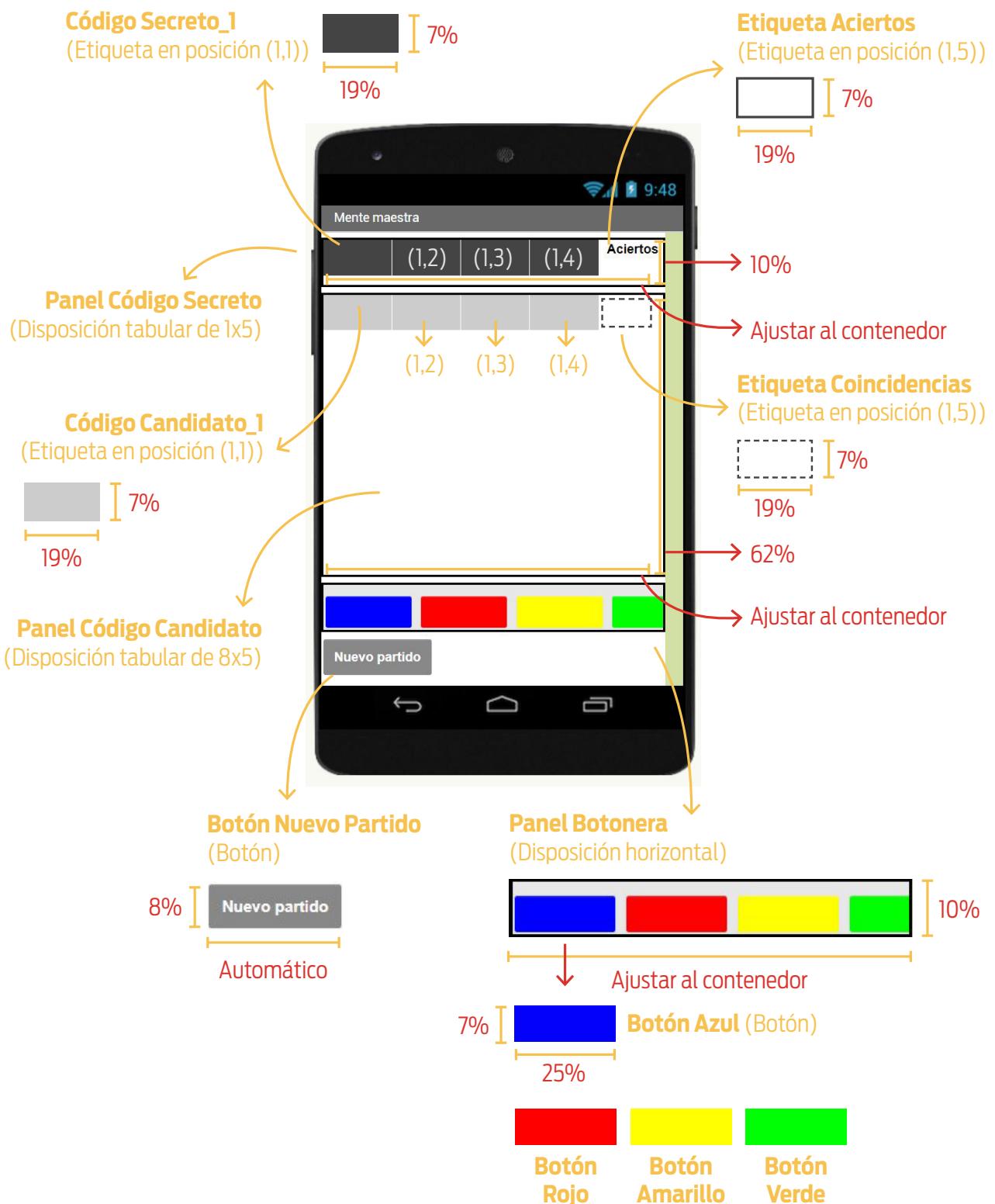
Los objetivos de esta segunda actividad de la serie son (*i*) presentar las listas como estructuras útiles en programación y (*ii*) construir una aplicación que use repeticiones. Para ello, los estudiantes programarán una nueva versión del Mente maestra en la que los códigos son de cuatro posiciones y los jugadores cuentan con un único intento para descubrirlos.

Nuevamente, puede ocurrir que los estudiantes propongan distintas soluciones para completar la consigna. A continuación, desarrollamos y analizamos una de ellas. Sin embargo, cualesquiera sean los programas que elaboren, es importante (*i*) que usen procedimientos para descomponer problemas en subproblemas más simples y para evitar las redundancias, introduciendo parámetros en los casos que sean necesarios; (*ii*) que incorporen el uso de listas; y (*iii*) que utilicen bloques de repetición para evitar copiar, una vez a continuación de otra, una misma secuencia de instrucciones.<sup>1</sup>

#### Consigna 1: interfaz gráfica

Les repartimos la ficha a los estudiantes y los invitamos a que resuelvan la primera consigna. Allí se dan instrucciones para armar la interfaz gráfica de la aplicación. La tabla a continuación muestra los valores de las propiedades que hay que cambiar (en relación con los que App Inventor asigna por defecto) para que la aplicación se vea tal como en la siguiente imagen.

<sup>1</sup> Una versión completa de la aplicación se encuentra disponible en la galería de proyectos de App Inventor del usuario “programar2020”.



NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen1	Mente maestra
Panel Código Secreto	Disposición tabular	Columnas	2	5
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	1
Código Secreto_1 <sup>1</sup>	Etiqueta	Color de fondo	Ninguno	Gris oscuro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Aciertos	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	Aciertos
		Posición del texto	Izquierda	Centro
Panel Código Candidato	Disposición tabular	Columnas	2	5
		Alto	Automático	62%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	8
Código Candidato_1 <sup>2</sup>	Etiqueta	Color de fondo	Ninguno	Gris claro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Coincidencias	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
		Posición del texto	Izquierda	Centro

<sup>1</sup>Las restantes etiquetas del código secreto requieren los mismos cambios.<sup>2</sup>Las restantes etiquetas del código candidato requieren los mismos cambios.

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Panel Botonera	Disposición horizontal	Disp. horizontal	Izquierda	Centro
		Disp. vertical	Arriba	Abajo
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
Botón Azul	Botón	Color de fondo	Por defecto	Azul
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Rojo	Botón	Color de fondo	Por defecto	Rojo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Amarillo	Botón	Color de fondo	Por defecto	Amarillo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Verde	Botón	Color de fondo	Por defecto	Verde
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Nuevo Partido	Botón	Color de fondo	Por defecto	Gris
		Negrita	(ninguno)	✓
		Alto	Automático	8%
		Texto	Texto para botón	Nuevo partido
		Color de texto	Por defecto	Blanco

## Consigna 2: generar el código secreto

Una vez que todos hayan compuesto la interfaz gráfica, les comentamos: “Como hemos visto, un código secreto de una posición se puede representar con una variable global. En este caso, el código tiene cuatro posiciones. ¿Cómo harían para representarlo?”. Es esperable que algún estudiante conteste que se pueden usar cuatro variables globales, una para cada posición. “Sí, esa es una forma de resolverlo, pero tratemos de pensar una forma más general. Por ejemplo, ¿les parecería una buena solución usar mil variables globales si el código tuviese mil posiciones? ¡Ni siquiera podríamos verlas en la pantalla de la computadora!”. Guiamos el intercambio para concluir que, independientemente de la cantidad de posiciones del código secreto, siempre podríamos representarlo como *una única lista* de colores.

Continuamos: “Si fuese un código de cuatro posiciones, la lista tendría cuatro colores; si fuese de cinco, cinco colores; y si fuese de mil, entonces tendría mil colores, ¿qué les parece?”. Escuchamos con atención sus observaciones y les comentamos: “A pesar de que el código tiene varias posiciones, el desafío es seguir representándolo con una única variable global”. Los invitamos, entonces, a que exploren el entorno en busca de herramientas que les permitan resolver el desafío, que consiste en generar el código secreto al comenzar la ejecución de la aplicación, y conservarlo en una única variable global.

A continuación, un apartado sobre las listas y, más adelante, la continuación del desarrollo de la actividad.

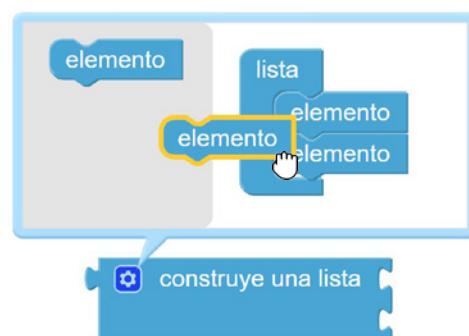
### Sobre las listas

En computación, una **lista** es un tipo de datos que representa una secuencia de elementos ordenados. Se puede pensar, por ejemplo, en listas de números, listas de colores o listas de palabras. Cada elemento, además, puede aparecer más de una vez en una lista.



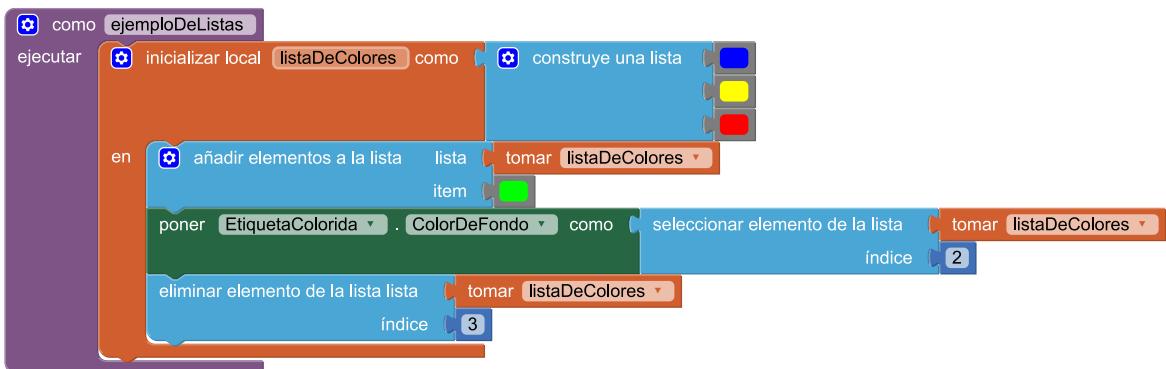
Listas de números, colores y palabras

Casi todos los lenguajes de programación modernos permiten trabajar con listas y tienen instrucciones para crearlas, agregarles y borrarles elementos, inspeccionar el valor de una cierta posición, etc. En App Inventor, una lista puede crearse a partir de una serie de elementos usando el bloque `construye una lista [ ] [ ]` disponible en *Bloques > Integrados > Listas*. La cantidad de valores que se incluyen en la lista al crearla de este modo se puede modificar accediendo a las opciones que se despliegan al hacer clic sobre la tuerca que se encuentra en la esquina superior izquierda del bloque.



Bloque para construir una lista a partir de una serie de elementos

En la imagen se encuentra un ejemplo que ilustra el uso de algunas operaciones que se pueden realizar con listas.



Ejemplo de uso de listas

En primer lugar se crea una variable `listaDeColores` cuyo valor inicial es una lista en la que en la primera posición está el color azul, en la segunda el color amarillo y en la tercera el color rojo. Luego, se utiliza el bloque `añadir elementos a la lista (lista) [ ] (ítem) [ ]` para agregar al final de `listaDeColores` el color verde (esta instrucción agrega un elemento al final de la lista). Aquí, el parámetro `lista` representa la lista en la que se agrega un nuevo elemento e `ítem` el elemento que se agrega. A continuación, se usa `seleccionar elemento de la lista [ ] (índice) [ ]` para establecer que el color de fondo de la etiqueta `EtiquetaColorida` sea el que se encuentra en la segunda posición de `listaDeColores`; es decir, de color amarillo. Finalmente, con el bloque `eliminar elemento de la lista (lista) [ ] (índice) [ ]` se remueve el elemento que está en la tercera posición; en este caso, el color rojo. En la tabla a continuación se observa la evolución de la lista `listaDeColores` a medida que avanza la ejecución del procedimiento `ejemploDeListas`.

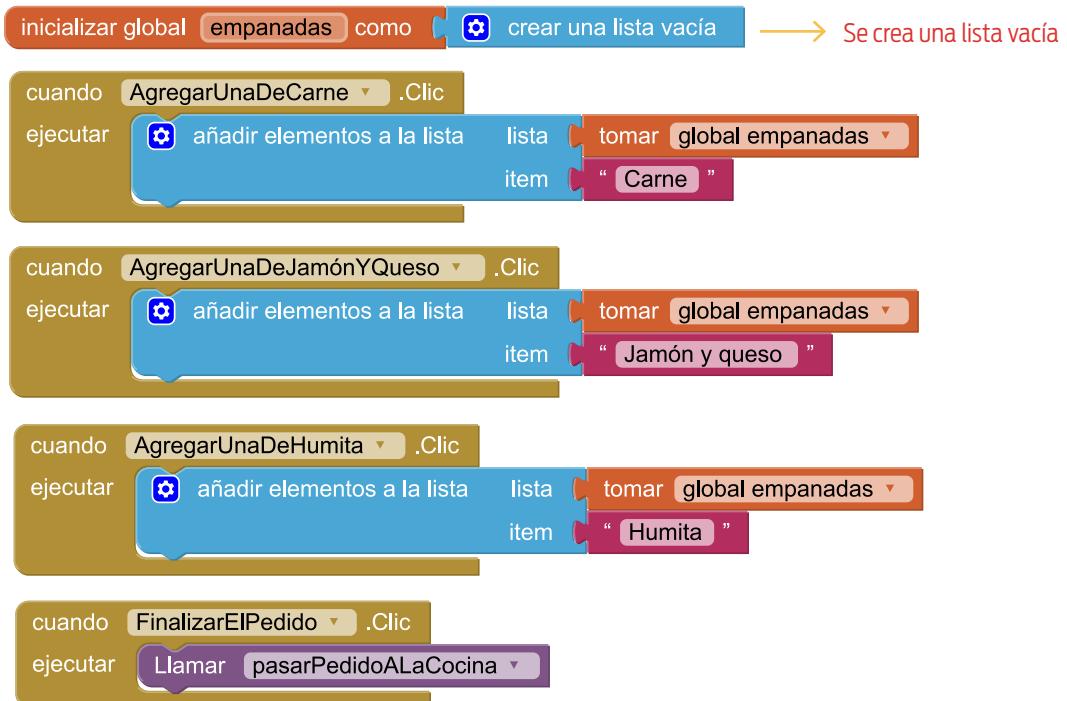
INSTRUCCIÓN	LISTA REPRESENTADA
<code>inicializar local [listaDeColores v] como</code> <code>construye una lista [azul, amarillo, rojo]</code>	
<code>añadir elementos a la lista [lista] [tomar [listaDeColores v] de [listaDeColores v] como item]</code>	

Pone amarillo el fondo de una etiqueta sin modificar la lista

INSTRUCCIÓN	LISTA REPRESENTADA
<pre>poner EtiquetaColorida . ColorDeFondo como seleccionar elemento de la lista tomar listaDeColores           indice 2</pre>	
<pre>eliminar elemento de la lista lista tomar listaDeColores           índice 3</pre>	

Evolución de `listaDeColores` a medida que se ejecuta `ejemploDeListas`

En App Inventor –al igual que en todos los lenguajes que admiten el uso de listas– es posible trabajar con **listas vacías**, es decir, con listas que no contienen ningún elemento. Estas son útiles en distintos contextos. A modo de ejemplo, puede considerarse una aplicación de *delivery* de empanadas. Las empanadas que forman parte de un pedido pueden representarse como una lista de los gustos que un cliente seleccione. Al comenzar un pedido, aún no se ha agregado ninguna empanada; por lo tanto, la lista se encuentra vacía. Luego, a medida que se agreguen empanadas, sus gustos se irán agregando a la lista. Una vez completado, el pedido pasa a la cocina para comenzar la preparación de la comida.

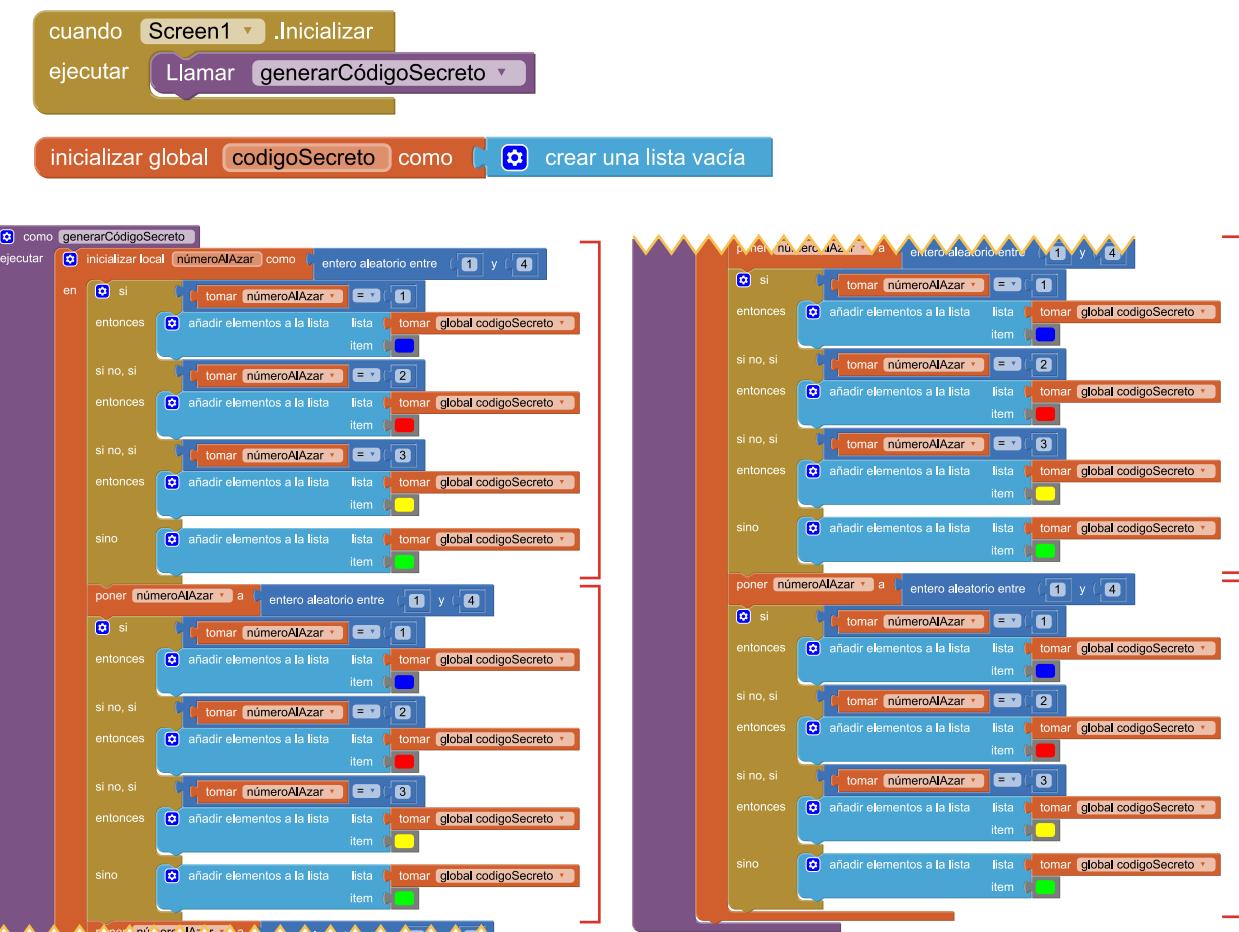


Uso de una lista vacía

## Consigna 2. Generar el código secreto (continuación)

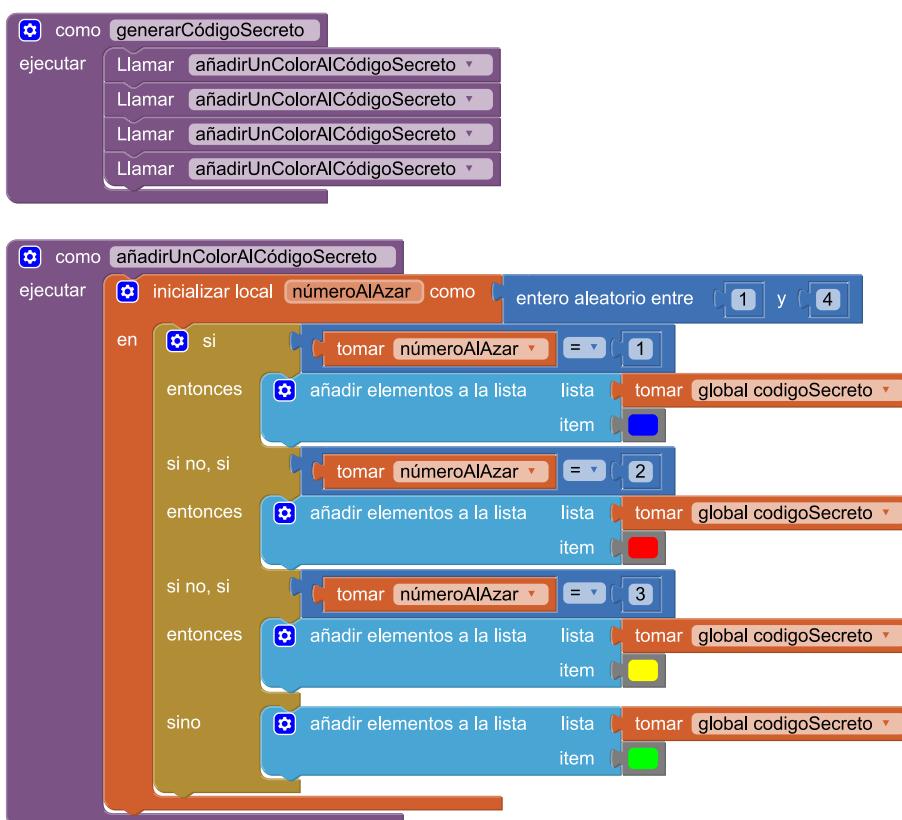
Como en distintas partes del programa hay que acceder a la variable en la que se almacenará el código secreto (por ejemplo, cuando la aplicación comienza su ejecución y se genera el código y, luego, cuando el jugador completa su código candidato y se cuentan las coincidencias), hay que usar una variable global.

Una solución que podría presentarse consiste en inicializar una variable `códigoSecreto` como una lista vacía y, manejando el evento que se produce cuando se carga la aplicación, invocar a un procedimiento –que podría llamarse `generarCódigoSecreto`– que genere cuatro números al azar, uno a continuación del otro, para definir los colores del código secreto.



Se realiza cuatro veces lo mismo

A los estudiantes que acerquen esta propuesta les hacemos notar que en `generarcódigoSecreto` están haciendo cuatro veces lo mismo y les sugerimos que piensen alguna solución alternativa. Una variación que mejora cualitativamente el programa consiste en encapsular el añadido de un color al código secreto en un procedimiento –que podría llamarse, por ejemplo, `añadirUnColorAlCódigoSecreto`– que sea invocado cuatro veces.



Se encapsula en un procedimiento la generación de un color del código

Frente a esta propuesta comentamos: “Está muy bien encapsular en un procedimiento la generación de un color del código pero, ¿no se sigue repitiendo cuatro veces lo mismo en `generarcódigoSecreto`? ¿Qué sucedería si el código secreto fuese de mil posiciones?”.

Como en casi cualquier lenguaje de programación, en App Inventor hay herramientas que permiten repetir instrucciones. En particular, el bloque `por cada (número) desde [ ] hasta [ ] en incrementos de [ ] / ejecuta { }` sirve para expresar en forma explícita la cantidad de veces que se tienen que ejecutar las instrucciones que se encuentren dentro de `{ }`.

`por cada [número] desde [ ] hasta [ ] en incrementos de [ ] ejecuta [ ]`

Bloque para expresar repeticiones

El modo más habitual de usar el bloque es completando `desde` con un 1, `hasta` con la cantidad de veces que queremos que se repita la ejecución de las instrucciones contenidas en `{ }` y `en incremento de` con un 1. En la figura se muestra una forma de completar `generarCódigoSecreto` para resolver el desafío.<sup>1</sup>

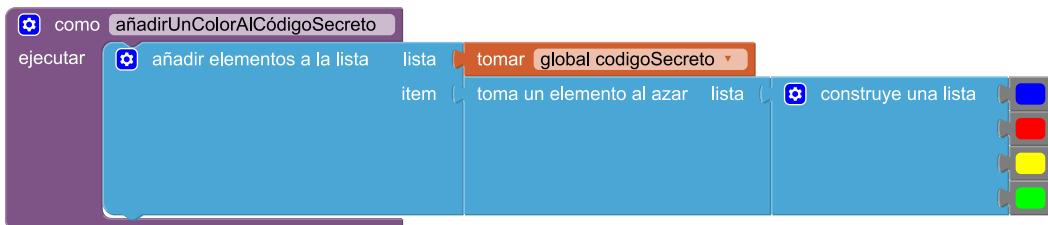
Una forma alternativa (y elegante) de elegir un color al azar para el código secreto sin necesidad de generar un número aleatorio puede realizarse usando el bloque `toma un elemento al azar [lista]`, disponible en *Bloques > Integrado > Listas*. De este modo, se puede agregar a la lista `códigoSecreto` el resultado de seleccionar un elemento al azar de otra lista (creada *ad hoc*) que contenga los cuatro colores –azul, rojo, amarillo y verde–.



Procedimiento que usa repeticiones

### toma un elemento al azar [lista]

Bloque para obtener un elemento al azar de una lista



Forma alternativa de `añadirUnColorAlCódigoSecreto`

### Consigna 3: manejar eventos de los botones de colores

La tercera consigna pide manejar los eventos que se producen cuando se presionan los botones de colores. Hacerlo involucra (*i*) actualizar en la pantalla el código candidato agregándole el color elegido por el jugador, y (*ii*) chequear si al agregar el nuevo color se completa el código candidato, caso en el cual hay que develar el código secreto, mostrar la cantidad de coincidencias y deshabilitar los botones de colores.

En primer lugar, los estudiantes tienen que reconocer que, al igual que con el código secreto, para el código candidato también es necesario usar una variable global. Esta, que podría llamarse `códigoCandidato`, se inicializa como una lista vacía al comenzar la ejecución de la aplicación y se irá actualizando a medida que se elijan los colores.

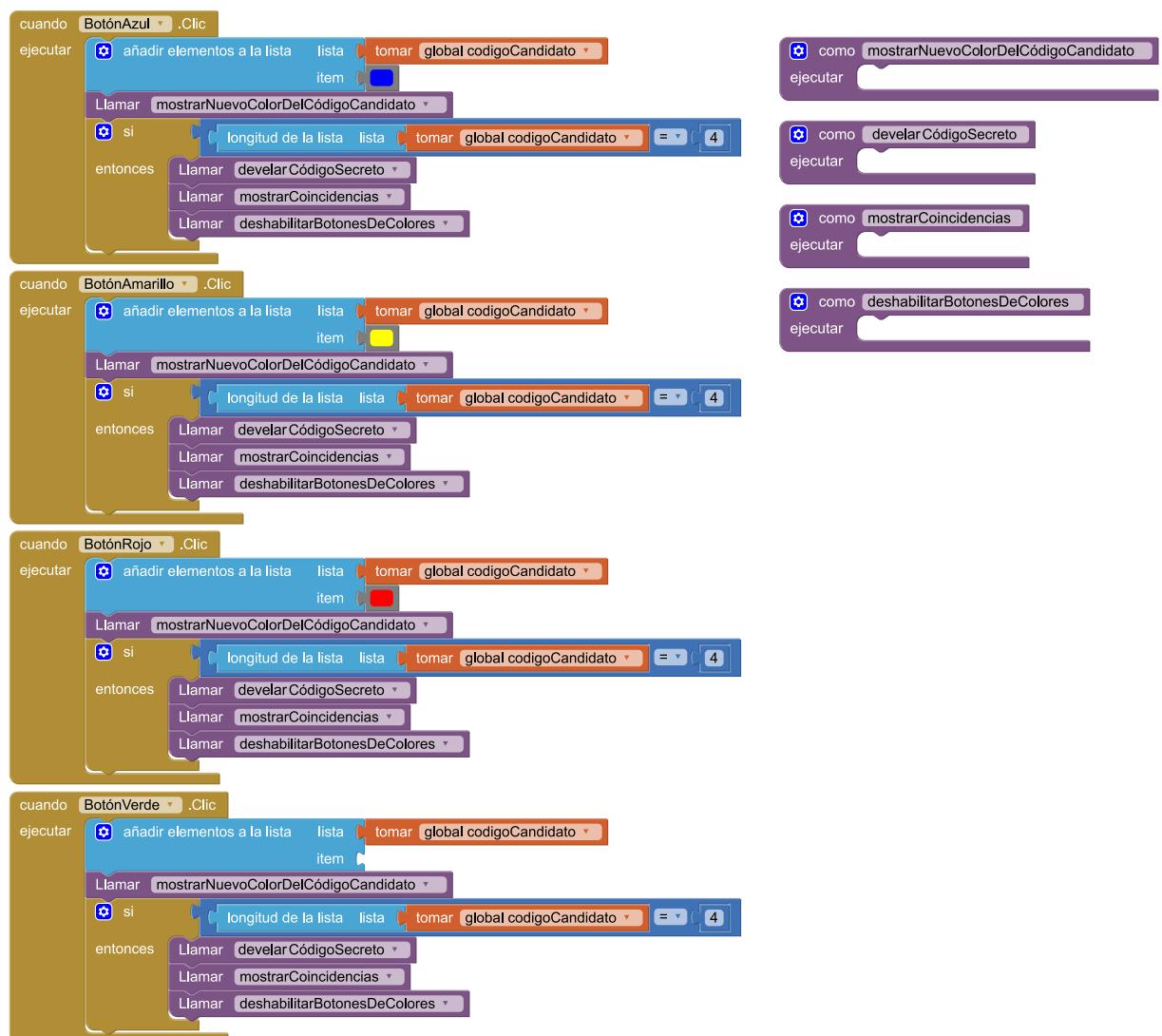
<sup>1</sup>Hay otras alternativas para describir un número  $n$  de repeticiones, aunque son más rebuscadas y menos recomendables. Por ejemplo, en lugar de completar (`desde, hasta, en incremento de`) con (1,4,1), se podría usar (7,10,1) o (1,7,2), entre otros, y generar el mismo efecto: 4 repeticiones.

```
inicializar global codigoSecreto como crear una lista vacía
```

Variable que guarda el código candidato

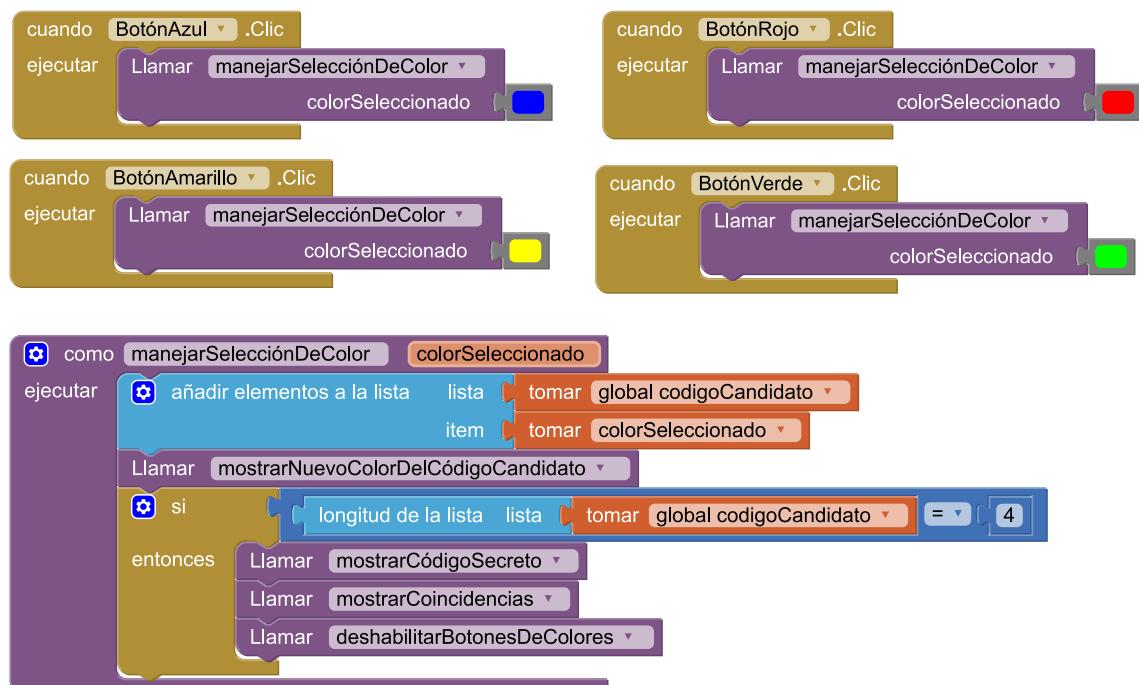
Para saber si se ha completado el código candidato, hay que chequear si la longitud de **codigoCandidato** es 4 (esto solo sucede cuando ya se le agregaron cuatro colores).

Además, para completar el manejo de los eventos que se generan al presionar los botones de colores, pueden crearse cuatro procedimientos –que luego nos encargaremos de resolver– donde cada uno encapsule una tarea específica. Podrían llamarse, por ejemplo, **mostrarNuevoColorDelCódigoCandidato**, **develarCódigoSecreto**, **mostrarCoincidencias** y **deshabilitarBotonesDeColores**.



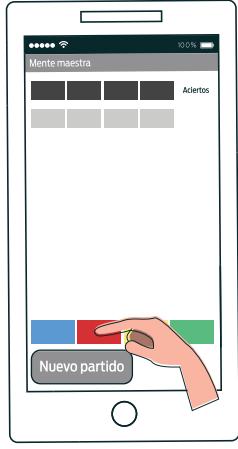
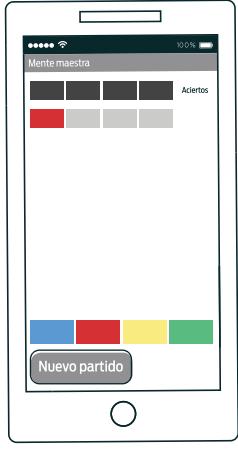
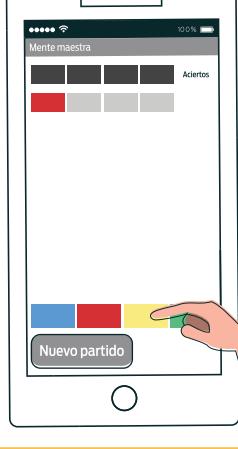
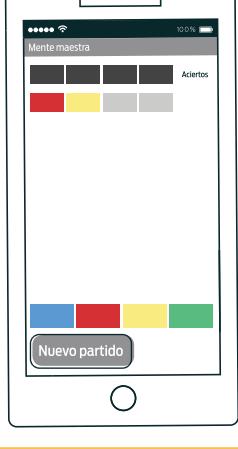
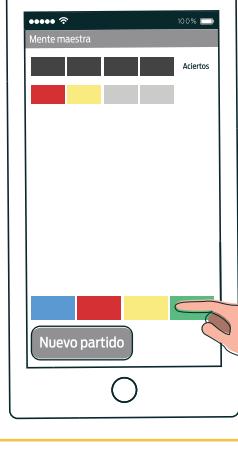
Programa incompleto y con redundancia

Como puede observarse, los manejos de los eventos de los cuatro botones son muy similares: únicamente difieren en la primera instrucción, en la que varía el color que se agrega a la lista `códigoCandidato`. Por lo tanto, se puede crear un único procedimiento que abstraiga el color en un parámetro y que sea invocado con el color adecuado en cada caso. Un nombre posible sería `manejarSelecciónDeColor (colorSeleccionado)`.



Refactorización del manejo de eventos

El objetivo de `mostrarNuevoColorDelCódigoCandidato` es reflejar visualmente la selección de un nuevo color por parte del usuario. En la pantalla, el código candidato está representado por cuatro etiquetas: `CódigoCandidato_1`, `CódigoCandidato_2`, `CódigoCandidato_3` y `CódigoCandidato_4`. ¿En cuál de ellas hay que mostrar el color elegido? Esta pregunta puede responderse a partir de la longitud de la lista `códigoCandidato`: si tiene un solo elemento, hasta el momento el usuario solo ha incorporado un color al código candidato, con lo que hay que mostrarlo en `CódigoCandidato_1`; siguiendo el mismo razonamiento, si tiene dos hay que mostrarlo en `CódigoCandidato_2`; si tiene 3, en `CódigoCandidato_3`; y si tiene 4 en `CódigoCandidato_4`. Además, el color que hay que mostrar es, siempre, el último agregado a `códigoCandidato`, pues allí se encuentra la última elección del usuario.

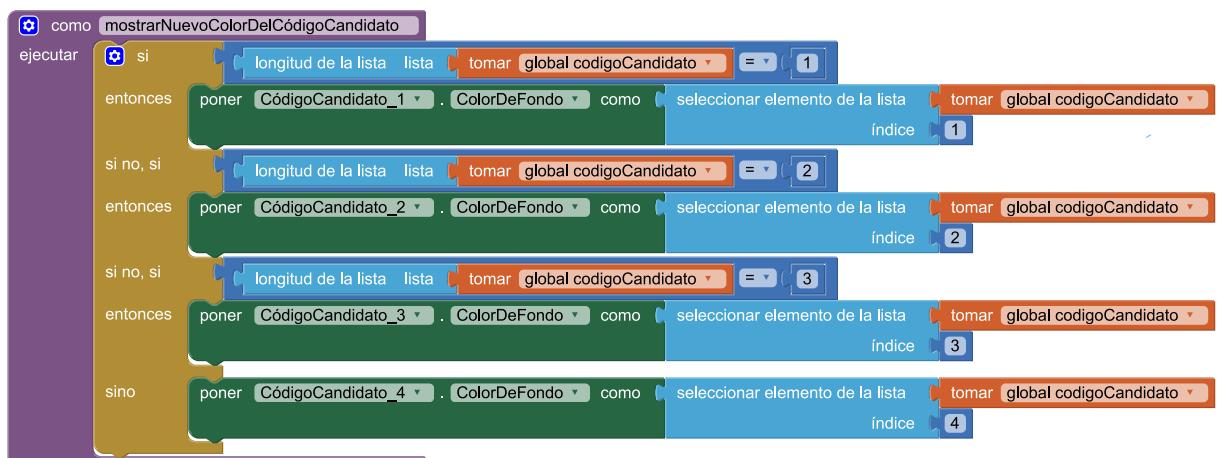
ACCIÓN DEL USUARIO	EFFECTO EN LA LISTA CÓDIGO CANDIDATO	REFLEJO EN LA PANTALLA
		
		
		



Evolución de una corrida de la aplicación

El último elemento de una lista puede obtenerse a partir de su longitud. Por ejemplo, en una lista con dos elementos –es decir, de longitud 2–, el último elemento es el que se encuentra en la segunda posición; y en una con tres elementos –de longitud 3–, el que está en la tercera posición. En general, en una lista con  $n$  elementos –de longitud  $n$ –, el último se encuentra en la enésima posición.<sup>1</sup>

Se muestra a continuación el procedimiento `mostrarNuevoColorDelCódigoCandidato`.



Se muestra un nuevo color del código candidato

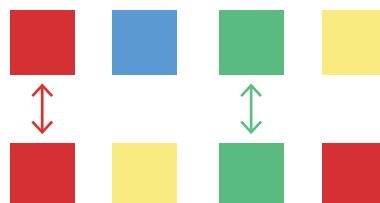
<sup>1</sup> Esta afirmación no es válida en una lista vacía. En tal caso, no hay elemento alguno.

El propósito del procedimiento `desvelarcódigoSecreto` es mostrar en la pantalla el código secreto. Alcanza, en este caso, con establecer el color de fondo de las etiquetas `CódigoSecreto_1`, `CódigoSecreto_2`, `CódigoSecreto_3` y `CódigoSecreto_4`, con los colores de la primera, la segunda, la tercera y la cuarta posición (respectivamente) de la lista `códigoSecreto`.



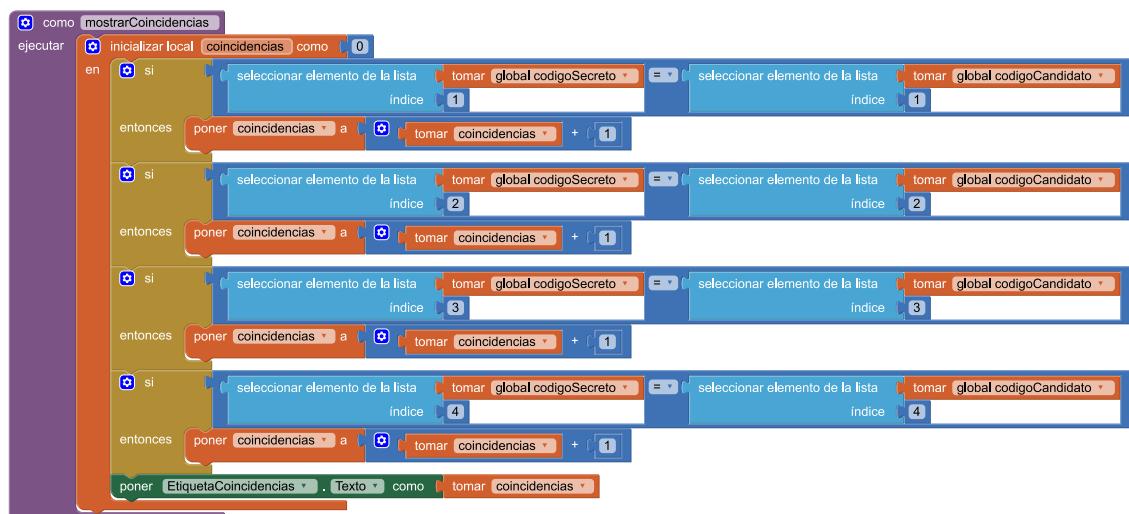
Se devela el código secreto

La cantidad de coincidencias entre el código secreto y el código candidato está dada por la cantidad de posiciones de las listas `códigoSecreto` y `códigoCandidato` que contienen el mismo color. Por ejemplo, si el código secreto es [rojo,azul,verde,amarillo] y el código candidato [rojo,amarillo,verde,rojo] hay dos coincidencias, que corresponden a las posiciones 1 y 3 de las listas.



Dos coincidencias

Para calcular las coincidencias se puede inicializar una variable en 0 y, por cada coincidencia, sumarle 1. Luego, para mostrar el resultado por pantalla, alcanza con actualizar el texto de la etiqueta `EtiquetaCoincidencias`. Una posible propuesta se muestra a continuación.



Propuesta endeble para calcular coincidencias

Si bien es funcionalmente correcta, esta solución no es buena. Por ejemplo, si los códigos pasasen a ser muy largos, habría que incluir una gran cantidad de sentencias condicionales, una por cada posición del código. El resultado sería un programa muy largo, menos comprensible y más propenso a errores.

Una forma de saldar el problema descrito es usar repeticiones. De este modo, se puede analizar una posición diferente de las listas en cada ejecución del cuerpo de la repetición.



Es interesante notar que, en este caso, se ha usado el valor que adquiere **posición** en cada ejecución del cuerpo de la repetición para indicar qué posición de las listas hay que inspeccionar: la primera vez vale uno, y evalúa si son iguales el primer color de **codigosecreto** y el primero de **codigocandidato**; la segunda vale dos, y se compara el segundo de uno contra el segundo del otro, y así siguiendo. De algún modo, **posición** es una variable y puede referenciarse únicamente en los bloques que componen la repetición.

Para completar la consigna, solo resta completar el procedimiento **deshabilitarBotonesDeColores**. En este caso alcanza con establecer la propiedad **Botón.Habilitado** de los cuatro botones de colores como **falso**.



Se deshabilitan los botones de colores

#### Consigna 4: todo como al principio

La cuarta y última consigna pide que al presionar el botón *Nuevo partido* se restablezca el estado inicial de la aplicación de forma tal que el usuario pueda intentar adivinar un nuevo código secreto. Esto requiere (i) vaciar las listas **codigosecreto** y **codigocandidato**, (ii) generar un nuevo código secreto, (iii) habilitar los botones de colores, (iv) agrisar las etiquetas que muestran los códigos en la pantalla, y (v) vaciar el texto de la etiqueta que muestra la cantidad de coincidencias.



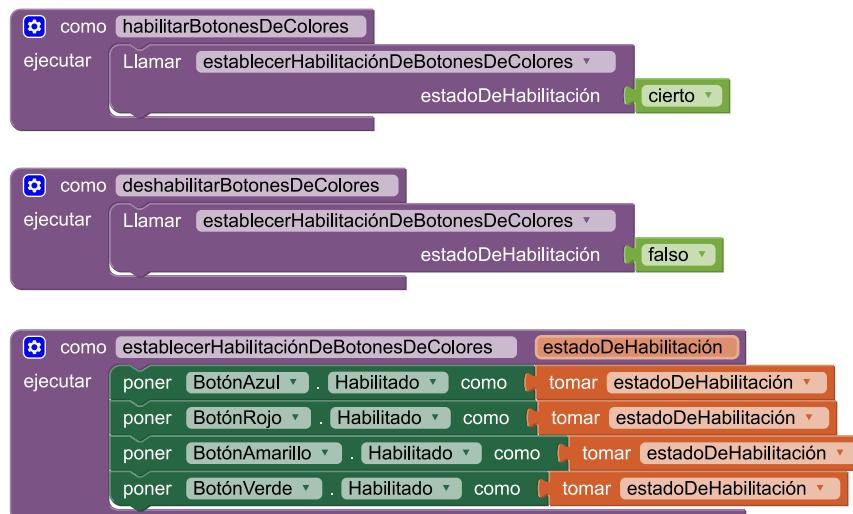
Comenzar un nuevo partido

Al igual que cuando son inicializadas al cargar la aplicación, a las variables `códigoSecreto` y `códigoCandidato` se les puede asignar una lista vacía. Por otro lado, el procedimiento `generarCódigoSecreto` ya está programado. Para habilitar los botones de colores, se puede crear un procedimiento que establezca la propiedad `Botón.Habilitado` de los cuatro botones de colores como `cierto`.



Procedimientos similares para habilitar y deshabilitar botones

Debido a la similitud entre `habilitarBotonesDeColores` y `deshabilitarBotonesDeColores`, se puede crear un nuevo procedimiento que tenga un parámetro que indique si se quiere habilitar o deshabilitar a los botones y, en cada caso, invocarlo con el argumento que corresponde.



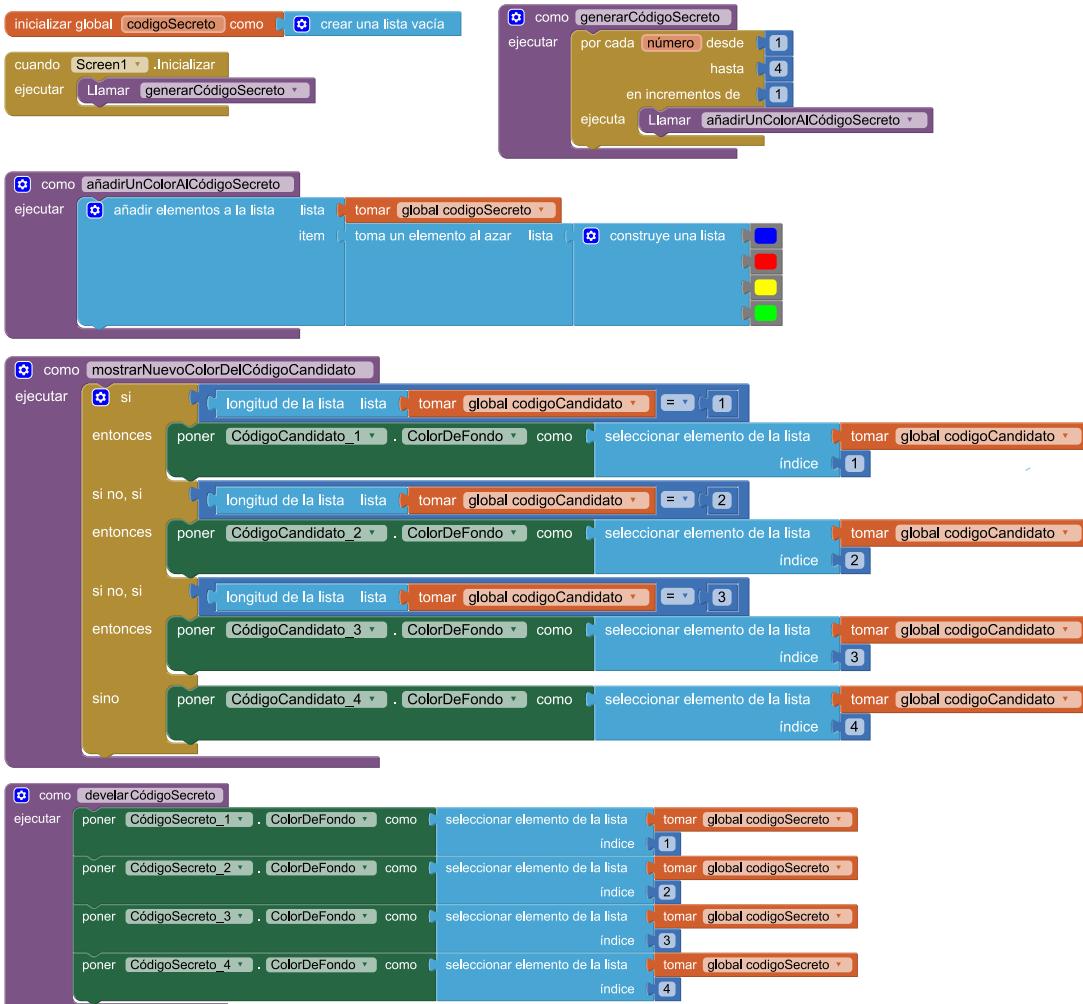
Refactorización para habilitar y deshabilitar botones

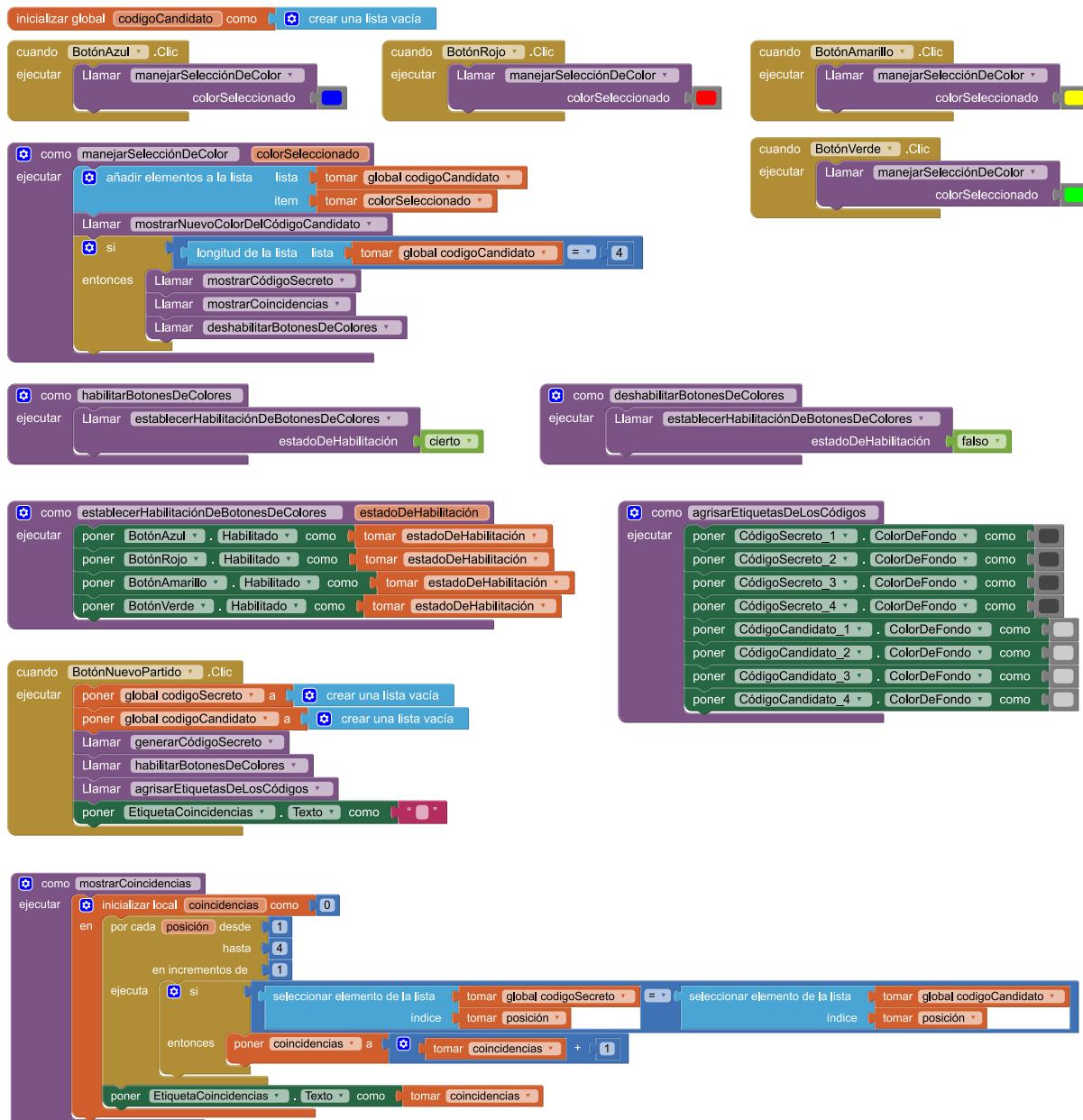
Por último, el procedimiento `agrisarEtiquetasDeLosCódigos` solo debe ocuparse de establecer el color de fondo de las etiquetas que muestran el código secreto y el código candidato como gris oscuro y gris claro respectivamente.



Se agrisan las etiquetas de los códigos

A continuación se exhibe una solución completa de la actividad.





Programa completo

## CIERRE

Rapasamos con los estudiantes que las listas dan la posibilidad de representar secuencias de valores ordenados sin necesidad de usar muchas variables. Además, hacemos hincapié en la utilidad de contar con bloques para hacer repeticiones, que permiten hacer muchas veces lo mismo sin tener que repetir instrucciones de forma explícita.

NOMBRE Y APELLIDO:

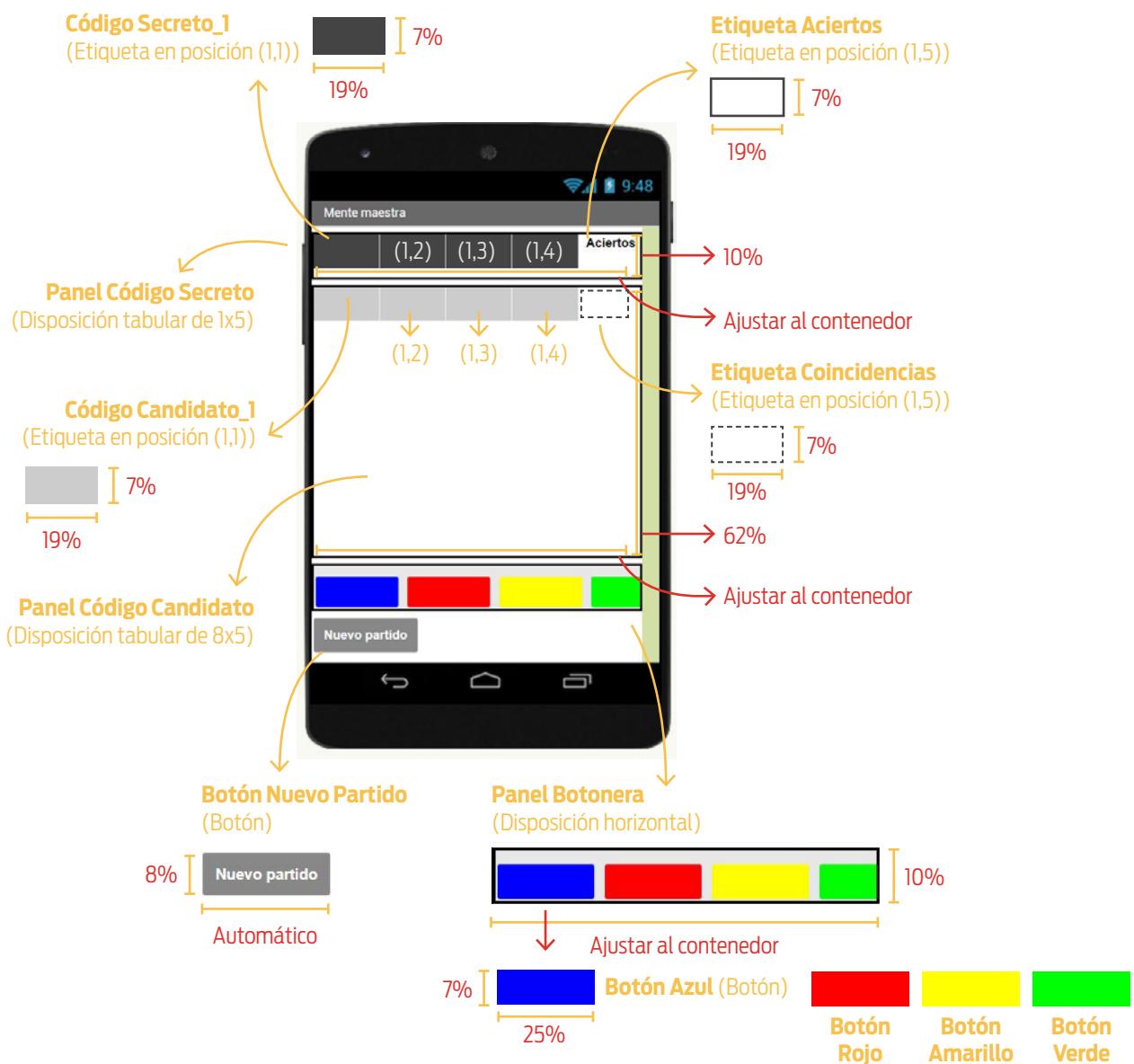
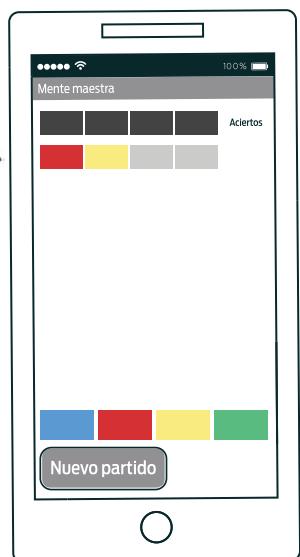
CURSO:

FECHA:

# MENTE MAESTRA, PARTE II

Ahora la cosa se pone más complicada. Los jugadores tienen un intento para adivinar un código de cuatro colores. ¡Solo una chance entre 256!

1. Comenzá armando la interfaz gráfica. Acá abajo podés ver los componentes y sus dimensiones. En la tabla, las propiedades que tenés que cambiar para que se vea igual a la que te mostramos. Seguí las indicaciones para componerla.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen1	Mente maestra
Panel Código Secreto	Disposición tabular	Columnas	2	5
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	1
Código Secreto_1 <sup>1</sup>	Etiqueta	Color de fondo	Ninguno	Gris oscuro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Aciertos	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	Aciertos
		Posición del texto	Izquierda	Centro
Panel Código Candidato	Disposición tabular	Columnas	2	5
		Alto	Automático	62%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	8
Código Candidato_1 <sup>2</sup>	Etiqueta	Color de fondo	Ninguno	Gris claro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Coincidencias	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
		Posición del texto	Izquierda	Centro

<sup>1</sup>Las restantes etiquetas del código secreto requieren los mismos cambios.

<sup>2</sup>Las restantes etiquetas del código candidato requieren los mismos cambios.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Panel Botonera	Disposición horizontal	Disp. horizontal	Izquierda	Centro
		Disp. vertical	Arriba	Abajo
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
Botón Azul	Botón	Color de fondo	Por defecto	Azul
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Rojo	Botón	Color de fondo	Por defecto	Rojo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Amarillo	Botón	Color de fondo	Por defecto	Amarillo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Verde	Botón	Color de fondo	Por defecto	Verde
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Nuevo Partido	Botón	Color de fondo	Por defecto	Gris
		Negrita	(ninguno)	✓
		Alto	Automático	8%
		Texto	Texto para botón	Nuevo partido
		Color de texto	Por defecto	Blanco

NOMBRE Y APELLIDO:

CURSO:

FECHA:

- 2.** Ni bien comienza su ejecución, la aplicación tiene que generar al azar un código secreto de 4 posiciones. Además, a pesar de tener cuatro posiciones con colores, tenés que conservar el código en una única variable.

### LISTAS

En computación, una **lista** es un tipo de datos que representa una secuencia de elementos ordenados. Se puede pensar, por ejemplo, en listas de números, listas de colores o listas de palabras, etc. Cada elemento, además, puede aparecer más de una vez en una lista.

1 7 21 3



Lista de cuatro números



Lista de cuatro colores

Hola matungo



Lista de dos palabras



¿Usaste listas? ¿Por qué?

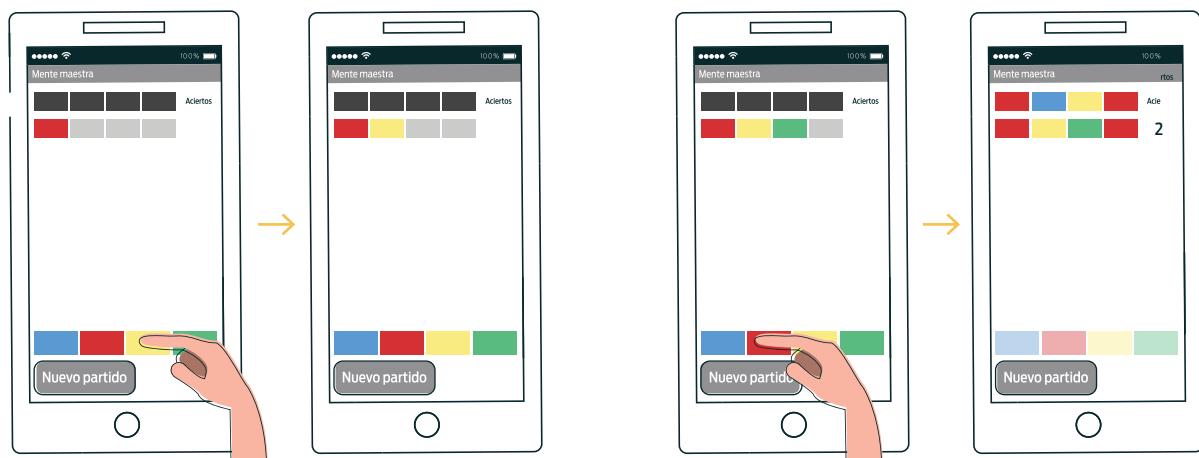
---

---

---

- 3.** Ahora tenés que programar el manejo de los eventos que se producen cuando se presionan los botones de colores. Hacerlo involucra (i) actualizar en la pantalla el código candidato agregándole el color elegido por el jugador, y (ii) chequear si al agregar el nuevo color se completa el código candidato, caso en el cual hay que develar el código secreto, mostrar la cantidad de coincidencias y deshabilitar los botones de colores.

Acá abajo podés ver dos ejemplos: uno en el que al agregar un color no se completa el código candidato y otro en el que sí.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

¿Cómo hiciste para que se calcule la cantidad de aciertos una vez que el jugador ingrese los cuatro colores del código candidato? ¿Usaste algún bloque que permita repetir secuencias de instrucciones? ¿Cuál?

---

---

---

---

---

- 4.** Por último, el botón *Nuevo partido* tiene que dejar todo listo para intentar adivinar el código secreto otra vez. ¿Reusaste alguna parte del programa para conseguir el objetivo? ¿Cuál? ¿Para qué?

---

---

---

---

---

## Actividad 4

### Mente maestra, parte III



#### OBJETIVOS

- Construir una aplicación partiendo de una versión previa.
- Presentar los operadores lógicos de disyunción y conjunción.
- Introducir las funciones.

#### MATERIALES

- Computadora
- Internet
- MIT App Inventor 2
- Ficha para estudiantes
- Teléfono con Android (opcional)

## DESARROLLO

Es habitual que las aplicaciones evolucionen y, a medida que pasa el tiempo, aparezcan nuevas versiones que cambian o agregan funcionalidades respecto a sus predecesoras. En estos casos, generalmente, el equipo que desarrolla el *software* no construye el programa desde cero; el punto de partida suele ser una versión previa de la aplicación.

En esta actividad, la propuesta es modificar el programa de la actividad “Mente maestra, parte II” para llegar a la versión final del juego, en la que los jugadores cuentan con 8 intentos para descifrar el código secreto.<sup>1</sup>

En el desarrollo de la actividad tomaremos como punto de partida la solución de “Mente maestra, parte II” presentada en el manual. En caso de querer que los estudiantes utilicen la desarrollada por ellos, la ficha debe ser adaptada.

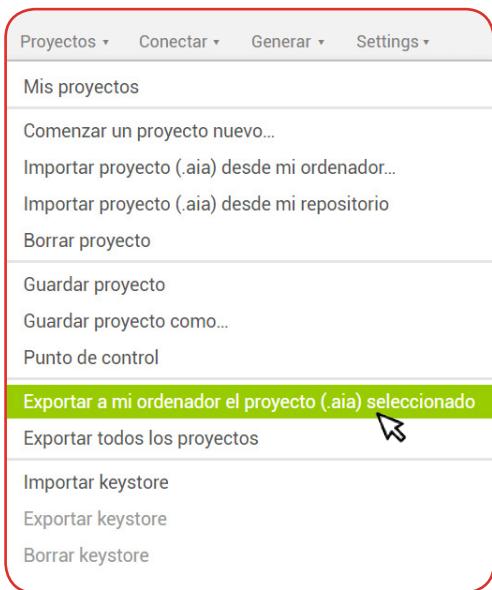
#### Preparación de la actividad

Antes de dar comienzo a la actividad debemos distribuir entre los estudiantes la solución de “Mente maestra, parte II” presentada en el manual. En primer lugar, para obtenerla, debemos seguir los siguientes pasos: (i) ingresar con nuestro usuario al entorno de App Inventor, (ii) acceder a la galería de aplicaciones y buscar “programar2020”, (iii) seleccionar la apertura del proyecto “Mente\_maestra\_parte\_II” y cambiarle el nombre por “Mente\_maestra\_parte\_III”, y (iv) exportar el proyecto para descargarlo. De este modo, obtendremos el archivo *Mente\_maestra\_parte\_III.aia*, que podremos compartir con los estudiantes haciendo circular un *pendrive*, mandándolo en un correo electrónico, etc.



Se obtiene y renombra el proyecto *Mente\_maestra\_parte\_II*

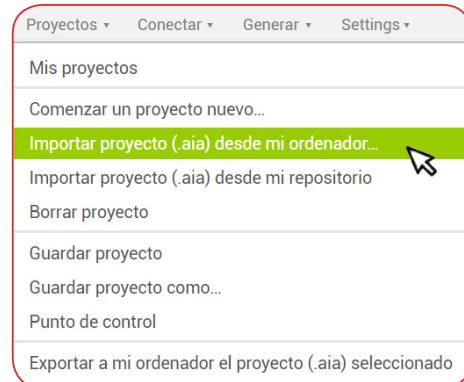
<sup>1</sup> Una versión completa se encuentra disponible en la galería de proyectos de App Inventor del usuario “programar2020”.



Se exporta el proyecto

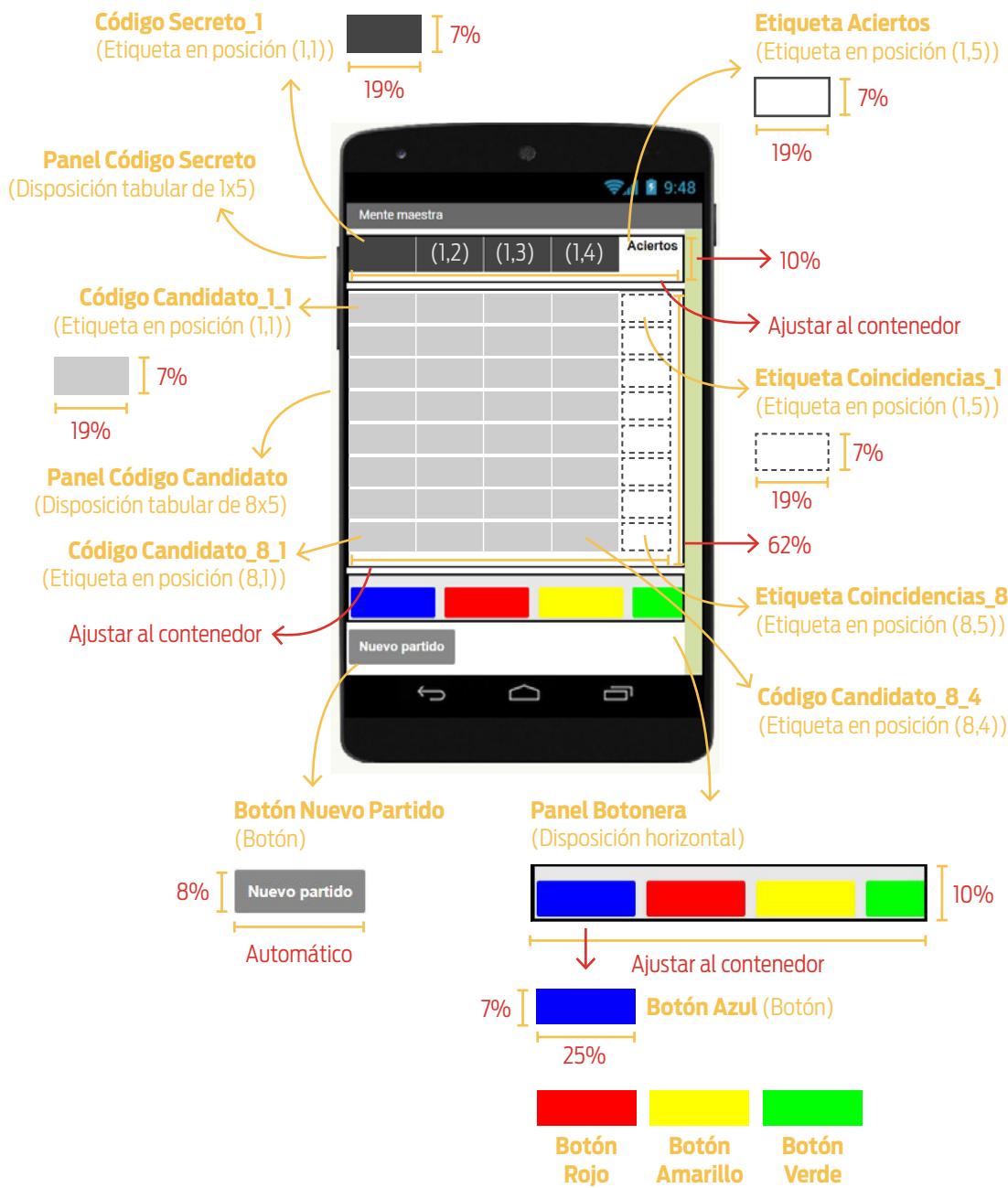
### Consigna 1: interfaz gráfica

Comenzamos la actividad distribuyendo entre los estudiantes el archivo *Mente\_maestra\_parte\_III.aia*. Luego, les indicamos que ingresen al entorno de App Inventor e importen el proyecto. Les comentamos: "Para alcanzar la versión completa del Mente maestra, este será nuestro punto de partida: una implementación de la actividad Mente maestra, parte II".



Se importa el proyecto

A continuación, les entregamos la ficha y los invitamos a que resuelvan la primera consigna. Allí se dan instrucciones para completar la interfaz gráfica de la aplicación. La diferencia entre la interfaz de esta versión respecto de la anterior es que, en el panel de los códigos candidatos, se incorporan 7 filas de 5 etiquetas para que, sumadas a la que ya estaba, se representen 8 códigos candidatos en lugar de 1. En cada fila, las primeras cuatro etiquetas irán mostrando los colores que el usuario vaya eligiendo, y la quinta, la cantidad de coincidencias cada vez que el jugador complete un intento. Les remarcamos que en este caso es conveniente renombrar los componentes de acuerdo a lo indicado en la ficha. Por ejemplo, la etiqueta *CódigoCandidato\_1* de la versión II (que correspondía al primer color del único código candidato) pasa a llamarse *CódigoCandidato\_1\_1*, pues ahora se trata de la primera posición del primer código candidato. Con el primer número indicaremos el número de intento y, con el segundo, la posición dentro de un código candidato. Por ejemplo, *CódigoCandidato\_3\_4* hará referencia a la cuarta posición del tercer código candidato.



Diseño de la interfaz

La tabla a continuación muestra los valores de las propiedades que hay que cambiar (en relación con los que App Inventor asigna por defecto) para que la aplicación se vea tal como en la imagen.

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen1	Mente maestra
Panel Código Secreto	Disposición tabular	Columnas	2	5
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	1
Código Secreto_1 <sup>1</sup>	Etiqueta	Color	Ninguno	Gris oscuro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Aciertos	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	Aciertos
		Posición del texto	Izquierda	Centro
Panel Código Candidato	Disposición tabular	Columnas	2	5
		Alto	Automático	62%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	8
Código Candidato_1_1 <sup>2</sup>	Etiqueta	Color de fondo	Ninguno	Gris claro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Coincidencias_1 <sup>3</sup>	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
		Posición del texto	Izquierda	Centro

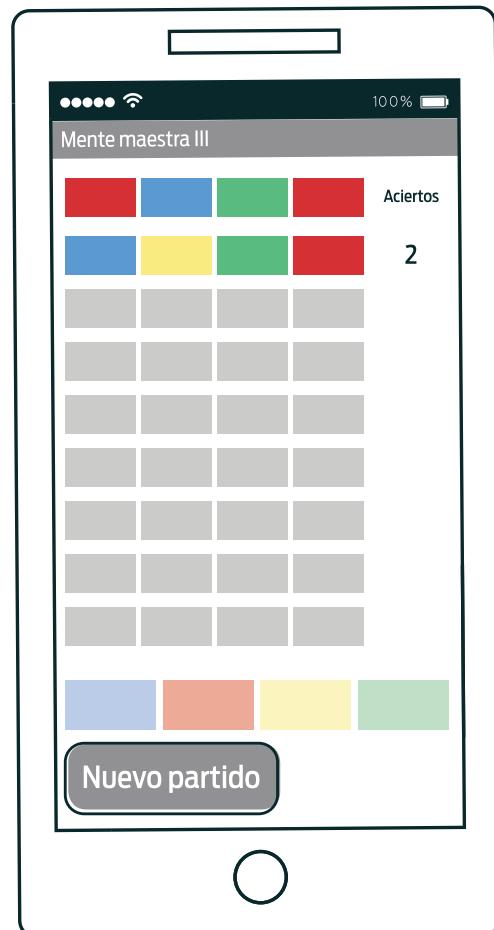
<sup>1</sup>Las restantes etiquetas del código secreto requieren los mismos cambios.<sup>2</sup>Las restantes etiquetas del código candidato requieren los mismos cambios.<sup>3</sup>Las restantes etiquetas para mostrar la cantidad de aciertos de un intento requieren los mismos cambios.

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Panel Botonera	Disposición horizontal	Disp. horizontal	Izquierda	Centro
		Disp. vertical	Arriba	Abajo
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
Botón Azul	Botón	Color de fondo	Por defecto	Azul
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Rojo	Botón	Color de fondo	Por defecto	Rojo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Amarillo	Botón	Color de fondo	Por defecto	Amarillo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Verde	Botón	Color de fondo	Por defecto	Verde
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Nuevo Partido	Botón	Color de fondo	Por defecto	Gris
		Negrita	(ninguno)	✓
		Alto	Automático	8%
		Texto	Texto para botón	Nuevo partido
		Color de texto	Por defecto	Blanco

Valores diferentes a los asignados por defecto

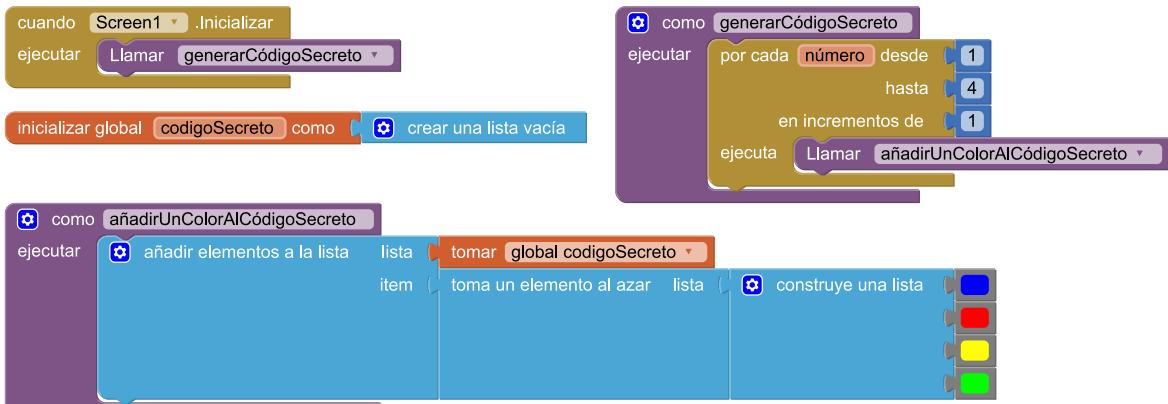
### Consigna 2: ejecución de la versión inicial

Una vez que los estudiantes hayan compuesto la interfaz de la aplicación, les indicamos que la ejecuten y jueguen una partida. Les damos un pequeño tiempo para que la prueben y les preguntamos: “¿Cómo les fue con el juego? ¿Con qué se encontraron?”. Si bien al ejecutarla aparecen en la pantalla las etiquetas que representan los 8 códigos candidatos, el comportamiento de la aplicación es el mismo que el de la versión anterior: una vez ingresados cuatro colores, el juego finaliza. Continuamos: “Más allá de que al ver la pantalla nos da la impresión de que vamos a poder hacer 8 intentos, lo cierto es que luego del primero se deshabilitan los botones de colores y, entonces, lo único que podemos hacer es comenzar un nuevo partido. ¿Por qué está pasando esto?”. Escuchamos las respuestas de los estudiantes y guiamos la discusión para concluir que solo se ha cambiado el aspecto, pero no los bloques que definen el comportamiento del programa. “En definitiva, las instrucciones de un programa son las que determinan su funcionamiento, no los componentes de la interfaz gráfica”.



El juego finaliza luego del primer intento

Continuamos: “Si bien en esta versión de la aplicación vamos a permitir ocho intentos para descifrar el código secreto (en lugar de uno), ¿no hay algún requerimiento que coincida con los de la versión previa?”. Escuchamos sus comentarios y preguntamos: “¿Qué tiene que pasar, en ambas versiones, cuando comienza su ejecución?”. Se debe generar el código secreto. “¡Claro! Y eso, ¿hace falta programarlo de nuevo? ¡Por supuesto que no!”.

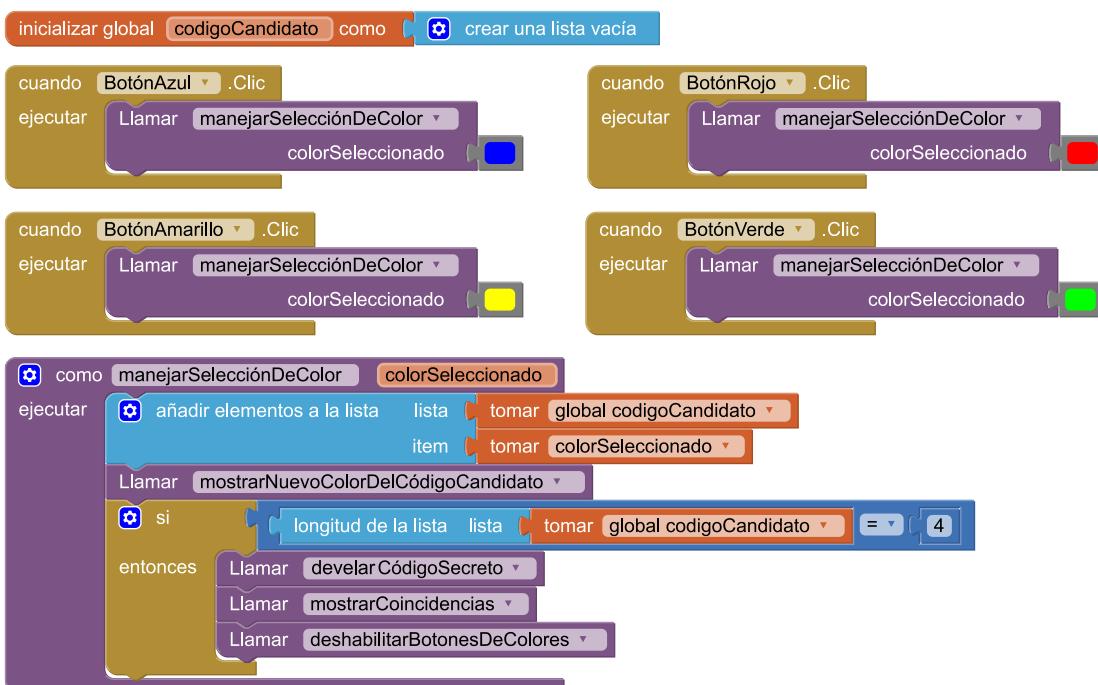


Fragmento del programa que genera el código secreto

Los invitamos a que inspeccionen por dentro el programa –desde el editor de bloques– y a que observen cómo está programado.

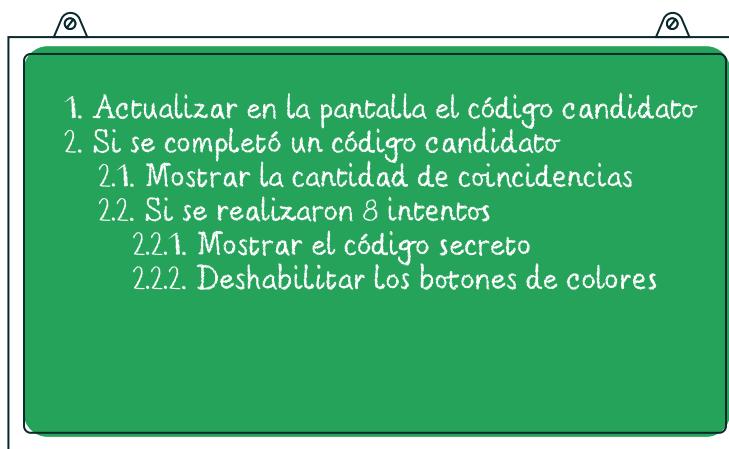
### Consigna 3: los ocho intentos

Luego de hacer una puesta en común, les proponemos a los estudiantes que observen en el editor de bloques cómo está programado el manejo de eventos de los botones de colores: cuando se hace clic en cualquiera de ellos, se invoca a `manejarSelecciónDeColor (colorSeleccionado)`.



Manejo de eventos de la versión anterior del Mente maestra

Les comentamos a los estudiantes: “En la versión anterior del Mente maestra, una vez que un jugador agregaba un color al código candidato había que mostrarlo en pantalla y, además, si se completaba el código propuesto se revelaba el código secreto, se mostraba la cantidad de coincidencias y se deshabilitaban los botones de colores. En la nueva versión, tenemos que permitir ocho intentos. ¿Alguno se anima a contar cómo tendría que ser ahora el comportamiento de la aplicación?”. Guiamos el intercambio para llegar a la conclusión de que, cuando se agrega un color hay que mostrarlo en la pantalla; si se completa un código candidato, hay que mostrar la cantidad de coincidencias; y solo si se han realizado los ocho intentos, hay que mostrar el código secreto y deshabilitar los botones de colores. A continuación, copiamos en el pizarrón las ideas delineadas.



Descripción del comportamiento esperado de la aplicación

Les indicamos a los estudiantes que resuelvan la tercera consigna, que pide modificar el programa para que se comporte de acuerdo a lo descrito. Para resolver el desafío, deben notar que para poder determinar si se han completado los ocho intentos hace falta una nueva variable global que indique cuál es el número de intento que está realizando el jugador. Podría llamarse, por ejemplo, `númeroDeIntento` y ser inicializada con un 1 –cuando el juego comienza el jugador puede realizar su primer intento–.

`inicializar global númeroDeIntento como 1`

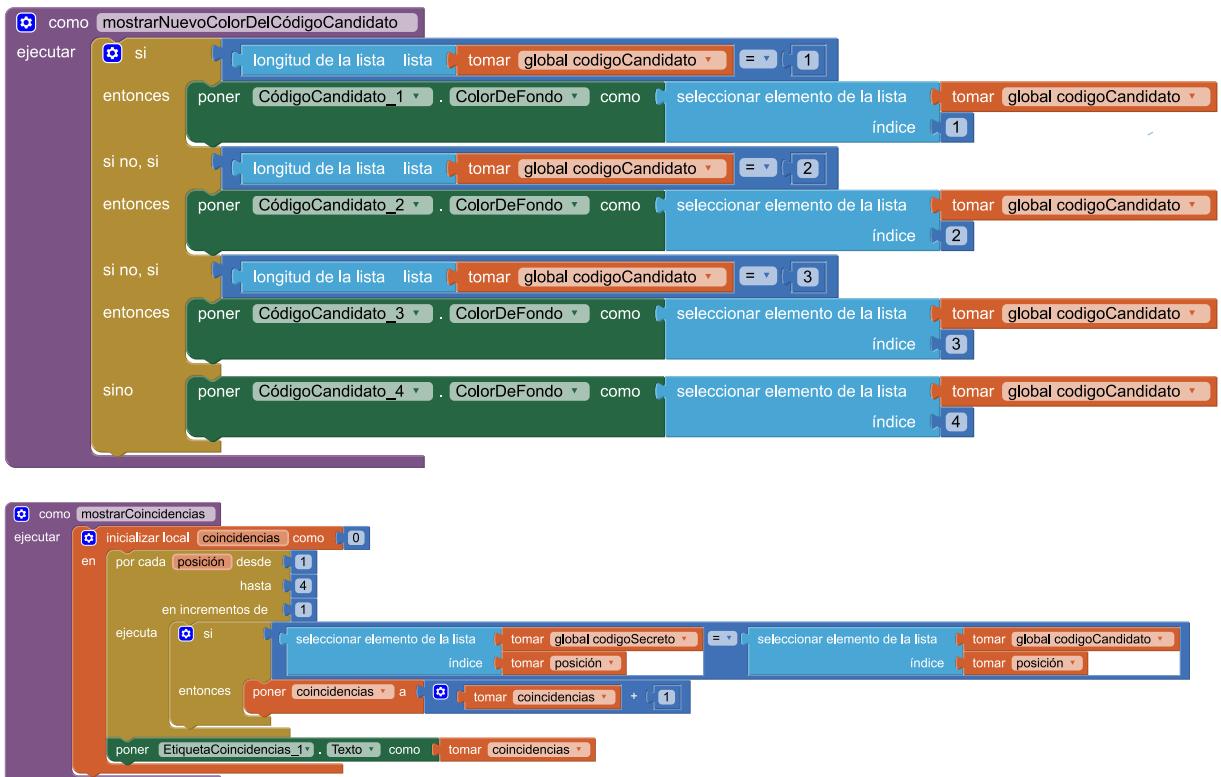
Variable que conserva el número de intento del jugador

Además, para completar la nueva versión de `manejarSelecciónDeColor (colorSeleccionado)`, hay que tener en cuenta que cada vez que se completa un código candidato hay que: (i) sumarle 1 a `númeroDeIntento` –el jugador comenzará el siguiente intento–, y (ii) vaciar la lista que representa el código candidato –el nuevo código se arma desde cero, es independiente del anterior–.



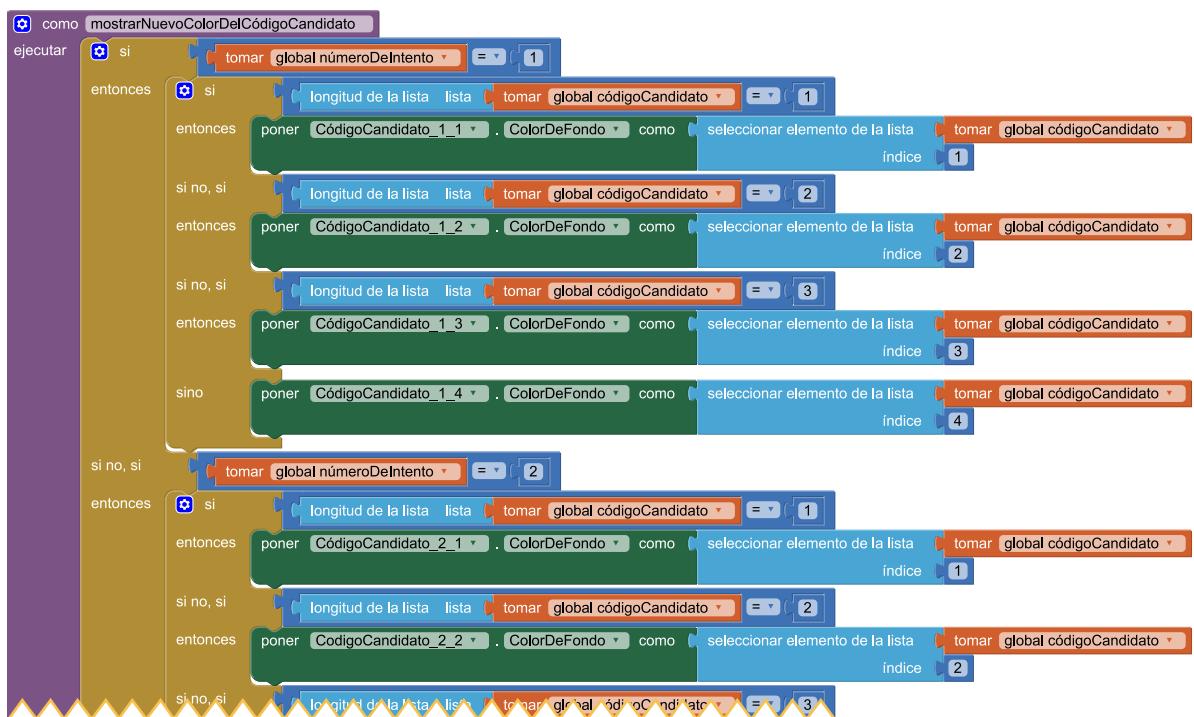
Nueva versión de `manejarselecciónDeColor (colorSeleccionado)`

Tal como vienen dados –por la versión anterior–, los procedimientos `mostrarNuevoColorDelCódigoCandidato` y `mostrarCoincidencias` solo modifican la primera fila de etiquetas del panel para los códigos candidatos. Por lo tanto, si se ejecuta la aplicación, no se mostrarían los ocho intentos uno debajo del otro.



Procedimientos de la versión anterior de la aplicación

En ambos casos, para determinar la etiqueta que hay que actualizar, alcanza con incorporar al análisis el número de intento del jugador. Así, por ejemplo, si el jugador ingresa la primera posición del segundo intento, hay que actualizar el color de la primera etiqueta de la segunda fila (*CódigoCandidato\_2\_1*); y si se trata de la cuarta posición del sexto intento, el color de la cuarta etiqueta de la sexta fila (*CódigoCandidato\_6\_4*) y el valor de las coincidencias (*EtiquetaCoincidencias\_6*).



Fragmento de `mostrarNuevoColorDelCódigoCandidato`



Fragmento de `mostrarCoincidencias`

### Consigna 4: al ganar, ya está

La cuarta consigna pide que el juego, además de terminar cuando se hayan alcanzado los ocho intentos, también finalice cuando el código secreto sea descubierto.

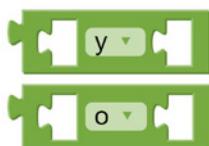
Antes de que comiencen a resolver el desafío, les proponemos diseñar grupalmente una solución que tenga en cuenta el nuevo requerimiento. Orientamos la discusión para concluir que, entonces, una partida puede finalizar por dos motivos: se descubre el código o se alcanzan ocho intentos. Copiamos en el pizarrón la estrategia que se muestra en la figura y los instamos a que modifiquen sus programas para resolver la consigna.

1. Actualizar en la pantalla el código candidato  
2. Si se completó un código candidato  
    2.1. Mostrar la cantidad de coincidencias  
    2.2. Si se adivinó el código secreto o fue el octavo intento  
        2.2.1. Mostrar el código secreto  
        2.2.2. Deshabilitar los botones de colores

El juego también finaliza si se descubre el código secreto

Lo primero que deben advertir los estudiantes es que, para resolver el desafío, es necesario modificar el procedimiento `manejarSelecciónDeColor (colorSeleccionado)`. Hasta aquí, el juego solo finaliza cuando se alcanzan los ocho intentos. Resta, entonces, construir una expresión que denote la condición “se adivinó el código secreto o se alcanzó el octavo intento”.

Al inspeccionar el entorno, los estudiantes observarán en *Bloques > Integrados > Lógica* dos bloques para armar expresiones booleanas a partir de otras más simples: `[ ] y [ ]` y `[ ] o [ ]`, que corresponden a las operaciones lógicas de conjunción y disyunción respectivamente. Dadas dos condiciones `c1` y `c2`, `[c1] y [c2]` es verdadera únicamente si ambas (`c1` y `c2`) lo son; en cambio, para que `[c1] o [c2]` sea verdadera, alcanza con que alguna de las dos sea verdadera. Si hiciese falta, orientamos a los estudiantes en el uso de estos dos bloques.



Bloques para conjunción y disyunción lógica

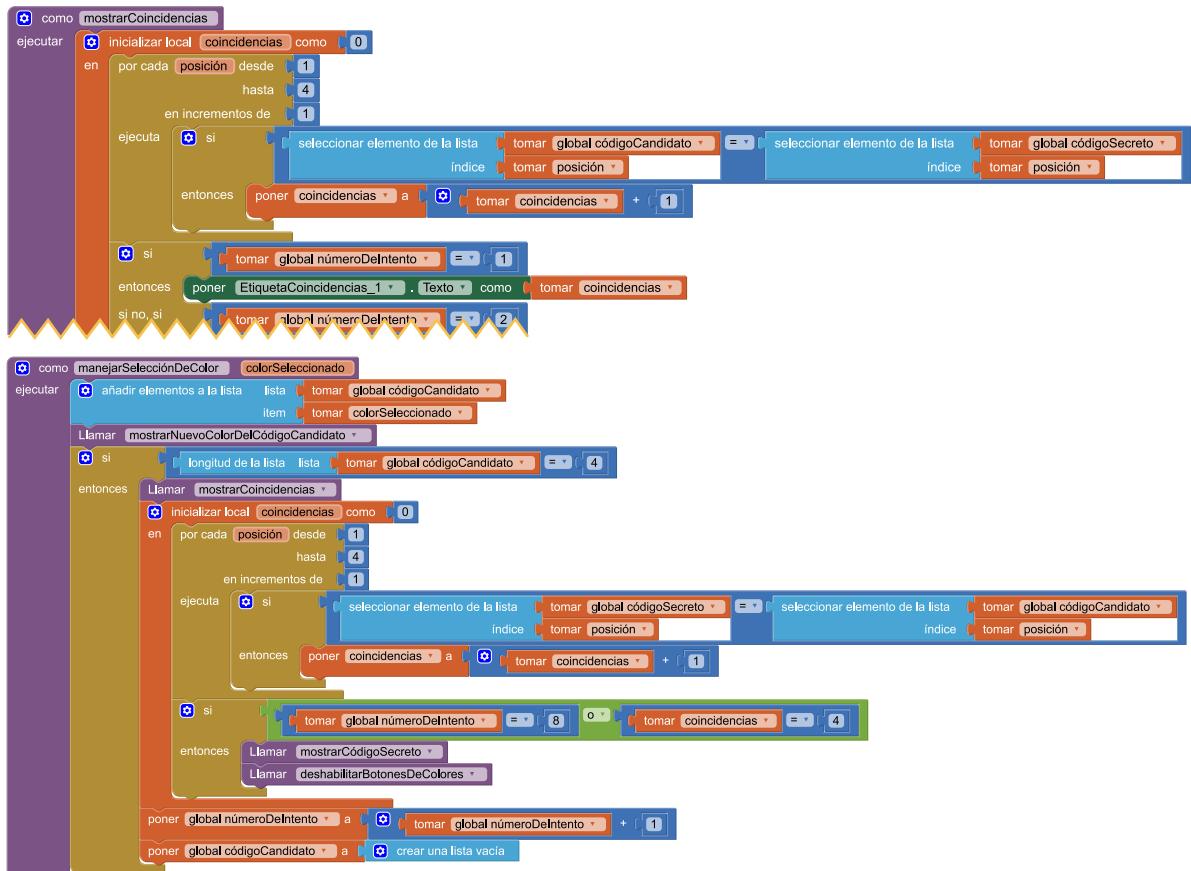
Una vez identificado el bloque de la disyunción lógica, se lo puede incorporar a la alternativa condicional.



Falta expresión que denota que se adivinó el código secreto

Se incorpora la disyunción a la alternativa

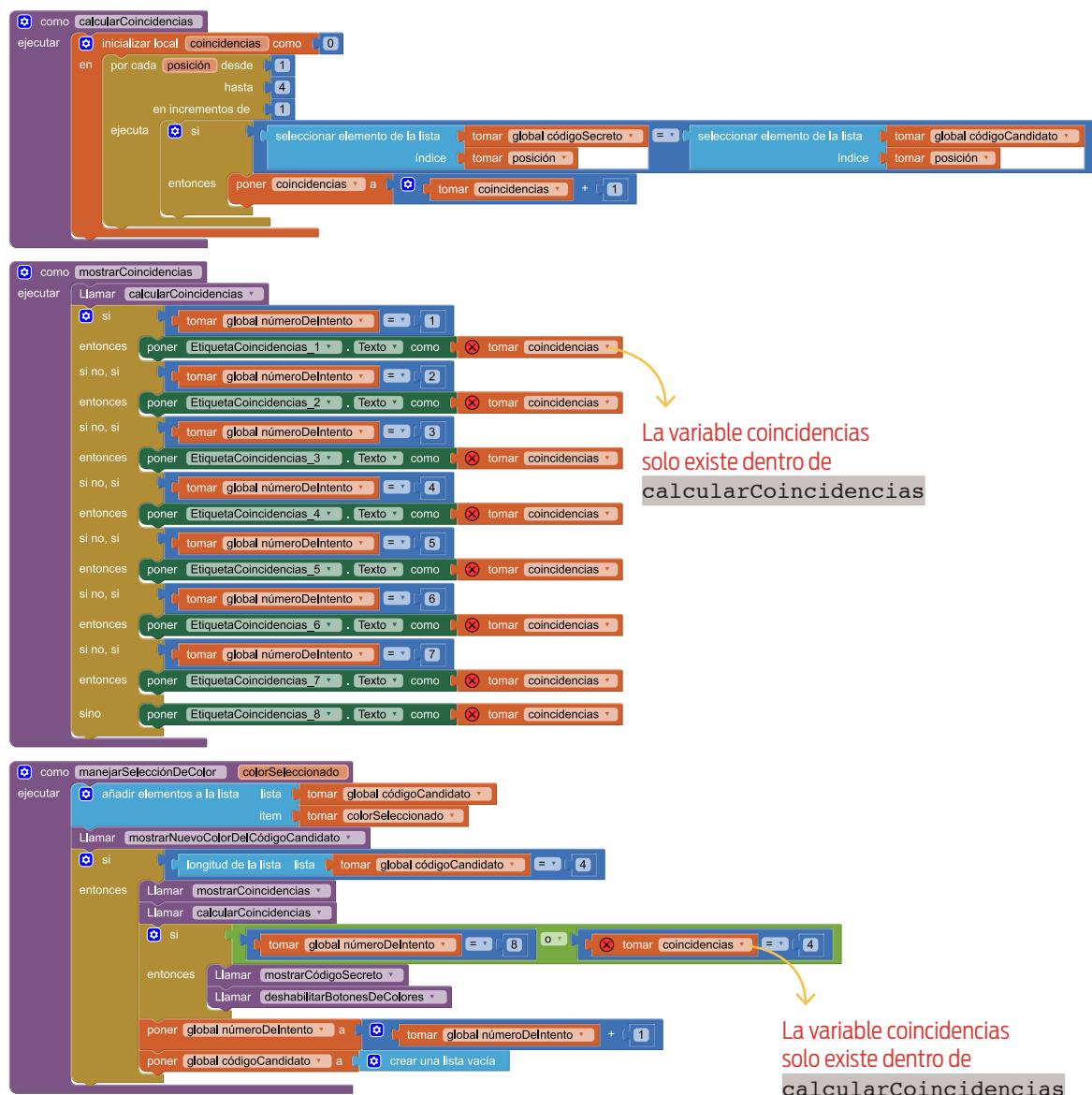
Resta definir la expresión que refleje que el código secreto ha sido descubierto. O, equivalentemente, que los cuatro colores del código candidato coincidan con los del código secreto. En `mostrarCoincidencias` se realiza un conteo de las posiciones en las que ambos códigos son iguales, con lo que es probable que algunos estudiantes armen un esquema similar dentro de `manejarSelecciónDeColor (colorSeleccionado)`.



Reproducción del enfoque para cálculo de coincidencias

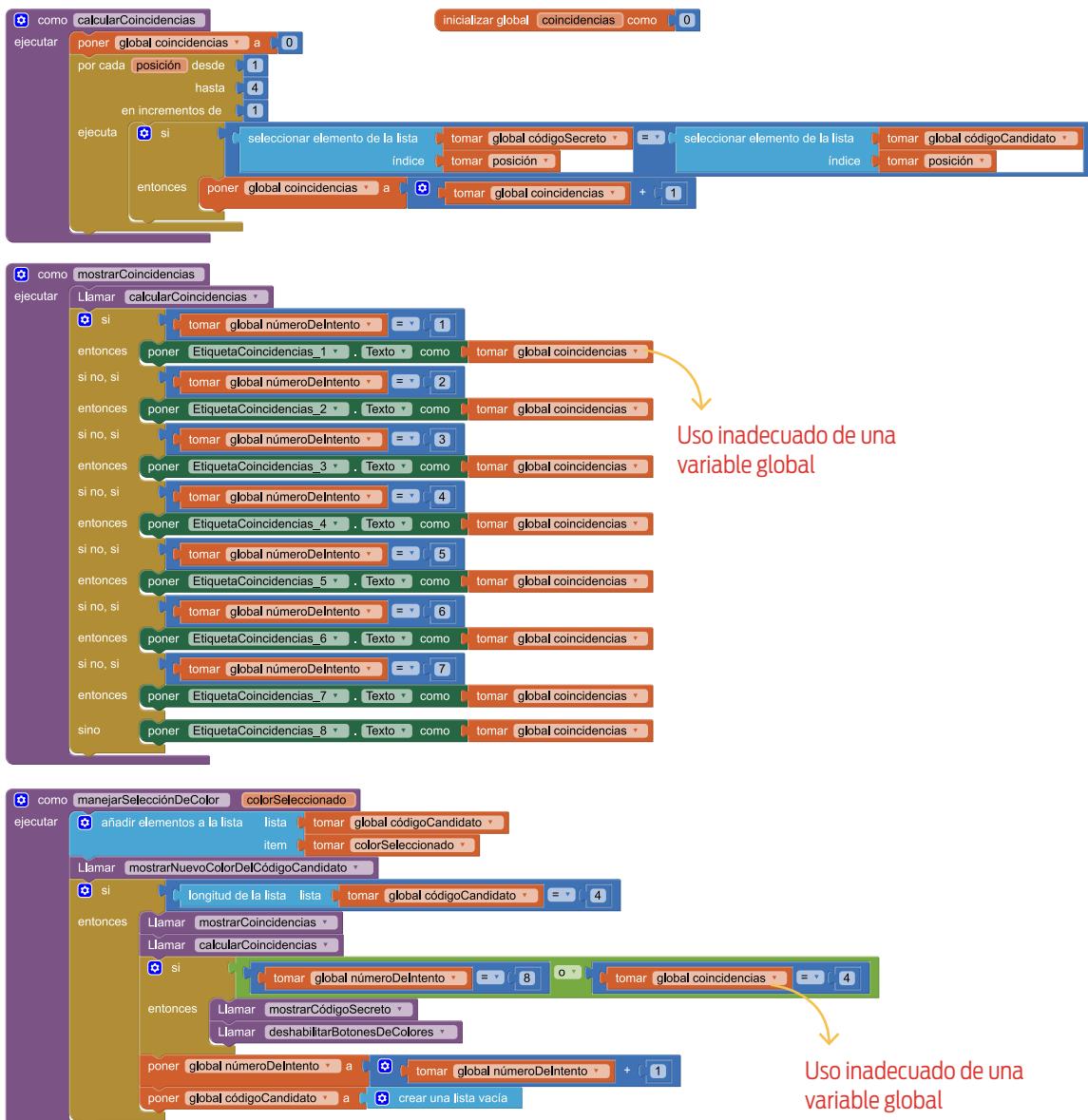
El procedimiento `manejarSelecciónDeColor (colorSeleccionando)` es funcionalmente correcto, aunque adolece de un problema grave: al observarlo, no resulta sencillo comprender qué hace. A aquellos estudiantes que hayan propuesto esta solución (u otra similar), les hacemos notar que en `mostrarCoincidencias` y `manejarSelecciónDeColor (colorSeleccionando)` hay fragmentos muy similares y les indicamos que piensen una solución alternativa.

Es probable que, a partir de reconocer que el conjunto de bloques que se repite en ambos procedimientos, lo primero que piensen sea extraer la porción común en un procedimiento nuevo (al que podrían llamar, por ejemplo, `calcularCoincidencias`) e invocarlo tanto desde `mostrarCoincidencias` como desde `manejarSelecciónDeColor (colorSeleccionando)`.



Las variables locales de un procedimiento no pueden ser referenciadas en otro procedimiento

Como la variable `coincidencias` solo existe dentro del procedimiento `calcularCoincidencias`, no puede ser referenciada ni desde `mostrarCoincidencias` ni desde `manejarSelecciónDeColor (colorSeleccionando)`. Frente a este problema, algún estudiante podría proponer que `coincidencias` sea una variable global (en lugar de local).

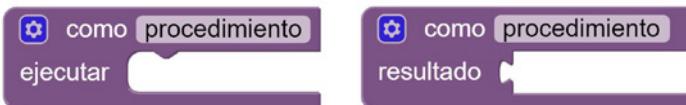


Se guardan las coincidencias en una variable global

Si bien de este modo el programa funciona, el uso de variables globales solo se justifica cuando hay un valor que va evolucionando a lo largo de la ejecución de un programa (por ejemplo, el puntaje que va acumulando un jugador al desarrollarse una partida de un juego). Sin embargo, en esta propuesta, se usa una para conservar el resultado de un cálculo, cuyo valor solo tiene una relevancia local (inmediatamente después de que `calcularCoincidencias` es invocado). En caso de que los estudiantes propongan resolver el problema usando una variable global, les sugerimos que exploren el entorno de App Inventor en busca de una solución alternativa.

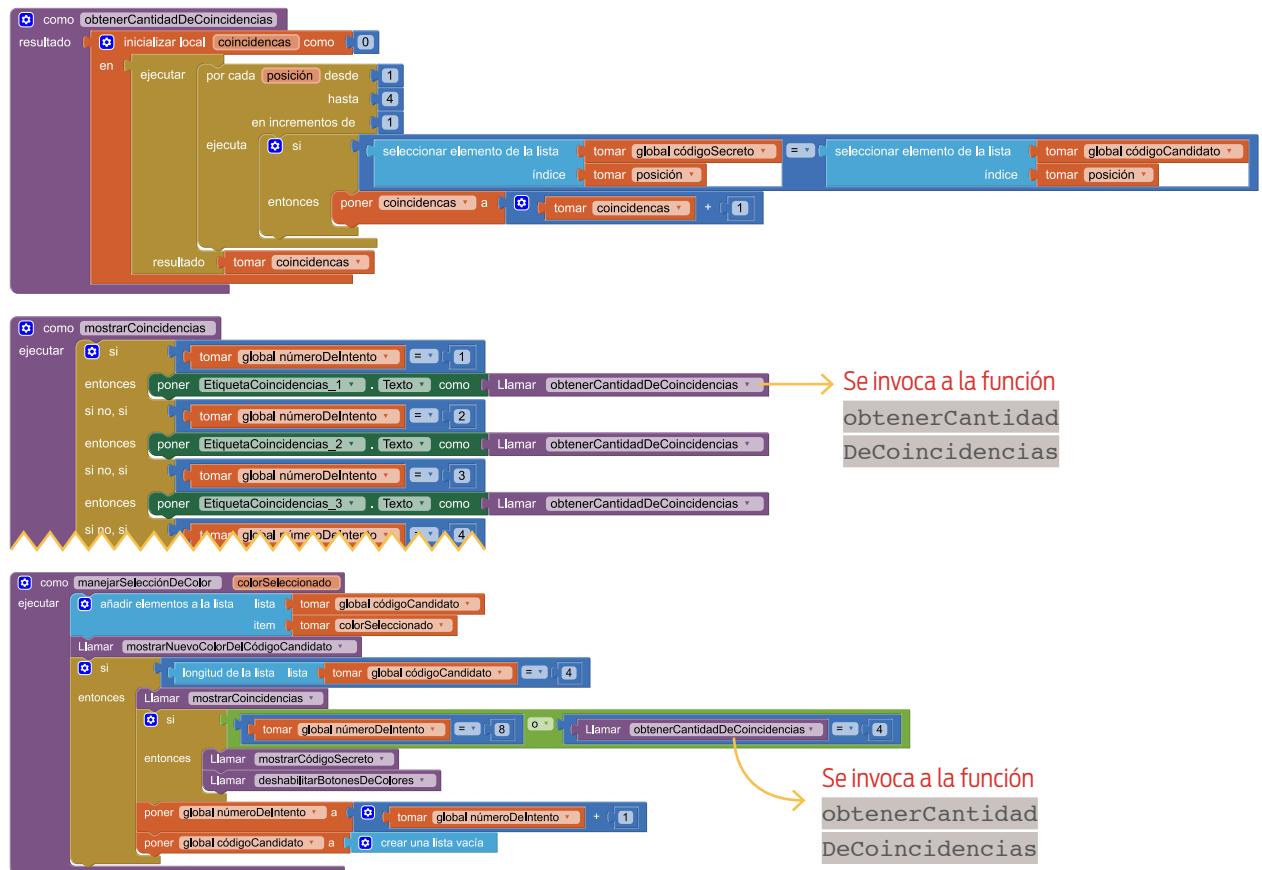
En **Bloques > Integrados > Procedimientos** se encuentran los bloques `como (procedimiento) / ejecutar { }` y `como (procedimiento) / resultado [ ]`. Mientras que el primero se utiliza

para realizar una tarea, el segundo se usa para calcular y retornar un valor. En los lenguajes de programación, a estos últimos se los refiere como **funciones**,<sup>1</sup> cuya característica principal es que producen nueva información relevante para un programa.



Bloques para definir un procedimiento y una función

Para calcular la cantidad de coincidencias entre un código candidato y el código secreto, lo adecuado es resolverlo en una función, que calcule y retorne el valor buscado (podría llamarse `obtenerCantidadDeCoincidencias`). A continuación se muestra una posible solución.

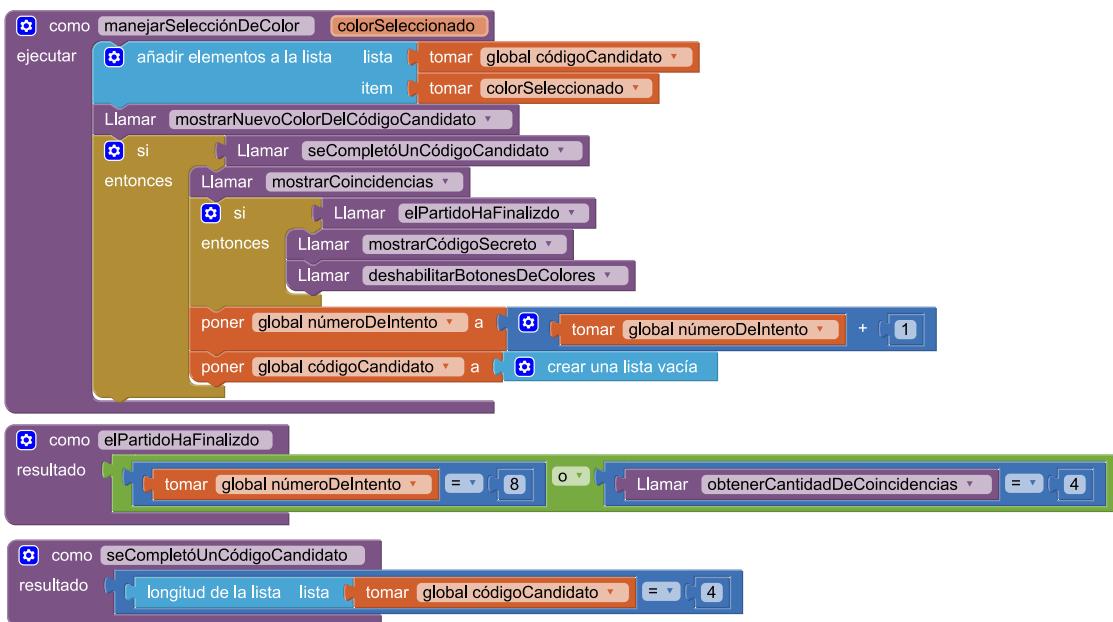


Solución que incorpora una función

Como se ha visto, los procedimientos se usan para encapsular un conjunto de acciones que componen una unidad de sentido y, al ponerles un nombre adecuado, se obtiene un programa cuya lectura resulta más clara. Con las funciones puede hacerse algo similar y, también, obtener como resultado un programa que sea más fácil de entender. En la figura a continuación se muestra una solución que incorpora dos

<sup>1</sup>En App Inventor no se hace una diferenciación explícita entre procedimientos y funciones; a ambos se los llama procedimientos.

nuevas funciones: una que chequea si se ha completado un código candidato y, otra, que chequea si ha finalizado la partida. De este modo, al leer el procedimiento `manejarSelecciónDeColor (colorSeleccionado)` resulta sencillo comprender lo que hace.

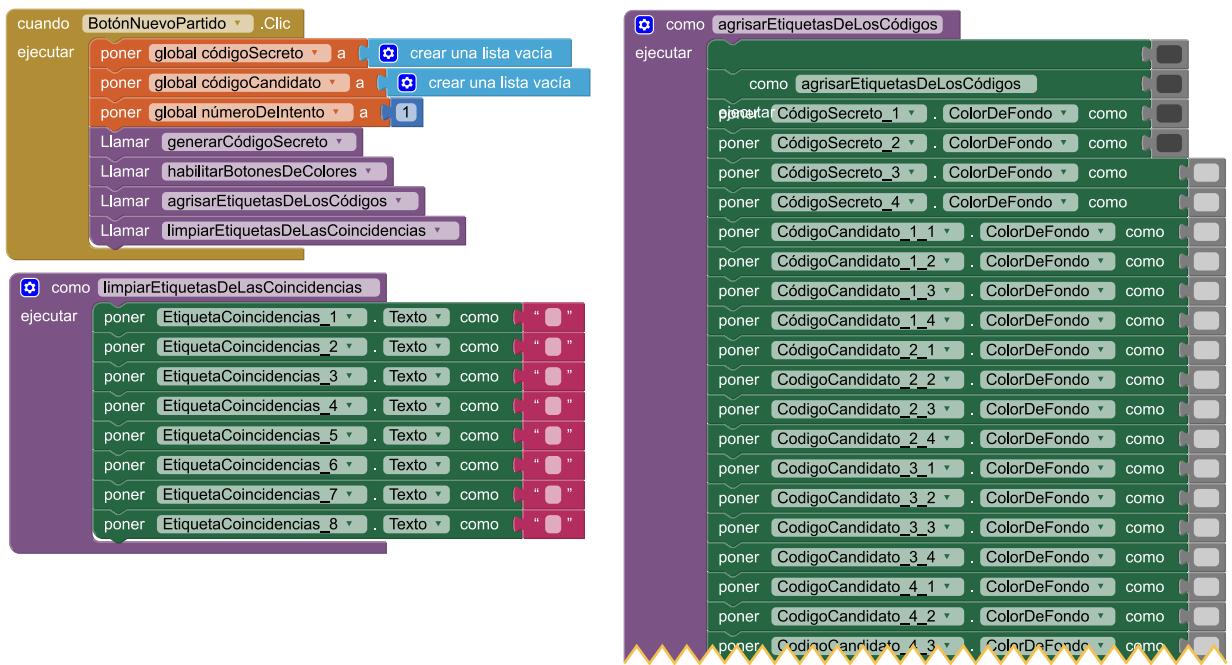


Definición de nuevas funciones

### Consigna 5: un nuevo partido

La quinta y última consigna pide que, al apretar el botón *Nuevo partido*, todo quede dispuesto para empezar una nueva partida. La solución es muy similar a la que ya tienen de la actividad anterior. Concretamente hay que: (i) vaciar las listas `códigoSecreto` y `códigoCandidato` y establecer `númeroDeIntento` en 1, (ii) generar un nuevo código secreto, (iii) agrisar las etiquetas que muestran los códigos en la pantalla, y (iv) vaciar los textos de las etiquetas que muestran las cantidades de coincidencias entre los distintos intentos de un usuario y el código secreto.

Para resolver el desafío se pueden usar los procedimientos `generarCódigoSecreto` y `habilitarBotonesDeColores` tal como vienen dados desde la versión anterior de la aplicación. Por su parte, puede adaptarse `agrisarEtiquetasDeLosCódigos` para incluir las etiquetas de los ocho códigos candidatos; y crear un nuevo procedimiento `limpiarEtiquetasDeCoincidencias` que deje vacíos los textos de todas las etiquetas que muestran la cantidad de posiciones en que coinciden los códigos candidatos y el código secreto.



Reinicio del Mente maestra

## CIERRE

Repasamos con los estudiantes que, como para determinar si una partida finaliza hubo que chequear si se verificaba al menos una de dos condiciones (se completaron los ocho intentos o se descubrió el código secreto), en esta actividad hemos utilizado el operador lógico de disyunción. Además, comentamos que cuando se tenga que chequear si se cumplen dos condiciones (y no al menos una de las dos), hay que usar el operador de conjunción. Finalmente, subrayamos que, así como los procedimientos permiten encapsular acciones, las funciones permiten encapsular operaciones que retornan un valor; es decir, cálculos que producen información.

NOMBRE Y APELLIDO:

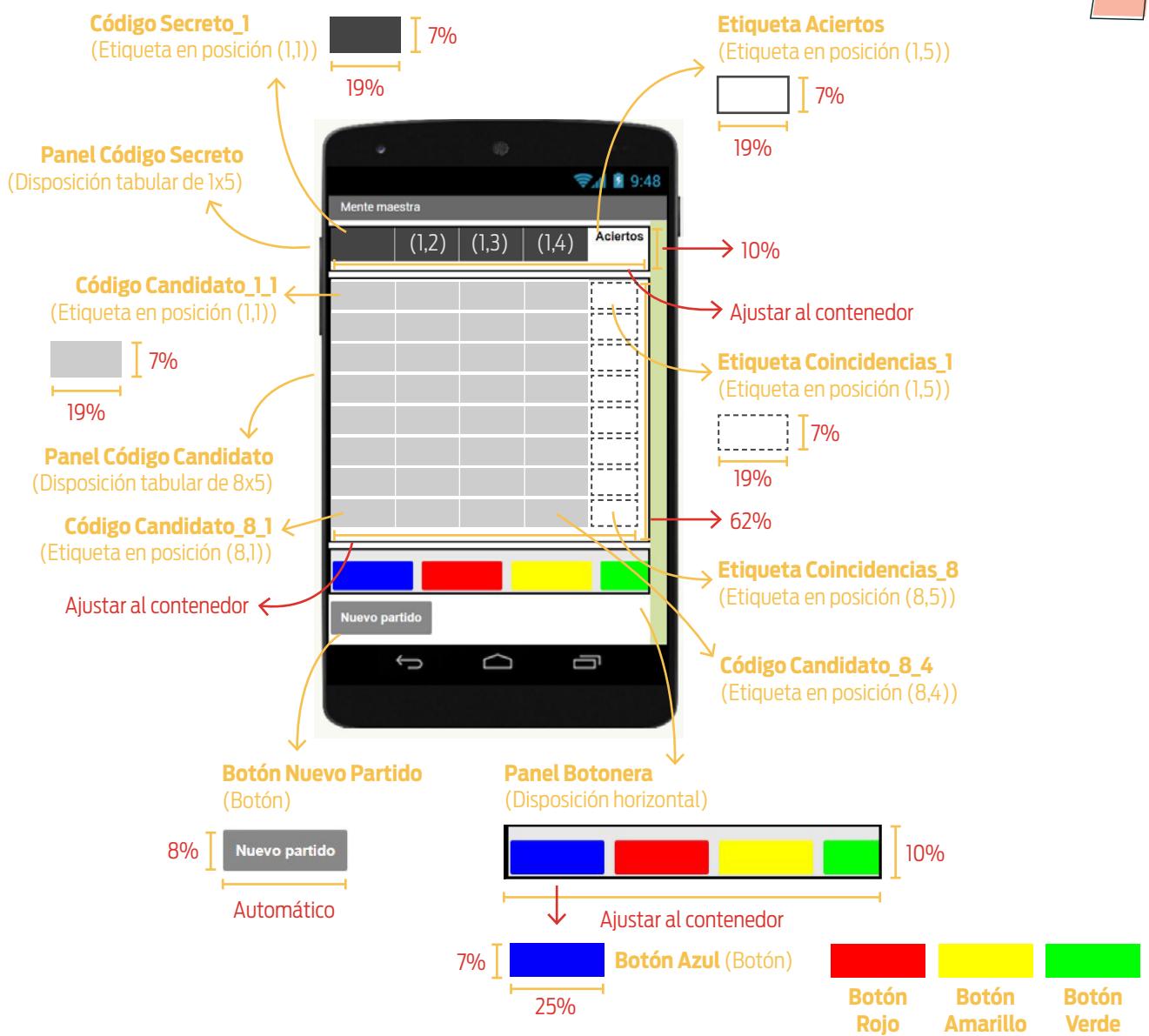
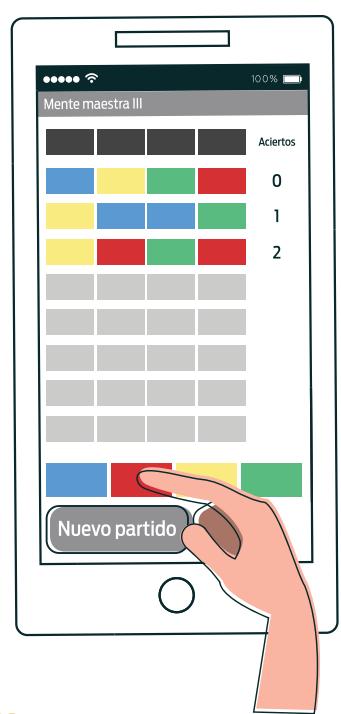
CURSO:

FECHA:

# MENTE MAESTRA, PARTE III

Y ahora sí: códigos de cuatro colores y ocho intentos.  
¡A armar la versión completa del Mente maestra!

1. Comenzá armando la interfaz gráfica. En la imagen están los componentes y sus dimensiones. En la tabla, las propiedades que tenés que cambiar para que se vea igual a la que te mostramos. Seguí las indicaciones para componerla.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Screen1	Pantalla	Título	Screen1	Mente maestra
Panel Código Secreto	Disposición tabular	Columnas	2	5
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	1
Código Secreto_1 <sup>1</sup>	Etiqueta	Color	Ninguno	Gris oscuro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Aciertos	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	Aciertos
		Posición del texto	Izquierda	Centro
Panel Código Candidato	Disposición tabular	Columnas	2	5
		Alto	Automático	62%
		Ancho	Automático	Ajustar al contenedor
		Registros	2	8
Código Candidato_1_1 <sup>2</sup>	Etiqueta	Color de fondo	Ninguno	Gris claro
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
Etiqueta Coincidencias_1 <sup>3</sup>	Etiqueta	Negrita	(ninguno)	✓
		Alto	Automático	7%
		Ancho	Automático	19%
		Texto	Texto para etiqueta	(ninguno)
		Posición del texto	Izquierda	Centro

<sup>1</sup>Las restantes etiquetas del código secreto requieren los mismos cambios.<sup>2</sup>Las restantes etiquetas del código candidato requieren los mismos cambios.<sup>3</sup>Las restantes etiquetas para mostrar la cantidad de aciertos de un intento requieren los mismos cambios.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

NOMBRE SUGERIDO	TIPO DE COMPONENTE	PROPIEDAD	VALOR PREVIO	VALOR NUEVO
Panel Botonera	Disposición horizontal	Disp. horizontal	Izquierda	Centro
		Disp. vertical	Arriba	Abajo
		Alto	Automático	10%
		Ancho	Automático	Ajustar al contenedor
Botón Azul	Botón	Color de fondo	Por defecto	Azul
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Rojo	Botón	Color de fondo	Por defecto	Rojo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Amarillo	Botón	Color de fondo	Por defecto	Amarillo
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Verde	Botón	Color de fondo	Por defecto	Verde
		Alto	Automático	7%
		Ancho	Automático	25%
		Texto	Texto para botón	(ninguno)
Botón Nuevo Partido	Botón	Color de fondo	Por defecto	Gris
		Negrita	(ninguno)	✓
		Alto	Automático	8%
		Texto	Texto para botón	Nuevo partido
		Color de texto	Por defecto	Blanco

NOMBRE Y APELLIDO:

CURSO:

FECHA:

- 2.** Ejecutá la aplicación. ¿Con qué te encontraste? ¿Cuántos intentos tuviste para descubrir el código secreto? ¿Por qué?

---

---

---

- 3.** El juego tienen que permitir que un usuario cuente con ocho intentos para develar los cuatro colores ocultos. Usando lenguaje coloquial, delineá una estrategia para conseguirlo.

Programá en App Inventor la estrategia propuesta.

**UNA AYUDA**

El procedimiento `manejarSelecciónDeColor (colorSeleccionado)` describe la estrategia de la versión anterior, en la que solo había un intento. A lo mejor te sirve como referencia.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

4. No solo cuando se completan los ocho intentos el juego tiene que terminar. ¡Si el jugador descubre el código secreto tampoco tiene sentido seguir! Modificá el programa para que también finalice cuando el jugador triunfa.

**PARA QUE TENGAS EN CUENTA**

En App Inventor hay dos bloques para armar expresiones booleanas que se construyen a partir de otras más simples: [ ] y [ ] y [ ] o [ ]. La primera corresponde a la operación lógica de **conjunción**, y la segunda, a la de **disyunción**. ¿Te imaginás cómo usar alguna de ellas para resolver la consigna?



5. Por último, ocúpate de que cuando se presione el botón Nuevo partido se pueda jugar otra vez. ¿Qué cambios tuviste que hacer para conseguirlo?

---

---

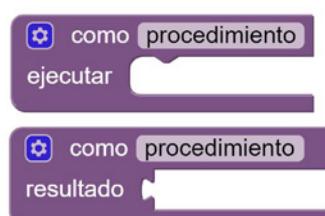
---

---

---

**PROCEDIMIENTOS Y FUNCIONES**

¿Sabés cuál es la diferencia entre **como (procedimiento) / ejecutar { }** y **como (procedimiento) / resultado [ ]**? Mientras que el primero se utiliza para realizar una tarea, el segundo se usa para calcular y retornar un valor. En los lenguajes de programación, a estos últimos se los refiere como **funciones**, cuya característica principal es que producen información nueva que es relevante para un programa. ¡Incorporá funciones en tu programa y elegí concientudamente sus nombres!



# 05

# REPRESENTACIÓN DE LA INFORMACIÓN

## SECUENCIA DIDÁCTICA 1

- REPRESENTACIÓN DE NÚMEROS  
Y TEXTO
- Luces binarias
  - Cartas y números binarios
  - Equis raya

## SECUENCIA DIDÁCTICA 2

- REPRESENTACIÓN DE IMÁGENES  
DIGITALES
- Imágenes en blanco y negro
  - Banderas de colores

## SECUENCIA DIDÁCTICA 3

- COMPRESIÓN DE IMÁGENES DIGITALES
- Encogemos imágenes
  - Experimentamos con imágenes  
digitales

Las computadoras son máquinas que manipulan **información**. Nos permiten ver, escuchar, crear y editar información: escribir un documento, editar imágenes, ver y grabar videos o escuchar música, entre otras actividades. Por ello resulta fundamental poder representar la información de alguna manera dentro de la memoria RAM, en el disco o para enviarla a través de una red.

En este capítulo, se presentan actividades que trabajan sobre las ideas subyacentes usadas por los sistemas digitales para la representación de números, texto e imágenes.



## Secuencia Didáctica 1

# REPRESENTACIÓN DE NÚMEROS Y TEXTO

Las computadoras modernas pueden almacenar y manipular información solo si ésta se encuentra codificada en forma binaria. Sin embargo, los seres humanos utilizamos otras formas de representación de la información. Para que las computadoras sean útiles y puedan manipular información resulta necesario traducirla a valores binarios. Análogamente, para que la información manipulada por las computadoras resulte comprensible para las personas debe llevarse a cabo el proceso inverso.

Las actividades que se presentan en esta secuencia didáctica apuntan a responder el siguiente interrogante: ¿cómo se representa la información en la computadora? Con tal propósito, se presentan los sistemas binarios y formas de codificar números y texto.

### ..... **OBJETIVOS**

- Comprender un sistema binario.
  - Representar información numérica.
  - Representar información textual.
- .....

## Actividad 1

### Luces binarias



#### OBJETIVO

- Presentar un sistema de codificación binario.

#### MATERIALES



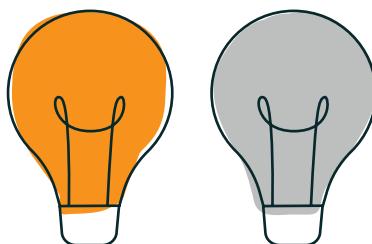
Fuente de luz que se pueda prender y apagar



Ficha para estudiantes

#### DESARROLLO

Comenzamos la clase mostrando que una lamparita puede estar solo en dos estados posibles: encendida o apagada. Puede usarse la luz del aula, una linterna, la función linterna de un teléfono inteligente o cualquier otra fuente de luz que pueda prenderse y apagarse. Luego, preguntamos a los estudiantes: “¿En cuántos estados puede encontrarse una lamparita?”. Se espera que respondan que son dos los estados posibles, tal como se muestra en la figura.



Los dos estados posibles en que puede encontrarse una lamparita

A continuación, repartimos la ficha a los estudiantes y les pedimos que resuelvan el ejercicio, en el que se deben identificar las distintas combinaciones posibles que existen al contar con dos lamparitas dispuestas en un determinado orden. La solución se muestra en la siguiente figura.



→ Apagada - Apagada



→ Apagada - Encendida



→ Encendida - Apagada

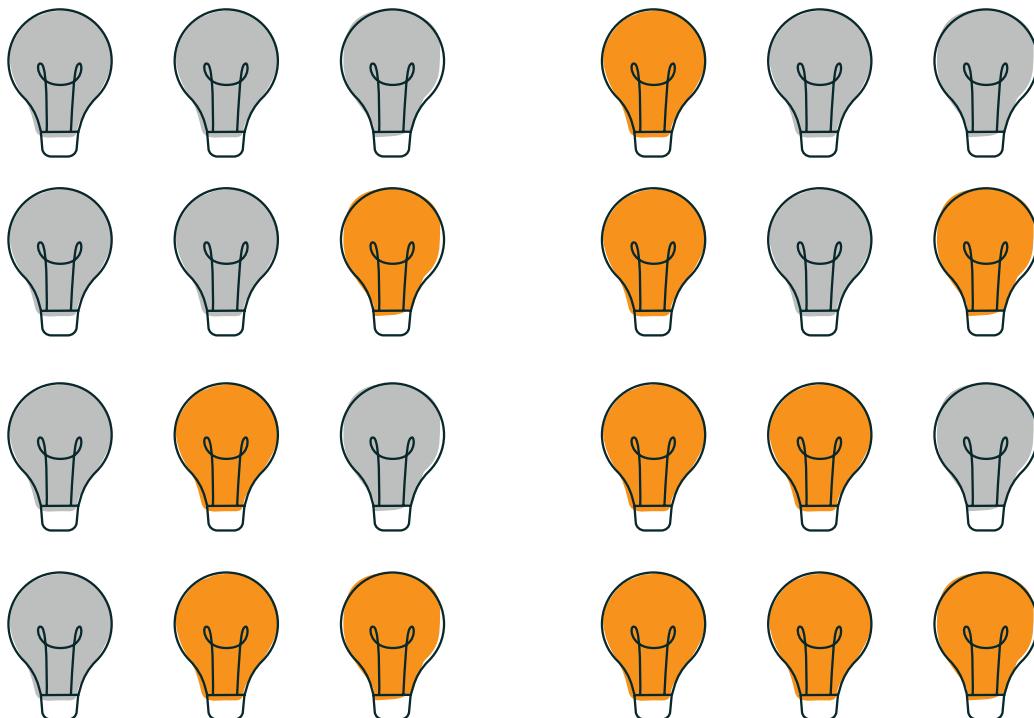


→ Encendida - Encendida

Estados posibles usando dos lamparitas

Se espera que los estudiantes adviertan que, al usar dos lamparitas dispuestas tal como se muestran en la ficha, hay cuatro combinaciones posibles: apagada-apagada, apagada-encendida, encendida-apagada y encendida-encendida.

Luego, preguntamos a toda la clase: “¿Cuántas combinaciones habría con tres lamparitas? ¿Cuál es la relación con la cantidad de combinaciones que teníamos con dos lamparitas?”. A partir de las respuestas obtenidas, hacemos una puesta en común explicando que, cuantas más lamparitas se tengan, mayor es la cantidad de combinaciones. Retomando las preguntas formuladas, mostramos que con tres lamparitas se obtienen ocho combinaciones distintas.



Los 8 estados posibles usando tres lamparitas

La cantidad de combinaciones posibles es ocho, el doble de las que se tenían con dos lamparitas. La tercera lamparita puede estar apagada o encendida para cada uno de las cuatro combinaciones que teníamos con dos lamparitas. Preguntamos: “¿Y si disponemos de cuatro lamparitas? ¿Cuál es la relación con la cantidad de combinaciones que teníamos con tres lamparitas?”. Se espera que, siguiendo el mismo esquema de razonamiento, los estudiantes puedan reconocer que con cuatro lamparitas se tienen dieciséis combinaciones diferentes, el doble de las que se tenían con tres. Análogamente, con cinco lamparitas se tienen treinta y dos combinaciones. Planteamos, entonces: “Si se pretende conseguir sesenta y cuatro combinaciones, ¿cuántas lamparitas se necesitan?” Una posibilidad para responder esta pregunta consiste en continuar con el razonamiento anterior. Esto es, duplicar la cantidad de combinaciones al agregar una lamparita. Con cinco lamparitas se tienen treinta y dos combinaciones y, por lo tanto, con seis lamparitas se tienen sesenta y cuatro combinaciones.

### CIERRE

A modo de cierre, mostramos la relación que existe entre la operación matemática de potenciación y la cantidad de valores que se pueden representar usando elementos que pueden estar en dos estados distinguibles. Al usar 1 lamparita, hay  $2 = 2^1$  combinaciones posibles; al usar 2 lamparitas,  $4 = 2^2$  combinaciones distintas; con 3,  $8 = 2^3$ ; y así sucesivamente. En general, al usar  $n$  lamparitas, se tendrán  $2^n$  combinaciones, y en consecuencia se podrán representar  $2^n$  valores diferentes.

---

NOMBRE Y APELLIDO:

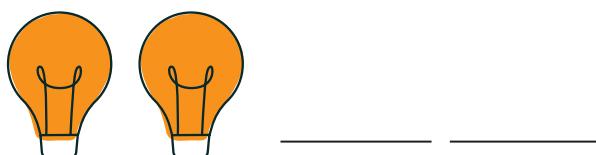
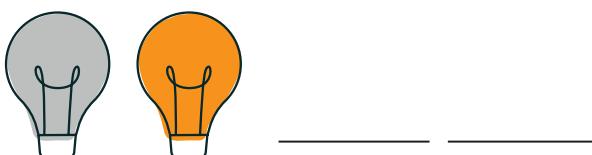
CURSO:

FECHA:

# LUCES BINARIAS

¿Sabías que, usando un elemento que pueda estar en 2 estados diferentes, podemos representar un montón de cosas? ¡Vamos a trabajar con luces para que veas qué poco hace falta para poder decir mucho!

- 1.** Completá las combinaciones posibles de acuerdo a los estados de 2 lamparitas. Usá las palabras *encendida* y *apagada*.



- 2.** Si usáramos cada combinación para representar un número, ¿cuántos números distintos podríamos representar? ¡Y si en lugar de usar 2 lamparitas usáramos 3?

---

---

- 3.** ¿Cuántas lamparitas hacen falta para representar 64 números distintos? ¡Y para representar 128?

---

---

## SISTEMA BINARIO

Todos los datos que usa una computadora se almacenan usando solo dos valores. Internamente, la computadora usa dos niveles de voltaje claramente diferenciables, y para referirnos a ellos solemos usar los dígitos 0 y 1; pero también podríamos usar sí y no, o blanco y negro, o puño y palma, o una lamparita encendida y una apagada. Solo es necesario elegir dos representaciones diferentes para que sea posible distinguirlas sin ninguna duda.

01

## Actividad 2

### Cartas y números binarios<sup>1</sup>



GRUPAL (5)

#### OBJETIVO

- Representar números en un sistema binario.

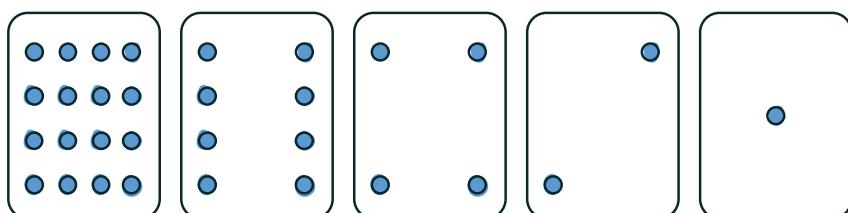
#### MATERIALES

Un juego de 5 cartas en tamaño A4 o superior

Ficha para estudiantes

#### DESARROLLO

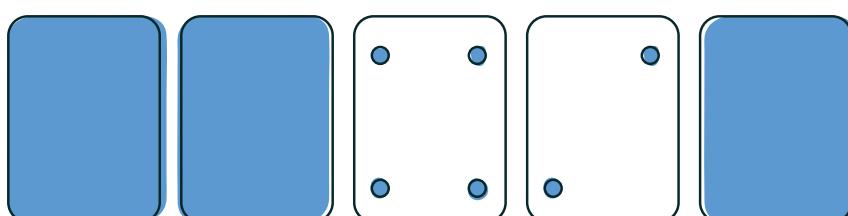
Para el desarrollo de la actividad, les pediremos a los estudiantes que formen grupos de 5 y luego solicitamos a uno de los grupos que pase al frente. Entregamos a ese grupo un juego de cartas como el de la figura, preferentemente en tamaño A4 o superior, de forma tal que el resto de la clase pueda observarlas sin dificultad.



Juego de 5 cartas de potencias de 2

Cada estudiante del grupo debe tener una de las cartas, ubicados en el orden que se muestra en la figura. Preguntamos a toda la clase: “¿Qué observan respecto del número de puntos de las cartas? Si hubiera una carta más a la izquierda, ¿cuántos puntos tendría?”. Se espera que los estudiantes adviertan que cada carta tiene el doble de puntos que la carta que está a su derecha y que, si la secuencia tuviese una carta más a la izquierda, esta tendría 32 puntos.

A continuación, pedimos al grupo que está en el frente que solo el segundo y el tercer estudiante, comenzando desde la derecha, muestren la carta del lado de los puntos. El resto de la clase verá las cartas tal como se muestra en la figura.



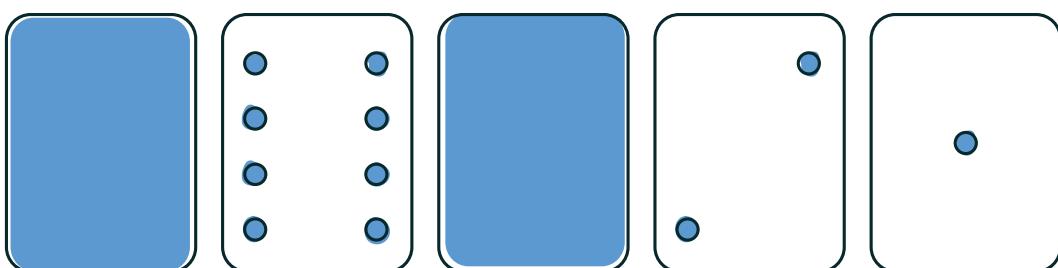
Disposición de las cartas para que haya solo 6 puntos visibles

Preguntamos a la clase qué número representan las cartas dispuestas de esta manera. Es probable que los estudiantes respondan que esta secuencia representa el número 6, que es la cantidad de puntos visibles, lo cual es correcto. Seguidamente les preguntamos: “¿Cómo pueden representar el 0?”. La respuesta correcta es: “Con todas las cartas dadas vuelta, ocultando todos los puntos”.

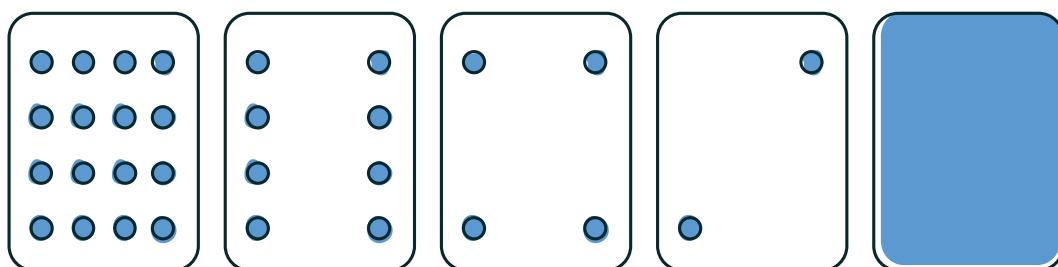
<sup>1</sup> Adaptación de la actividad "Count the dots" de CS Unplugged, disponible en <http://bit.ly/2mc82uo>.

Para continuar la actividad, repartimos la ficha a los estudiantes y les pedimos que recorten las cartas. Les proponemos a los grupos que indiquen qué cartas mostra-

rían y cuáles ocultarían para representar los números 11 y 30. Se espera que los estudiantes dispongan las cartas tal como se muestra en las siguientes figuras.



Representación del número 11



Representación del 30

Luego, pedimos a los estudiantes que representen todos los números que puedan, comenzando por el 0 y en orden ascendente. Preguntamos: “¿Cuál es el número más grande que pudieron representar?”. Deberían poder concluir que el mayor número representable es el 31, que se alcanza cuando todas las cartas muestran los puntos.

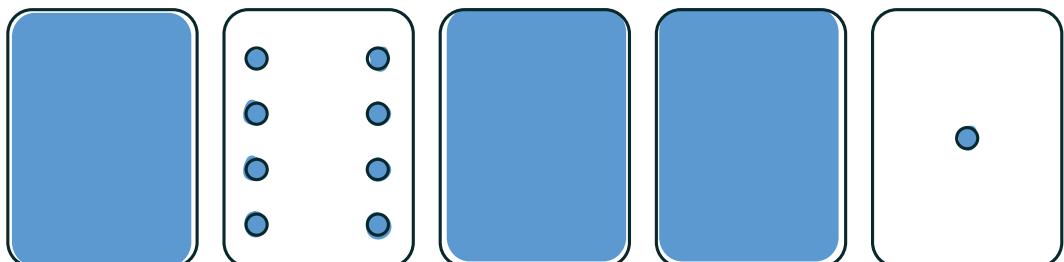
A continuación, requerimos que expresen la cantidad de puntos de cada carta como una potencia de 2. Indagamos: “¿Qué valor tiene el exponente en cada caso?”. Con nuestra ayuda, podrán deducir que la carta de más a la derecha tiene exponente 0, la siguiente 1 y así sucesivamente hasta 4. En caso de que los estudiantes tengan problemas con esta pregunta, se los puede ayudar mostrándoles un esquema como el de la página siguiente, que asocia cada carta con una posición en la secuencia.

(2 <sup>4</sup> )	(2 <sup>3</sup> )	(2 <sup>2</sup> )	(2 <sup>1</sup> )	(2 <sup>0</sup> )	CANTIDAD DE PUNTOS
4	3	2	1	0	POSICIÓN

Relación entre la cantidad de puntos de las cartas y las potencias de 2

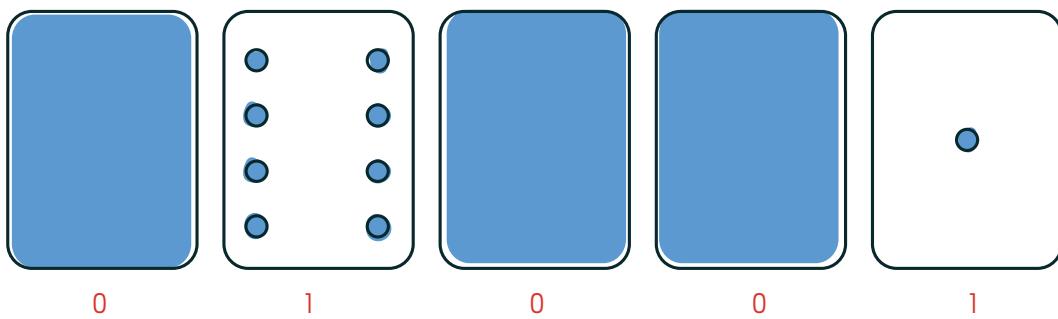
La cantidad de puntos de las cartas se puede expresar de la siguiente manera:  $2^0 = 1$  punto;  $2^1 = 2$  puntos;  $2^2 = 4$  puntos;  $2^3 = 8$  puntos, y  $2^4 = 16$  puntos.

Pedimos a otro de los grupos que pase al frente y les indicamos que solo el primero y el cuarto estudiante, comenzando desde la derecha, muestren la carta del lado de los puntos, mientras que los demás no mostrarán sus cartas. De este modo, el resto de la clase vería las cartas tal como se muestra en la figura:



Disposición de las cartas para que queden 9 puntos visibles

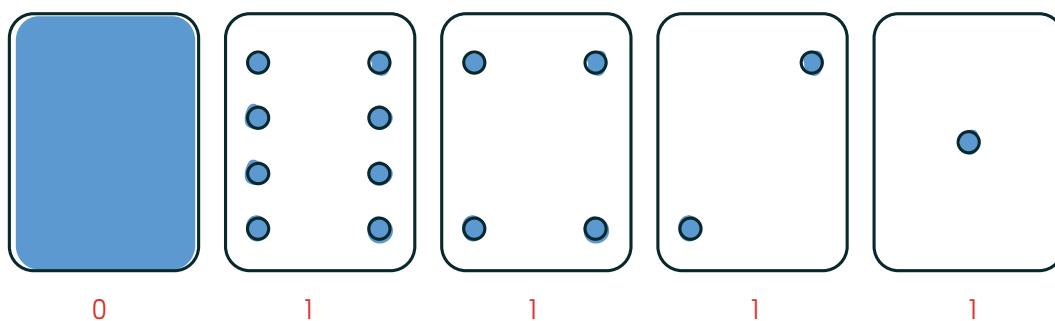
A lo largo de toda la actividad, cada carta ha estado o bien mostrando el frente o bien mostrando el dorso. Es decir que cada carta puede estar en uno y solo uno de esos dos estados. Les decimos: "Si en lugar de usar cartas usáramos el dígito 0 para representar que una carta muestra el dorso y el dígito 1 para representar que una carta muestra el frente, ¿qué secuencia de ceros y unos resultaría para las cartas exhibidas?". La respuesta es 01001, como se ve en la figura de la página siguiente.



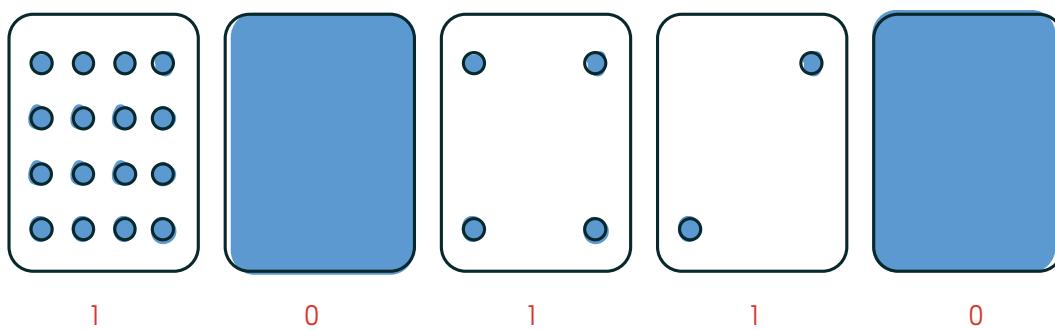
Representación binaria del número 9

Es interesante hacer notar a los estudiantes que  $9 = 2^3 + 2^0 = 8 + 1$ . Mirando la figura anterior, se puede llegar a la conclusión de que, en la representación de un número en binario, las posiciones donde hay un 1 indican las potencias de 2 que deben sumarse para obtener el número representado.

A continuación, pedimos a los grupos que con las cartas obtengan la representación binaria de los números 15 y 22. Las soluciones se muestran en las siguientes figuras.



Representación binaria del número 15



Representación binaria del número 22

## CIERRE

Mencionamos que el uso de los dígitos 0 y 1 es muy común y que se los llama *dígitos binarios*. Su importancia es tal que se inventó una palabra para describirlos tomando las dos primeras letras y la última letra de la expresión dígito binario en inglés: *binary digit*: bit. El sistema binario presentado en esta actividad permite representar todos los números naturales. Éste es el sistema que usan las computadoras internamente para representar información.

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# CARTAS Y NÚMEROS BINARIOS

¡Vamos a jugar a las cartas! Para empezar, recortá las cartas de la ficha.

1. ¿Qué cartas hay que dejar visibles y cuáles hay que dar vuelta para representar el número 11? ¿Y para el número 30?

---

---

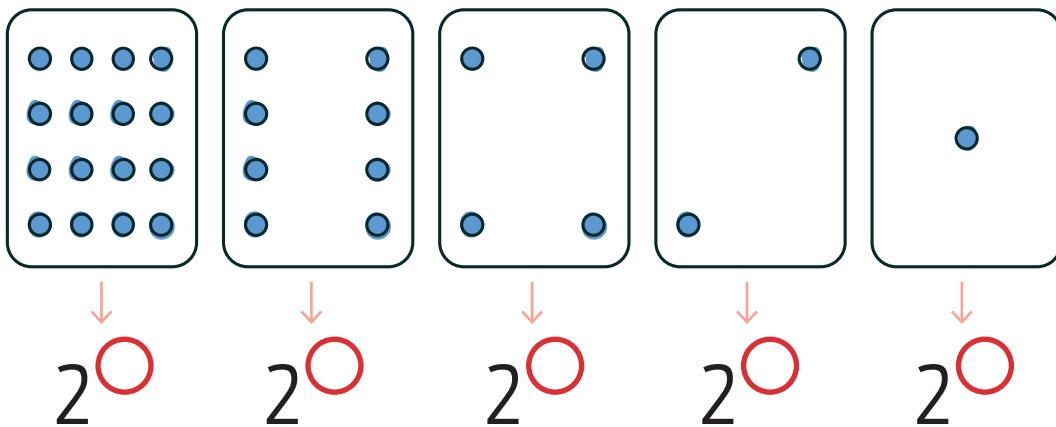


2. Ahora tenés que representar todos los números que puedas, comenzando por el 0, siguiendo por el 1, y continuando en orden hasta el más grande al que puedas llegar. ¿Cuál es el mayor número que alcanzaste?

---

---

3. Expresá la cantidad de puntos de cada carta como una potencia de 2. ¿Qué relación hay entre el exponente y la posición que ocupa cada carta en la secuencia, si la leemos de derecha a izquierda?



4. Con la ayuda de las cartas, escribí la representación binaria de los números 15 y 22.

15: \_\_\_\_\_

22: \_\_\_\_\_

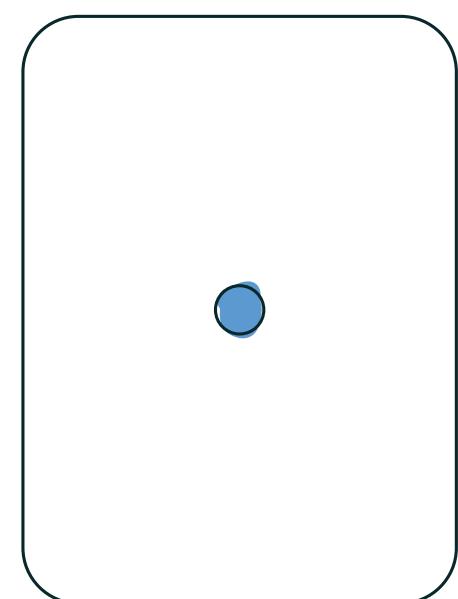
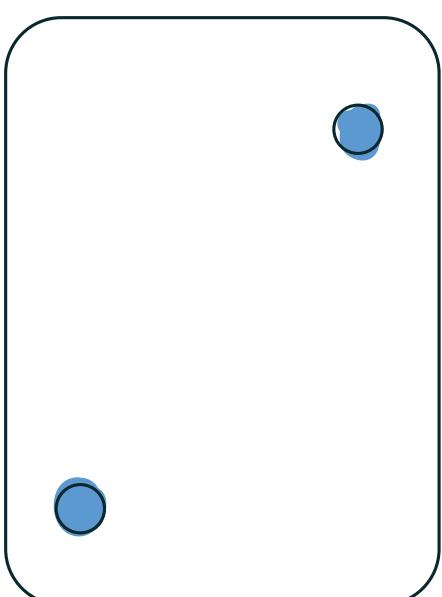
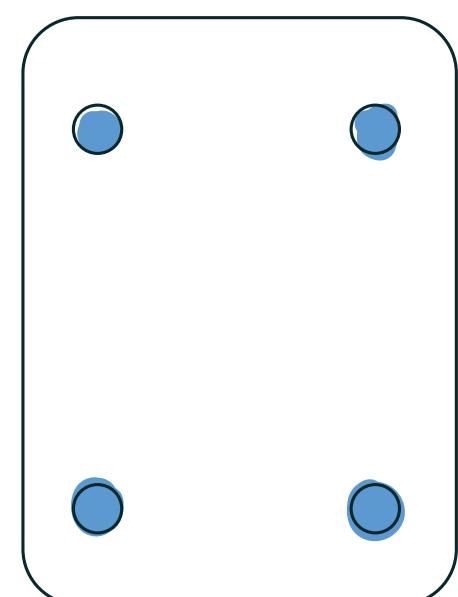
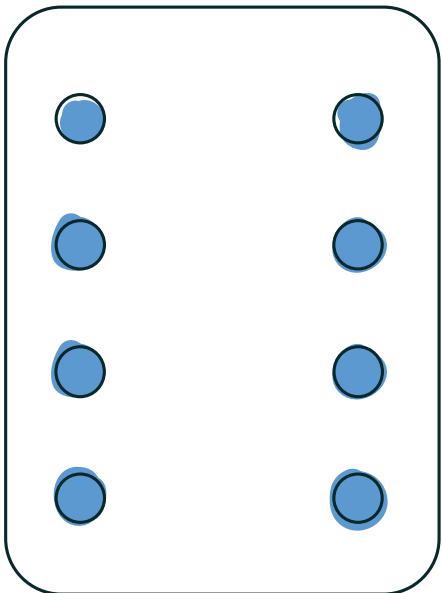
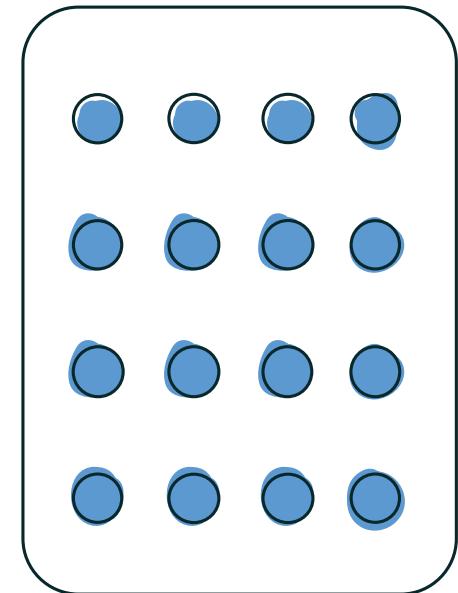
## SISTEMA BINARIO

El uso de los dígitos 0 y 1 es muy común y se los llama *dígitos binarios*. Su importancia es tal que se inventó una palabra para describirlos tomando las dos primeras letras y la última letra de la expresión *dígito binario* en inglés, *binary digit*: bit.

NOMBRE Y APELLIDO:

CURSO:

FECHA:



## Actividad 3

### Equis raya



#### OBJETIVOS

- Representar información textual con un sistema binario.
- Reconocer y corregir ambigüedades de un sistema de representación.

#### MATERIALES

- Cartulina grande con la codificación X-RAYA original
- Cartulinas grande con la codificación desambiguada de X-RAYA
- Ficha para estudiantes

#### DESARROLLO

Para dar comienzo a la actividad, presentamos la codificación de letras X-RAYA, que usa combinaciones de los símbolos **X** y **-**, según se muestra en la siguiente tabla.

X-RAYA					
<b>A: X</b>	<b>F: XX-</b>	<b>K: X-XX</b>	<b>O: X----</b>	<b>T: X-X-X</b>	<b>Y: XX-X-</b>
<b>B: X-</b>	<b>G: XXX</b>	<b>L: XX--</b>	<b>P: X---X</b>	<b>U: X-XX-</b>	<b>Z: XX-XX</b>
<b>C: XX</b>	<b>H: X---</b>	<b>M: XX-X</b>	<b>Q: X--X-</b>	<b>V: X-XXX</b>	
<b>D: X--</b>	<b>I: X--X</b>	<b>N: XXX-</b>	<b>R: X--XX</b>	<b>W: XX---</b>	
<b>E: X-X</b>	<b>J: X-X-</b>	<b>Ñ: XXXX</b>	<b>S: X-X--</b>	<b>X: XX--X</b>	

Codificación del abecedario X-RAYA

Preguntamos a la clase: “¿Cómo sería la codificación de la palabra SOY?”. Probablemente surja como respuesta **X-X--X---XX-X-**, lo cual es correcto porque de acuerdo a X-RAYA la S se codifica con la secuencia **X-X--**, la O con **X----** y la Y con **XX-X-**. Repartimos la ficha a los estudiantes y les pedimos que, en forma individual, codifiquen las palabras *POTUS* y *RUSO*. Deberían llegar a construir las secuencias que se muestran a continuación.

**X---XX----X-X-XX-XX-X-X--**      **X--XXX-XX-X-X--X----**

POTUS

RUSO

Una vez completada esta primera consigna, planteamos el problema inverso, y les pedimos que identifiquen a qué palabra del castellano corresponde la secuencia **XX-X-XX**. En este punto pueden surgir varias respuestas, entre ellas las palabras *FEA* y *ATA*. *FEA* es el resultado de agrupar la secuencia como **(XX-)(X-X)(X)** y *ATA* el de agruparla como **(X)(X-X-X)(X)**. En caso de que todos los estudiantes lleguen a la misma palabra (por ejemplo, *FEA*), les pedimos que codifiquen la otra (*ATA*) usando X-RAYA. Es importante que noten que ambas palabras se codifican con la misma secuencia en X-RAYA.

A continuación les preguntamos: “¿Cómo se podría hacer para que, dada una secuencia de X-RAYA, no exista posibilidad de ambigüedad al decodificarla?”. Con nuestra ayuda, se espera que surja la idea de asignar una cantidad fija de símbolos para representar cada una de las letras. Dado que 5 símbolos son suficientes

para codificar todas las letras, les pedimos que propongan una modificación a X-RAYA de forma tal que la codificación de cada letra tenga longitud 5. Analizamos las propuestas de los estudiantes para asegurarnos de que cada letra esté representada con una secuencia de longitud 5 y que no existan dos letras que se codifiquen con la misma secuencia. Esta propiedad alcanza para garantizar que la codificación no sea ambigua.

A continuación, hacemos una puesta en común proponiendo completar la codificación de cada letra de X-RAYA agregando rayas a la izquierda de forma tal que todas queden de longitud 5, tal como se muestra en la siguiente tabla.

X-RAYA DESAMBIGUADA					
A: ----X	F: --XX-	K: -X-XX	O: X----	T: X-X-X	Y: XX-X-
B: ---X-	G: --XXX	L: -XX--	P: X---X	U: X-XX-	Z: XX-XX
C: ---XX	H: -X---	M: -XX-X	Q: X--X-	V: X-XXX	
D: --X--	I: -X--X	N: -XXX-	R: X--XX	W: XX---	
E: --X-X	J: -X-X-	Ñ: -XXXX	S: X-X--	X: XX--X	

Modificación de X-RAYA para evitar ambigüedades

Les pedimos a los estudiantes que codifiquen las palabras *FEA* y *ATA*, para ilustrar que la ambigüedad ha desaparecido. *FEA* se codifica como **--XX---X-X----X** y *ATA* como **----XX-X-X----X**; como puede observarse, son dos secuencias distintas.

### CIERRE

Reflexionamos con toda la clase acerca del hecho de que es posible codificar texto usando solo dos símbolos. Podemos preguntar a la clase qué diferencia hubiera existido si, en lugar de los símbolos X y -, se hubieran usado 0 y 1, para llegar a la conclusión de que la elección de los símbolos es arbitraria. Es decir, alcanza con tener dos símbolos diferentes para hacer la codificación.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# EQUIS RAYA

Todos los días usamos textos en nuestros dispositivos digitales. Enviamos mensajes de texto, hacemos comentarios en las redes sociales, leemos páginas web y muchas cosas más. En esta actividad vamos a trabajar sobre los principios de la codificación de texto usados por las computadoras.



- 1.** Mirá la codificación de letras X-RAYA y a continuación representá las palabras *SOY, POTUS, RUSO, ESCURRIDIZO* y *ALBAÑIL*.

X-RAYA					
A: X	F: XX-	K: X-XX	O: X----	T: X-X-X	Y: XX-X-
B: X-	G: XXX	L: XX--	P: X---X	U: X-XX-	Z: XX-XX
C: XX	H: X---	M: XX-X	Q: X--X-	V: X-XXX	
D: X--	I: X--X	N: XXX-	R: X--XX	W: XX---	
E: X-X	J: X-X-	Ñ: XXXX	S: X-X--	X: XX--X	

SOY:

POTUS:

RUSO:

ESCURRIDIZO:

ALBAÑIL:

- 2.** Ahora codificá las palabras *ATA* y *FEA*. ¿Notás algún problema?

ATA:

FEA:

NOMBRE Y APELLIDO:

CURSO:

FECHA:

3. Proponé una sistema de codificación usando los símbolos **X** y **-**, en el que una secuencia de **X** y **-** tenga una y sólo una interpretación posible.

X-RAYA SIN AMBIGÜEDADES					
A:	F:	K:	O:	T:	Y:
B:	G:	L:	P:	U:	Z:
C:	H:	M:	Q:	V:	
D:	I:	N:	R:	W:	
E:	J:	Ñ:	S:	X:	

4. ¿Cómo se codifican ahora las palabras *ATA* y *FEA*? ¿Resolviste la ambigüedad?

ATA:

FEA:

#### SISTEMAS DE CODIFICACIÓN DE SÍMBOLOS

A lo largo de la historia, las computadoras utilizaron diferentes maneras para codificar textos en binario.

Uno de los sistemas más difundidos es el sistema ASCII, que utiliza 7 dígitos binarios para representar cada símbolo. Esto permite codificar todas las letras del idioma inglés en mayúsculas y en minúsculas, los números del 0 al 9 y una variedad de símbolos especiales. Sin embargo, no permite representar otros alfabetos o letras del alfabeto castellano como la *ñ*. Para esto se extendió el sistema ASCII a 8 dígitos binarios y se crearon otras normas. Actualmente el sistema más usado es Unicode, que utiliza hasta 32 dígitos binarios.





## Secuencia Didáctica 2

# REPRESENTACIÓN DE IMÁGENES DIGITALES

La presente secuencia didáctica parte del interrogante de cómo se utiliza el sistema binario para representar imágenes digitales. La primera actividad de la secuencia permite representar imágenes en blanco y negro con valores binarios utilizando mapas de bits. En la segunda actividad se incorpora la propiedad del color usando la codificación estándar RGB, sigla que proviene del inglés *Red, Green y Blue* (rojo, verde y azul).

.....  
**OBJETIVO**

- Comprender formas de codificación de imágenes digitales.
- .....

## Actividad 1

### Imágenes en blanco y negro<sup>1</sup>

 DE A DOS

#### OBJETIVOS

- Representar imágenes usando mapas de bits.
- Interpretar un mapa de bits para reconstruir una imagen.

#### MATERIALES

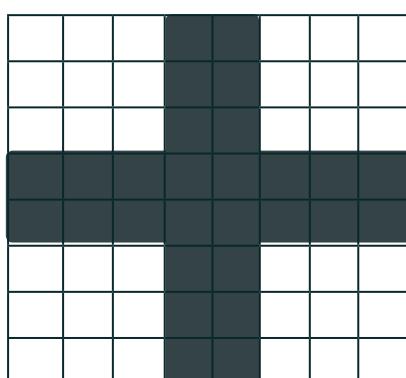
 Pizarrón

 Tizas

 Ficha para estudiantes

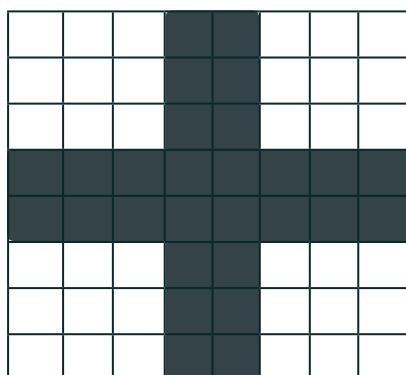
#### DESARROLLO

Comenzamos la actividad dibujando en el pizarrón una cuadrícula de 8 x 8, con una cruz en su interior, tal como se muestra en la figura.



Una cruz

Preguntamos a la clase: "¿Qué características tiene la imagen?". Guiamos la discusión para llegar a la conclusión de que la imagen de la cruz se encuentra enmarcada en una cuadrícula de 8 x 8, compuesta por cuadraditos cada uno de los cuales puede estar en dos estados diferentes: o bien pintado de blanco o bien de negro. Pedimos a la clase que proponga una forma de representar la imagen usando únicamente los símbolos 0 y 1 (o, lo que es lo mismo, usando bits). Es probable que los estudiantes propongan que el 0 represente los cuadraditos blancos y el 1 los negros. A partir de la codificación propuesta, les preguntamos cómo se codificaría la primera fila de la cuadrícula. La respuesta correcta es la secuencia **00011000**. Una vez que se haya comprendido el procedimiento, pedimos a la clase que dicte la codificación de cada una de las filas y, a medida que las dictan, las escribimos al lado de la imagen. De este modo, debería llegarse a la codificación que se muestra en la siguiente figura.



0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0

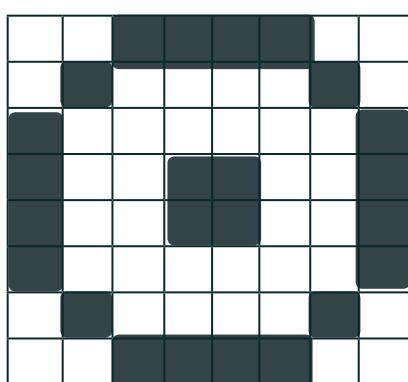
<sup>1</sup> Adaptación de la actividad "Bitmaps" de

cse4k12.org, disponible en <http://bit.ly/2lR2NAd>.

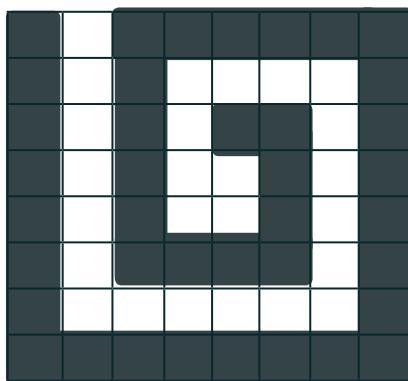
La cruz y su representación como mapa de bits.

A continuación, explicamos que la codificación usada se conoce como **mapa de bits**, y que se usa para representar imágenes digitales.

Repartimos la ficha de la actividad a los estudiantes y les pedimos que completen el mapa de bits para cada una de las figuras que allí se muestran. Se espera que lleguen a las soluciones que se muestran en la figura.



0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1
1	0	0	0	0	0	0	1
0	1	0	0	0	0	1	0
0	0	1	1	1	1	0	0

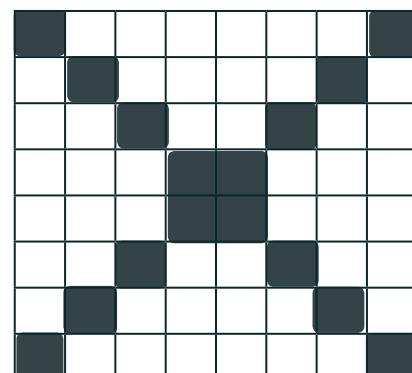


1	0	1	1	1	1	1	1
1	0	1	0	0	0	0	1
1	0	1	0	1	1	0	1
1	0	1	0	0	1	0	1
1	0	1	0	0	1	0	1
1	0	1	1	1	1	0	1
1	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1

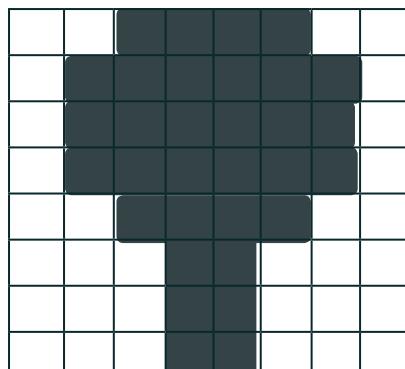
Dos dibujos y sus mapas de bits asociados

Luego, les solicitamos que completen la segunda actividad de la ficha, en la que se hace el proceso inverso: a partir de dos mapas de bits hay que obtener las imágenes representadas, que son las que muestra la figura.

1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	1	0
0	0	1	0	0	1	0	0	0
0	0	0	1	1	0	0	0	0
0	0	0	1	1	0	0	0	0
0	0	1	0	0	1	0	0	0
0	1	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	1



0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0



Mapas de bits y sus correspondientes imágenes

Finalmente, les pedimos que hagan sus propios dibujos en la ficha y luego escriban su correspondiente codificación como mapa de bits.

### CIERRE

Les explicamos a los estudiantes que cada una de las celdas de las cuadrículas con las que trabajaron se conoce como **píxel**, que es la unidad mínima en la composición de imágenes digitales. Les comentamos que lo que ven en las pantallas de sus computadoras y teléfonos inteligentes son conjuntos de muchos píxeles, uno al lado de otro, que forman las imágenes.

---

NOMBRE Y APELLIDO:

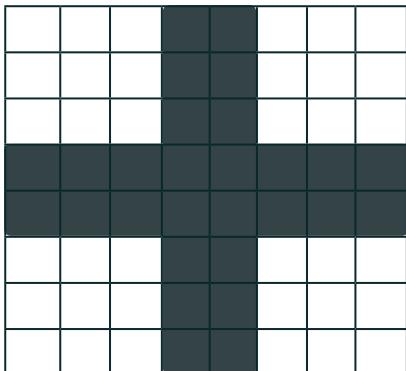
CURSO:

FECHA:

# IMÁGENES EN BLANCO Y NEGRO

Los mapas de bits son una forma de codificar imágenes en blanco y negro en la que se emplean números binarios. El 0 se utiliza para representar un cuadrado blanco en la imagen y el 1 se utiliza para representar un cuadrado negro. En esta actividad vas a traducir algunas imágenes al lenguaje de las computadoras y vas a interpretar ese lenguaje para poder reconstruir otras imágenes.

- 1.** Completá los mapas de bits de las siguientes imágenes.



---

---

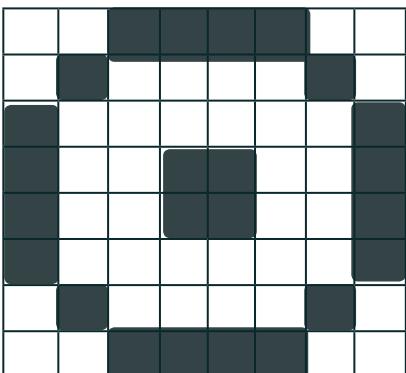
---

---

---

---

---



---

---

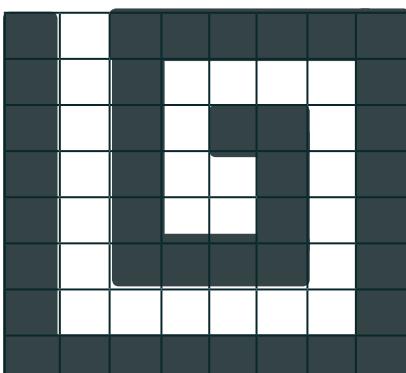
---

---

---

---

---



---

---

---

---

---

---

---



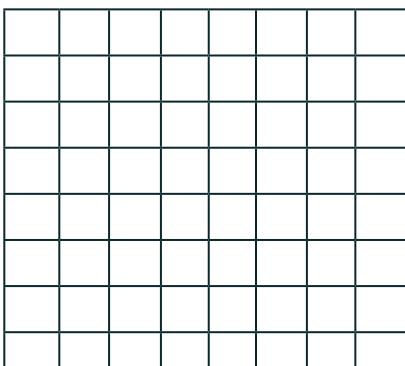
NOMBRE Y APELLIDO:

CURSO:

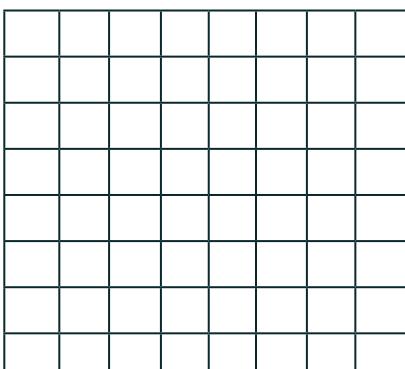
FECHA:

2. Seguí la representación binaria de las imágenes y dibujalas.

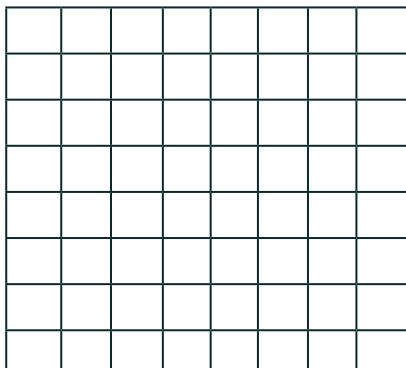
1	0	0	0	0	0	0	1
0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	0	1	0
1	0	0	0	0	0	0	1



0	0	1	1	1	1	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0



3. Ahora dibujá lo que quieras y escribí su representación como mapa de bits.



---

---

---

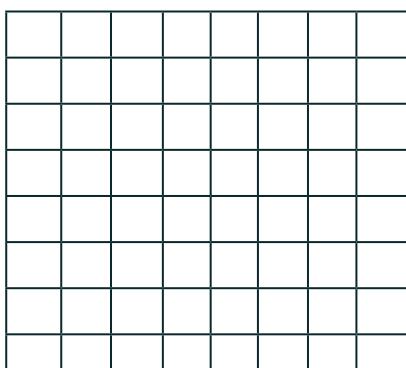
---

---

---

---

---



---

---

---

---

---

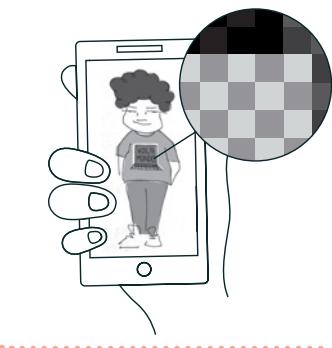
---

---

---

#### IMÁGENES DIGITALES

Cada imagen que ves en las pantallas de los dispositivos digitales está compuesta por puntos que se dibujan uno al lado del otro. En computación, a estos puntos los llamamos **píxeles**. El píxel es la unidad mínima de la composición de las imágenes digitales.



## Actividad 2

### Banderas de colores

 DE A DOS

#### OBJETIVOS

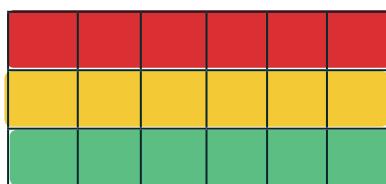
- Comprender que las imágenes en color poseen más información que las imágenes en blanco y negro.
- Conocer el modelo de color RGB usado en las imágenes digitales.

#### MATERIALES

-  Lámina con la bandera de Bolivia en una grilla de 6 x 3
-  Lámina con la codificación de colores RGB
-  Ficha para estudiantes

#### DESARROLLO

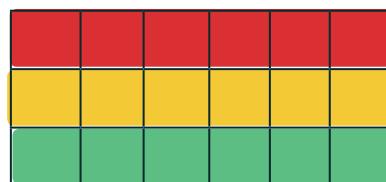
Comenzamos la actividad presentando la lámina de la bandera de Bolivia dentro de una grilla de 6 x 3, como se muestra en la figura. Si se prefiere, se la puede dibujar en el pizarrón del aula.



Bandera de Bolivia

Les preguntamos a los estudiantes cómo harían para codificar la bandera de Bolivia. Con lo que aprendieron hasta el momento no es posible codificar la imagen, porque hasta ahora representamos cada píxel usando solo dos valores: 0 y 1. Esto es insuficiente cuando se quiere codificar más de dos colores, como en este caso. Guiamos la discusión para llegar a la conclusión de que un bit por píxel es insuficiente si queremos representar muchos colores.

Preguntamos luego: “¿Se les ocurre algún modo de poder hacerlo?”. Una imagen en color tiene más información que una en blanco y negro y, por lo tanto, necesitamos más de un bit por píxel para poder codificarla. La cantidad de bits necesaria dependerá de la cantidad de colores que se quiera diferenciar. Si solo son tres, como es el caso de la bandera presentada, alcanza con codificar cada píxel con dos bits. Siguiendo el ejemplo, y usando solo dos bits, se podría tomar como convención que el **00** represente el rojo, el **01** el amarillo y el **10** el verde. En ese caso, la bandera boliviana podría representarse como muestra la figura.



00	00	00	00	00	00
01	01	01	01	01	01
10	10	10	10	10	10

Codificación de la bandera de Bolivia usando 2 bits

Sin embargo, las imágenes que vemos en las computadoras no tienen solo tres colores. Cada píxel en una pantalla se compone típicamente de tres luces minúsculas: una roja, una verde y otra azul. Al aumentar y disminuir la cantidad de luz que sale de cada uno de estos tres puntos se pueden hacer todos los colores. La codificación más usada se llama RGB –de las siglas de los colores en inglés *red, green* y *blue*– y permite diferenciar más de 16 millones de colores!

Cada píxel se representa con 24 bits y se interpreta agrupándolos de a 8 bits: cada grupo de 8 bits representa la intensidad de uno de los colores primarios aditivos. De esta manera, se contará con  $2^8 = 256$  variantes para representar la intensidad de cada color primario aditivo.

Vale la pena aclarar que, para no tener que manejar números tan grandes, en general los bits se agrupan de a 8 y a esos grupos de 8 bits se los llama bytes. En este punto presentamos la siguiente tabla, que tiene la codificación RGB de los colores que se usan en esta actividad. En este caso, en lugar de usar la representación binaria, se muestran los números de 0 a 255 en su representación decimal para facilitar su lectura y comprensión.

	R	G	B
ROJO	255	0	0
AZUL	0	0	255
AMARILLO	255	255	0
VERDE	0	255	0
CELESTE	116	169	218
BLANCO	255	255	255

Codificación RGB de algunos colores

Ahora, mostramos la bandera de Bolivia nuevamente y les pedimos a los estudiantes que digan cómo es su codificación usando RGB. Deberían llegar a la siguiente respuesta.

(255,0,0)	(255,0,0)	(255,0,0)	(255,0,0)	(255,0,0)	(255,0,0)
(255,255,0)	(255,255,0)	(255,255,0)	(255,255,0)	(255,255,0)	(255,255,0)
(0,255,0)	(0,255,0)	(0,255,0)	(0,255,0)	(0,255,0)	(0,255,0)

Codificación RGB de la bandera de Bolivia

Repartimos la ficha a los estudiantes y les pedimos que, individualmente, completen la codificación RGB de las banderas de Argentina, Paraguay y México. Las soluciones se muestran a continuación.

(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)
(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)
(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)

(255,0,0)	(255,0,0)	(255,0,0)	(255,0,0)	(255,0,0)	(255,0,0)
(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)
(0,0,255)	(0,0,255)	(0,0,255)	(0,0,255)	(0,0,255)	(0,0,255)

(0,255,0)	(0,255,0)	(255,255,255)	(255,255,255)	(255,0,0)	(255,0,0)
(0,255,0)	(0,255,0)	(255,255,255)	(255,255,255)	(255,0,0)	(255,0,0)
(0,255,0)	(0,255,0)	(255,255,255)	(255,255,255)	(255,0,0)	(255,0,0)

Codificación RGB de las banderas de Argentina, Paraguay y México

## CIERRE

Reflexionamos sobre la diferencia de espacio que se requiere para codificar imágenes en color en relación a las imágenes en blanco y negro. Al usar dos colores alcanzaba con un bit para representar la información de un píxel. Para codificar la información de color usando RGB se necesitan 24 bits por pixel. Por lo tanto, el espacio requerido para codificar imágenes de idénticas dimensiones es 24 veces mayor. En general, mientras mayor sea la cantidad de información que se quiera representar, mayor es la cantidad de espacio que se necesita para hacerlo.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# BANDERAS DE COLORES

¿Alguna vez mezclaste diferentes colores de pintura para hacer otros nuevos? Es común utilizar rojo, amarillo y azul como tres colores **primarios** que se pueden mezclar para producir muchos más colores. Rojo y azul dan violeta, rojo y amarillo dan naranja, por ejemplo.



Las pantallas también se basan en la mezcla de tres colores, pero necesitan un conjunto diferente de colores primarios porque comienzan con una pantalla negra y van añadiendo color a ella a través de luces. Estas mezclas de colores se denominan **aditivas**. Cada píxel en una pantalla se compone típicamente de tres luces minúsculas: una roja, una verde y otra azul. Al aumentar y disminuir la cantidad de luz de cada uno de estas tres, se pueden hacer todos los colores diferentes. Esta mezcla de colores se conoce como RGB por sus siglas de las palabras en inglés: *Red* (rojo), *Green* (verde) y *Blue* (azul). Acá te presentamos la codificación de algunos colores que van a servirte para resolver esta actividad.

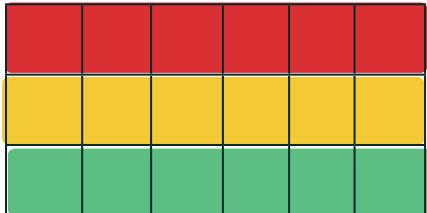
	R	G	B
ROJO	255	0	0
AZUL	0	0	255
AMARILLO	255	255	0
VERDE	0	255	0
CELESTE	116	169	218
BLANCO	255	255	255

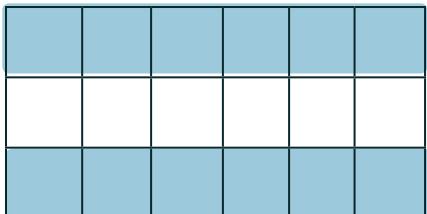
NOMBRE Y APELLIDO:

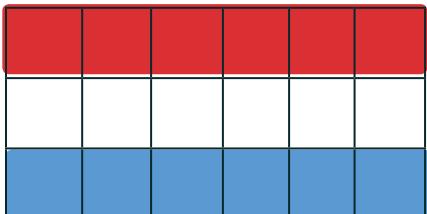
CURSO:

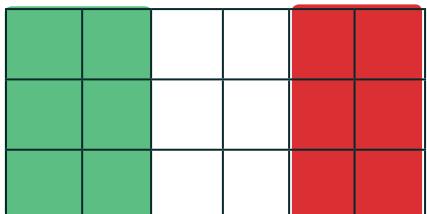
FECHA:

Por ejemplo, rojo se indica (255,0,0) y azul, (0,0,255). Ahora tenés que dar la codificación RGB de las banderas de Bolivia, Argentina, Paraguay y México.






¿SABÍAS QUE...

...la codificación por mapa de bits es la que usan los archivos con extensión  **bmp**?



## Secuencia Didáctica 3

# COMPRESIÓN DE INFORMACIÓN

La cantidad de información que actualmente manipulan los sistemas de computación suele ser muy grande. Por ejemplo, es habitual que las fotos sacadas con cámaras de teléfonos inteligentes sean de alrededor de 15 megapíxeles. Si una foto se representara usando el estándar RGB, en el que cada píxel se codifica con 24 bits, cada foto ocuparía aproximadamente 45 MB. Esto se transforma en un problema al querer guardarla en un disco o transmitirla a través de una red. La compresión de datos aborda el problema de la reducción del volumen de datos necesarios para representar información.

Esta secuencia didáctica está compuesta de dos actividades. La primera presenta un método de compresión sin pérdida de información; es decir, que una vez comprimida se puede recuperar la imagen original. En la segunda se propone experimentar con distintos formatos de archivos de imágenes. Algunos de ellos comprimen perdiendo información de la imagen original y otros no.

### OBJETIVOS

- Comprender la motivación y los efectos de la compresión de datos.
- Presentar métodos de compresión con y sin pérdida de información.

## Actividad 1

### Encogemos imágenes

 DE A DOS

#### OBJETIVOS

- Presentar un método de compresión sin pérdida de información.
- Observar que el espacio necesario para representar información puede reducirse en gran medida.

#### MATERIALES

 Una lámina con la bandera de Argentina y su codificación RGB.

 Ficha para estudiantes

#### DESARROLLO

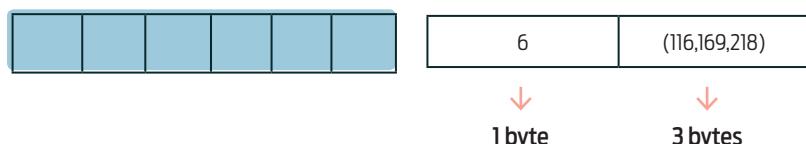
Para comenzar la actividad, mostramos la lámina de la bandera de Argentina y su codificación RGB en dos grillas de  $6 \times 3$  como las de la figura. Alternativamente, se pueden copiar en el pizarrón.


(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)
(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)	(255,255,255)
(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)	(116,169,218)

Bandera argentina y su codificación RGB

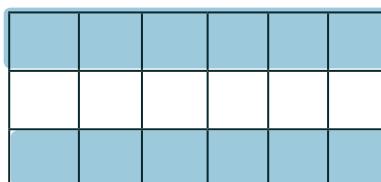
Al usar la codificación planteada, y teniendo en cuenta que hacen falta 3 bytes para codificar el color de cada píxel, la bandera completa, compuesta por 18 píxeles, se codifica usando  $3 \times 18 = 54$  bytes. Le planteamos a la clase: “¿Qué pasaría si quisieramos hacer una bandera más grande, usando por ejemplo  $900 \times 600 = 540.000$  píxeles?”. Se espera que respondan que en ese caso, en lugar de necesitar 54 bytes, harían falta  $540.000 \times 3 = 1.620.000$  bytes  $\approx 1,62$  Mb. “Para ciertos usos frecuentes, como transmitirla a través de una red, esta cantidad se transforma en un problema”.

Una vez comprendido el problema, preguntamos: “¿Se les ocurre alguna otra forma de codificar la imagen de forma tal que ocupe menos espacio?”. En caso de que la clase no consiga aproximarse a una representación de menor volumen, le hacemos notar que hay mucha información que se repite. Por ejemplo, la primera fila de la bandera está compuesta en su totalidad por píxeles celestes. Guiamos la discusión de forma tal de arribar a la conclusión de que la información repetida podría codificarse indicando primero cuántas veces se repite el valor de un píxel y, seguidamente, la representación RGB del color que se repite. Por ejemplo, la primera fila se puede codificar como muestra la figura.



Codificación de una fila usando compresión RLE

De esta forma, se puede codificar la primera fila usando 4 bytes en lugar 18: el primero para indicar las repeticiones y los tres restantes para el código RGB del color. Extendiendo esta idea, la bandera completa se puede representar usando  $3 \times 4 = 12$  bytes en lugar de los 54 usados en la codificación presentada en primera instancia.



6	(116,169,218)
6	(255,255,255)
6	(116,169,218)

Representación de la bandera argentina usando 12 bytes

A esta altura, se puede comentar que la magnitud de la compresión puede ser mucho mayor que la mostrada en el ejemplo. Por ejemplo, si la bandera tuviese 100 píxeles por fila, de todas formas podría representarse usando 12 bytes, en lugar de los 900 bytes que harían falta si no se usase compresión.

100	(116,169,218)
100	(116,169,218)
100	(116,169,218)

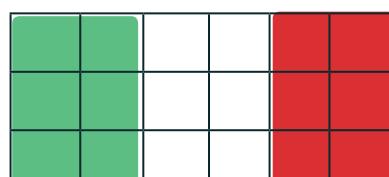
Representación de la bandera argentina de  $100 \times 3 = 300$  píxeles

Comentamos que este sistema de compresión se llama RLE –de las siglas del inglés, *Run-Length Encoding*–. A continuación, repartimos la ficha de la actividad a los estudiantes y les pedimos que completen la codificación de cada una de las banderas usando RLE y comparan la cantidad de espacio necesario en cada caso, usando compresión y sin usarla. Las soluciones se dan a continuación.



6	(255,0,0)
6	(255,255,0)
6	(0,0,255)

Espacio usado: 12 bytes



2	(0,255,0)	2	(255,255,255)	2	(255,0,0)
2	(0,255,0)	2	(255,255,255)	2	(255,0,0)
2	(0,255,0)	2	(255,255,255)	2	(255,0,0)

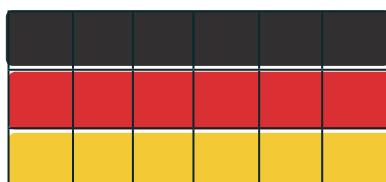
Espacio usado: 36 bytes

Banderas de Bolivia y México y su codificación RLE

Una vez que los estudiantes hayan completado el ejercicio, hacemos notar que la compresión fue mayor en el caso de las banderas de Argentina y Bolivia que en la de México. En este punto podemos preguntar la clase: “¿A qué se debe esta diferencia?”. RLE funciona muy bien para imágenes en las que muchos píxeles consecutivos tienen el mismo color, lo que sucede en mayor medida en las banderas de Argentina y Bolivia que en la de México. En general, esta compresión alcanza buenos grados de compresión en imágenes que tienen esta característica, como logotipos o íconos, en las que suele haber áreas de color plano.

Finalmente, les pedimos que decodifiquen la bandera del último punto de la ficha a partir de la representación obtenida al aplicar RLE. Al hacerlo, encontrarán la bandera de Alemania.

6	(0,0,0)
6	(255,0,0)
6	(255,255,0)



Bandera de Alemania y su codificación usando RLE

### CIERRE

Les comentamos a los estudiantes que RLE es uno entre los muchos métodos de compresión que son usados por las computadoras. Así como RLE funciona bien sobre imágenes en las que muchos píxeles consecutivos son del mismo color, no es tan efectiva en otros contextos, como al trabajar con texto o video. La compresión de información es un área activa de investigación en Ciencias de la Computación, y existen diversas técnicas que abordan el problema teniendo en cuenta las características del tipo de información que se quiere comprimir.

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# ENCOGEMOS IMÁGENES

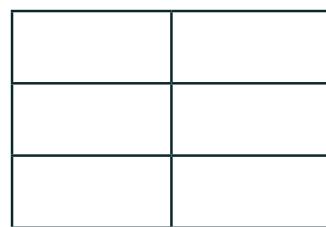
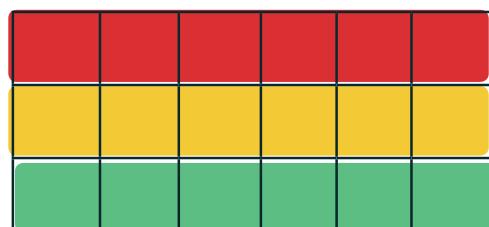
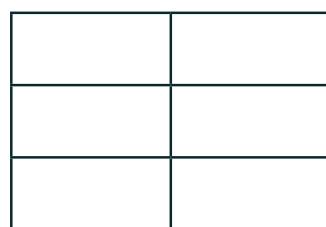
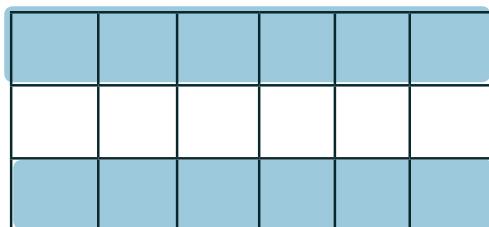
¿Alguna vez te pasó que quisiste mandar un archivo por mail y no pudiste porque era demasiado grande? ¿Sabés cómo hacen las computadoras cuando tienen que almacenar o transmitir enormes volúmenes de información? En esta actividad vas a usar un método que permite reducir el espacio requerido para representar imágenes digitales.



Acá te presentamos los códigos RGB de los colores que vas a necesitar para resolver la actividad. Recordá que, por ejemplo, para indicar el código del rojo se escribe: (255,0,0).

	R	G	B
ROJO	255	0	0
BLANCO	255	255	255
AMARILLO	255	255	0
VERDE	0	255	0
CELESTE	116	169	218
NEGRO	0	0	0

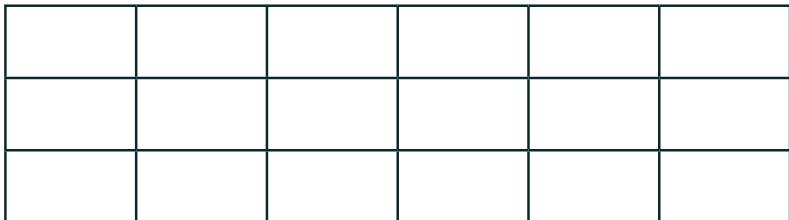
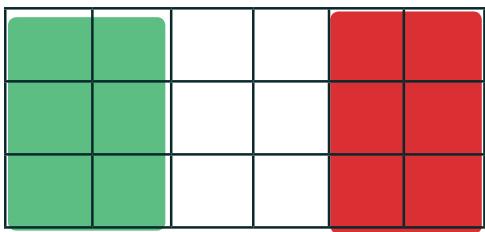
1. ¿Cómo sería la codificación RLE de las banderas de Argentina, Bolivia y México?



NOMBRE Y APELLIDO:

CURSO:

FECHA:



- 2.** ¿Cuántos bytes necesitaste para codificar cada una? ¿Cuántos hubieran hecho falta si no hubieses usado RLE?

---

---

- 3.** ¿Se te ocurre por qué no todas las codificaciones necesitaron la misma cantidad de espacio?

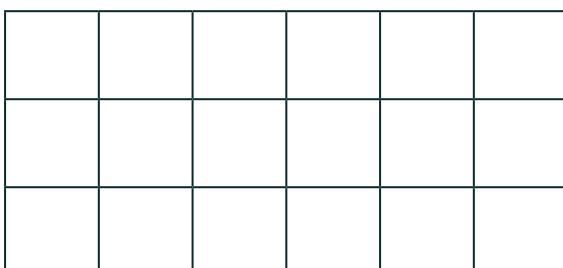
---

---

---

- 4.** ¡Ahora te toca descomprimir una imagen! ¿Qué bandera se formó?

6	(0,0,0)
6	(255,0,0)
6	(255, 255, 0)



#### COMPRESIÓN RLE

¿Sabías que RLE pertenece a una familia de métodos de compresión que no pierden información? Todos ellos permiten recuperar la información original tal cual era antes de comprimirla. Los archivos .zip y .rar también se generan usando algoritmos de esta familia.



## Actividad 2

### Experimentamos con imágenes digitales

2 personas DE A DOS

#### OBJETIVOS

- Identificar contextos en los que conviene perder información.
- Conocer distintos formatos de archivos para representar imágenes.

#### MATERIALES

 Computadoras

 Editor de imágenes

#### DESARROLLO

Comenzamos la actividad preguntando a la clase: "¿Cómo hacen para enviar fotos sacadas con cámaras digitales a amigos o familiares?". Es probable que surjan respuestas tales como que las envían usando algún servicio de mensajería instalado en sus teléfonos inteligentes. Esto da pie para que preguntemos: "¿Notan alguna diferencia entre las fotos originales y las enviadas?". Esos programas no suelen enviar las fotos originales porque ocupan demasiado espacio y eso dificulta su envío a través de una red. Es importante hacer notar que, en muchos casos, no tiene mayor importancia que una imagen conserve la calidad original. Por ejemplo, al mandar una *selfie* a un familiar, es probable que sea más importante mandar rápidamente la imagen que hacerlo con mucha calidad.

Continuamos preguntando a los estudiantes qué formatos de imágenes conocen y si saben si en estos se usa o no compresión. Se espera que mencionen varios formatos conocidos, como JPEG, GIF, PNG o BMP. JPEG –sigla del inglés *Joint Photographic Experts Group*– es un formato muy frecuente que utiliza un algoritmo de compresión con pérdida de información para la codificación de las imágenes. Esto significa que los archivos de imágenes se reducen en tamaño, pero no es posible a partir de ellos reconstruir toda la información de la imagen original. En otros formatos, como BMP –sigla del inglés *Bit Mapped Picture*– y PNG –sigla del inglés *Portable Network Graphics*–, las imágenes generalmente ocupan más espacio que en JPEG debido a que codifican la información de la imagen con algoritmos de compresión sin pérdida de información.

A continuación, les pedimos a los estudiantes que se ubiquen de a pares y abran el programa de edición de imágenes que tengan instalado en sus computadoras. Si no tienen instalado un programa para tal propósito, pueden usar miniPaint, disponible para uso gratuito en <http://bit.ly/2kmG7aJ>. En lo que sigue, el desarrollo de la actividad se ilustra usando miniPaint, pero las funcionalidades usadas están disponibles en la mayoría de las aplicaciones de edición de imágenes digitales.

Para que la actividad sea efectiva, es importante partir de imágenes con alta resolución. Hay que tener en cuenta que el efecto de la compresión podrá observarse más fácilmente si las imágenes usadas cuentan con distintos tonos de un mismo color.

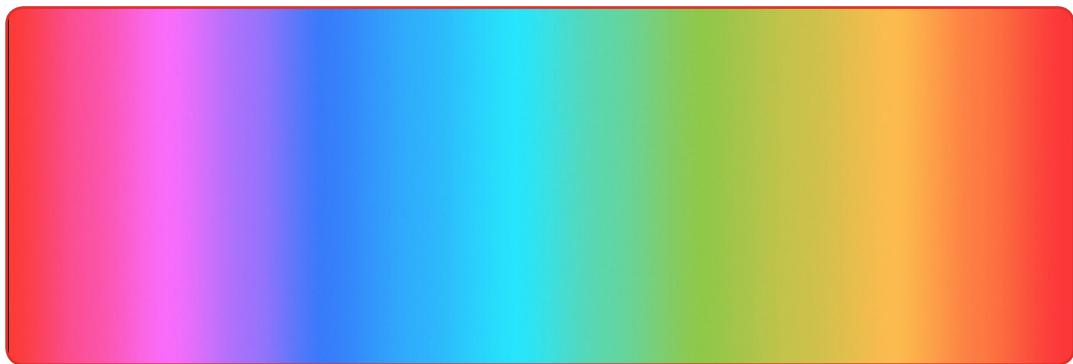


Imagen de referencia para la actividad

En primer lugar, los estudiantes deben abrir con el programa de edición la imagen seleccionada.

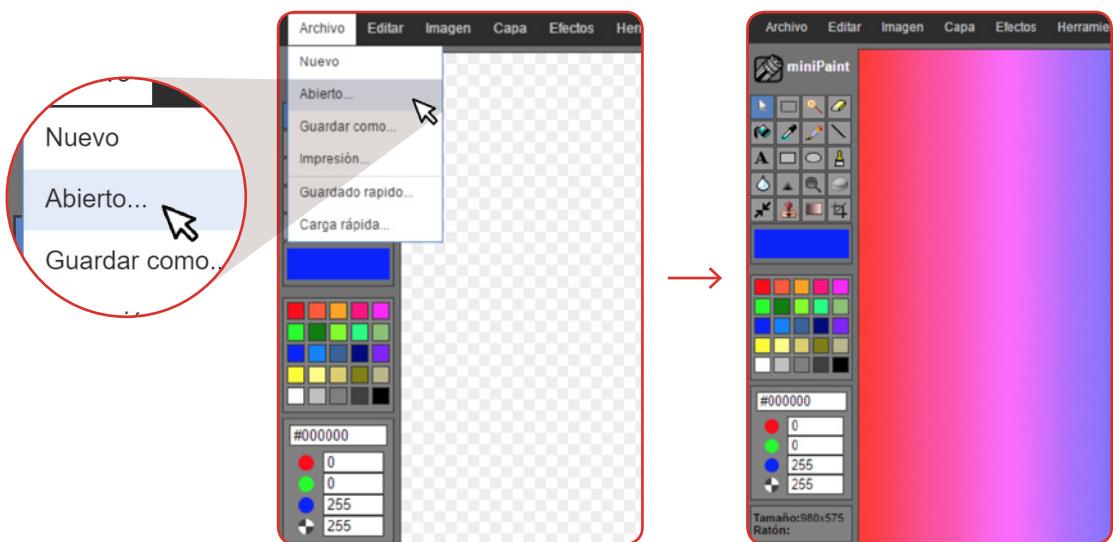
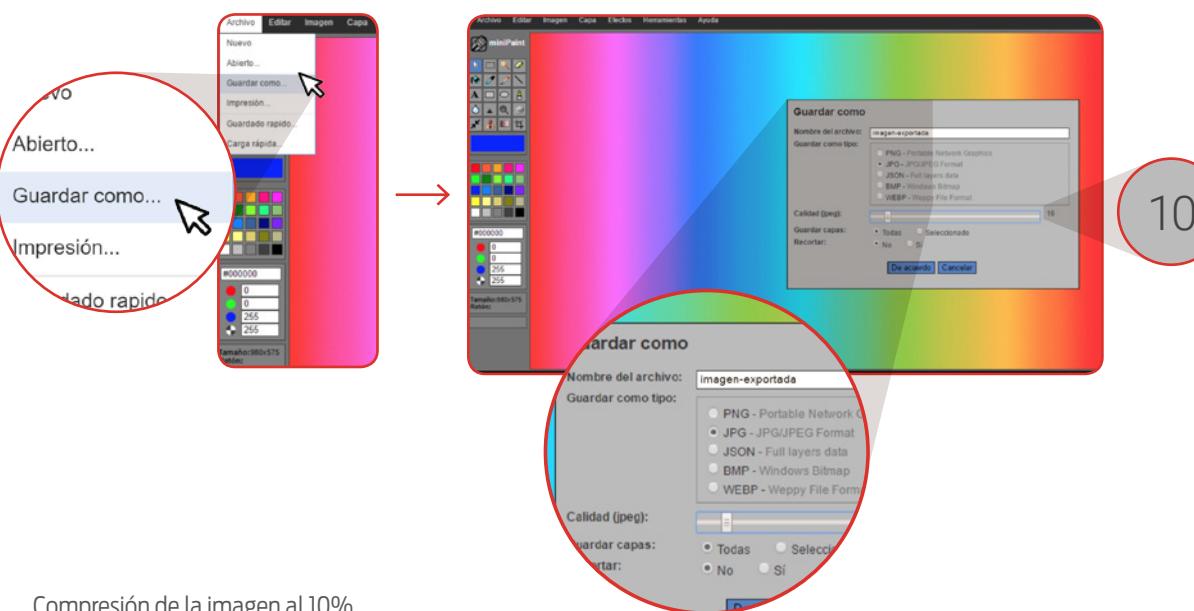


Imagen seleccionada en el programa de edición

A continuación, les pedimos que exporten la imagen como .JPG ajustando la calidad al 10%, como se muestra en la figura.



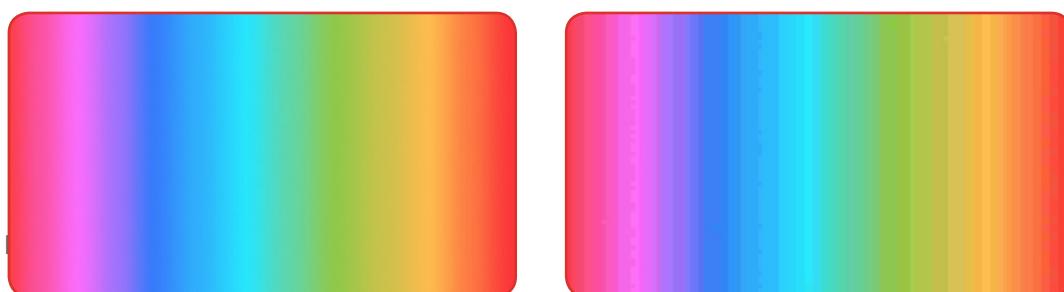
Es importante que los estudiantes observen la diferencia de tamaño entre la imagen original y la imagen exportada. Para hacerlo, deberán usar el explorador de archivos de la computadora. En el caso del ejemplo, el tamaño de la imagen comprimida es considerablemente menor que el de la imagen original.

<input checked="" type="checkbox"/> <b>imagen-exportada</b>	17/10/2019 14:32	Archivo JPG	80 KB
<input type="checkbox"/> <b>imagen-original</b>	17/10/2019 14:32	Archivo PNG	424 KB

80 KB  
424 KB

Diferencia de tamaño entre el archivo original y el exportado

También es importante que observen que la imagen se ha degradado notablemente.



En este caso, las diferencias entre ambas imágenes son muy notables, tanto en términos de tamaño como de calidad. La compresión usada aplana los colores, con lo que se pierde la posibilidad de distinguir toda la gama original.

Finalmente, les pedimos a los alumnos que experimenten exportando con distintos grados de compresión y distintos tipos de archivos, de manera que puedan observar las diferencias en los resultados. En cada caso, los estudiantes deben no solo comprobar el tamaño que ocupa cada archivo exportado, sino también volver a abrir los archivos para observar la diferencia en la definición de las imágenes. Esto permitirá verificar que algunos de estos formatos utilizan algoritmos de compresión que logran disminuir notablemente el tamaño de una imagen pero pierden calidad en el proceso.

### CIERRE

A modo de cierre, proponemos a los estudiantes que tomen una foto con la cámara de sus teléfonos inteligentes y se la envíen a un compañero mediante alguna aplicación de mensajería que usen habitualmente. Luego, les pedimos que comparan la foto original y la enviada en términos de tamaño y calidad. De esta forma, podrán corroborar experimentalmente que las aplicaciones de mensajería suelen comprimir las imágenes para enviarlas.

---

# 06

# LA COMPUTADORA

## SECUENCIA DIDÁCTICA 1

DEL CONCEPTO AL ELEMENTO

¿Qué son las computadoras?

Una computadora a cielo abierto

---

## SECUENCIA DIDÁCTICA 2

LA UNIDAD CENTRAL

DE PROCESAMIENTO Y LA MEMORIA

*Zoom in* a la unidad central

de procesamiento

¿Cuánto cabe en la memoria?

La caché

---

## SECUENCIA DIDÁCTICA 3

LA COMPUTADORA EN ACCIÓN

Bajamos hasta el lenguaje de máquina

Ejecución de programas

Los programas son descripciones sin ambigüedades del comportamiento que se espera que pueda tener una máquina. En rigor, se trata de piezas intangibles cuya existencia adquiere sentido solo porque hay una contraparte capaz de ejecutarlas: el **hardware**. Con este término hacemos referencia al conjunto de componentes físicos que forma parte de una computadora.

El universo del **hardware** abre la posibilidad para que nos hagamos muchas preguntas: ¿Qué es una computadora? ¿Cómo son sus componentes, qué funciones cumplen y cómo interactúan? ¿Qué cualidades definen el desempeño de un sistema de computación? ¿Cómo es que el **hardware** ejecuta efectivamente los programas que escribimos? En este capítulo se proponen actividades que buscan contribuir a que los estudiantes esbozen respuestas a estas y otras preguntas sobre la computadora.



## Secuencia Didáctica 1

# DEL CONCEPTO AL ELEMENTO

En esta secuencia didáctica se destruye la noción de computadora, en general asociada a las computadoras de escritorio y a las portátiles. Partimos de una pregunta fundamental: ¿Qué es una computadora? Para aproximarnos a una respuesta, se introduce el modelo arquitectónico de computadoras presentado por el matemático John von Neumann en la década de 1940. Luego, se propone una actividad para desensamblar una computadora que permite, por un lado, que los estudiantes conozcan y manipulen sus componentes tangibles, y, por otro, que reconozcan que los elementos que encuentran forman un sistema que es descrito con mucha precisión por el modelo conceptual de von Neumann, aun cuando este fue formulado muchísimos años antes de la fabricación de la computadora desensamblada.

.....  
**OBJETIVOS**

- Presentar el modelo de von Neumann.
  - Reconocer que las computadoras actuales son descritas por este modelo.
- .....

## Actividad 1

### ¿Qué son las computadoras?



INDIVIDUAL

#### OBJETIVOS

- Deconstruir la noción habitual de computadora.
- Presentar la arquitectura de von Neumann.
- Reconocer que las computadoras actuales se ajustan al modelo de von Neumann.

#### MATERIALES



Computadora



Conexión a Internet



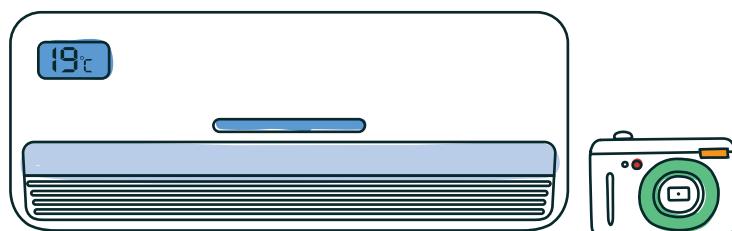
Ficha para estudiantes

#### DESARROLLO

Los objetivos de esta actividad son los siguientes: que los estudiantes (i) reconozcan que las computadoras no se limitan a las de escritorio y las portátiles, (ii) conozcan el modelo conceptual de arquitectura de computadoras de von Neumann, y (iii) comprendan que ese modelo describe a las computadoras que se usan cotidianamente.

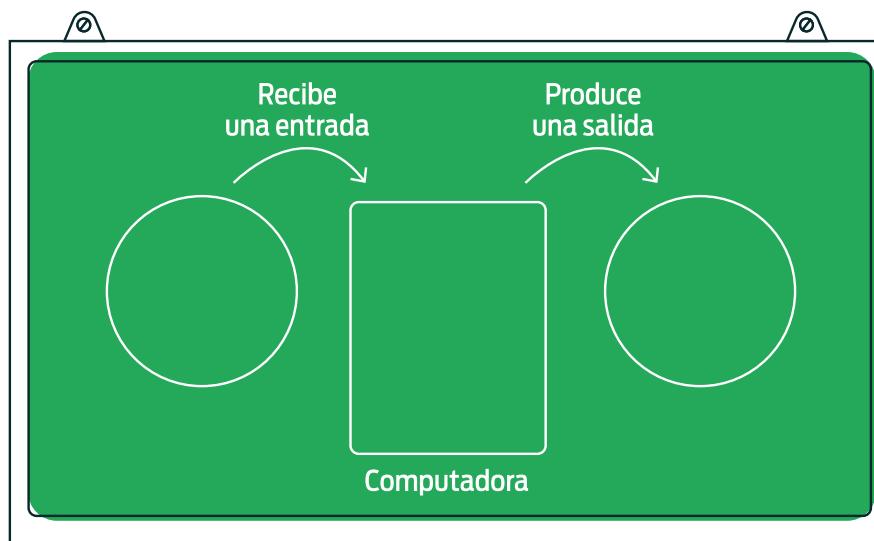
Comenzamos preguntándoles a los estudiantes: “¿Qué computadoras conocen?”. Es probable que mencionen las computadoras de escritorio y las portátiles. Continuamos: “Podría decirse que, en esencia, lo que hace una computadora es tomar una o varias entradas, hacer algo con ellas y producir un resultado o salida. Por ejemplo, usando un micrófono podemos ingresar un audio en la computadora, que luego esta transforma en un archivo de sonido y guarda en un disco rígido; o, presionando un botón, podemos mandar un documento para que se imprima en una impresora. Teniendo esto en cuenta, ¿se les ocurren otras cosas que también sean una computadora o tengan dentro una?”.

Guiamos la discusión para llegar a la conclusión de que, por ejemplo, los teléfonos inteligentes y las tabletas también son computadoras. Mencionamos, además, otros ejemplos menos evidentes: un artefacto de aire acondicionado contiene una computadora que, a partir de la información que obtiene de un sensor de temperatura, activa y desactiva el mecanismo de refrigeración; o una máquina fotográfica digital, que a partir de la información que recibe de una placa fotosensible, genera una imagen que muestra en una pantalla y un archivo que guarda en una tarjeta de memoria.



Ejemplos de artefactos que contienen una computadora

Para graficar lo conversado, copiamos en el pizarrón el siguiente esquema (que iremos completando en el desarrollo de la actividad).



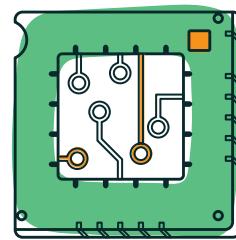
Esquema incompleto de un sistema de computación

A continuación, preguntamos: “¿Cómo hace una computadora para, a partir de una entrada, generar una salida?”. La respuesta esperable es: siguiendo al pie de la letra las instrucciones de un programa. En caso de que esta no surja espontáneamente, les recordamos a los estudiantes que, en actividades anteriores, ellos construyeron programas y vieron que la computadora seguía sus instrucciones.

Les explicamos que una computadora está formada por una serie de componentes interconectados, cada uno con una función específica, que le permiten llevar a cabo todo lo mencionado. Los componentes más importantes son la unidad central de procesamiento, cuya función principal es procesar información siguiendo las instrucciones de los programas, y la memoria, que es el componente físico donde se almacena la información para ser procesada. Además, para recibir información, una computadora se vale de dispositivos de entrada –como un teclado o un ratón– y, para comunicar un resultado o salida, de dispositivos de salida –como una impresora o un monitor–. Les comentamos que también hay dispositivos que cumplen ambas funciones, llamados *de entrada y salida*, como, por ejemplo, las pantallas táctiles.

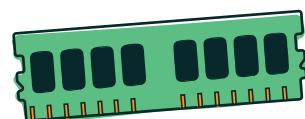
#### UNIDAD CENTRAL DE PROCESAMIENTO

La unidad central de procesamiento, también llamada *procesador* o *CPU* (por la sigla en inglés de *Central Processing Unit*), es el componente que procesa los datos de entrada para producir una salida. Para hacerlo, ejecuta una por una las instrucciones de un programa realizando operaciones aritméticas y lógicas.



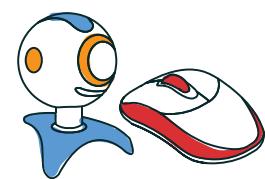
### MEMORIA

La memoria es un componente físico que permite almacenar información. En general, al hablar de memoria se hace referencia a la memoria principal o RAM,<sup>1</sup> por la sigla en inglés de *Random Access Memory*.



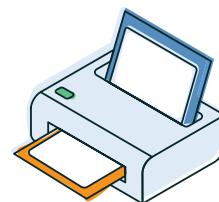
### DISPOSITIVOS DE ENTRADA

Los dispositivos de entrada permiten que la computadora reciba información. Entre ellos, están los que sirven para que los usuarios ingresen datos y, de esta manera, controlen el funcionamiento de las computadoras. Algunos ejemplos son el teclado, el ratón, las pantallas táctiles, las cámaras web y los lectores de códigos de barras. Hay otros que, en algunas circunstancias, funcionan sin intervención humana, como los sensores de distintos tipos –de temperatura o de proximidad, por ejemplo–.

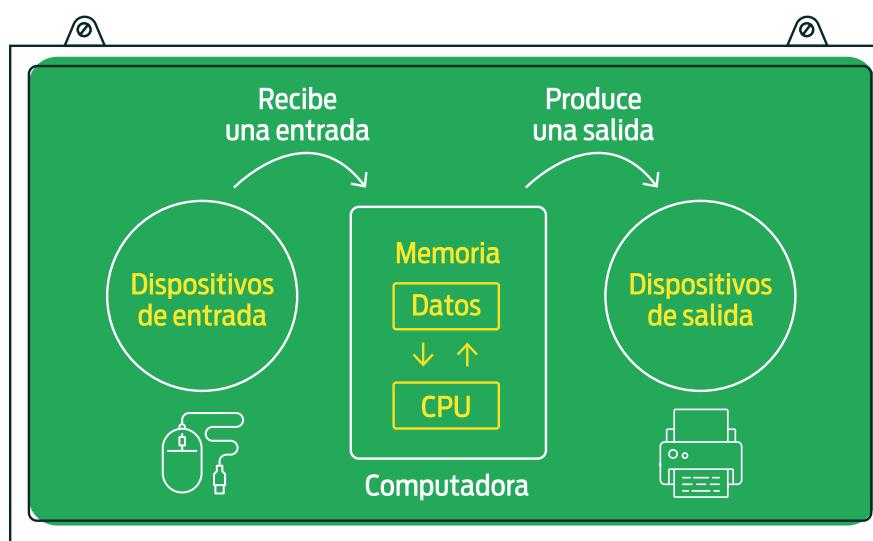


### DISPOSITIVOS DE SALIDA

Los dispositivos de salida son aquellos que usa la computadora para comunicar al exterior los resultados de un procesamiento; por ejemplo, una impresora, un monitor, parlantes, auriculares, proyectores, etc.



Continuamos: “¿Les parece que existe comunicación entre la unidad central de procesamiento y la memoria? ¿En qué dirección?”. Sí, existe comunicación en ambos sentidos. La CPU lee datos de la memoria para procesarlos y también escribe allí algunos resultados que producen sus cómputos. Entonces, agregamos al gráfico los componentes mencionados.

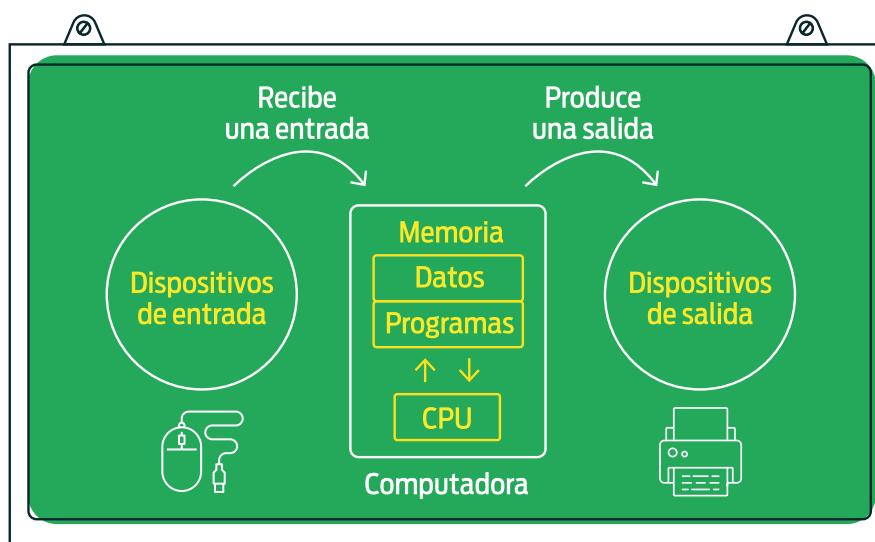


Incorporación de componentes al esquema

<sup>1</sup>Si bien hay otros tipos de memoria con características distintas, en esta actividad solo nos referiremos a la memoria principal.

A continuación les decimos: "Lo más habitual es que, en una computadora, usemos muchos programas. Por ejemplo, en nuestros teléfonos tenemos aplicaciones de mensajería instantánea, juegos, un visor de fotos, etc., y no hay ningún problema para ejecutar cada uno de ellos. De hecho, instalamos y desinstalamos aplicaciones con mucha frecuencia, ¿verdad? Es decir, las computadoras que usamos habitualmente no son máquinas construidas para ejecutar un programa en particular, sino que tienen la capacidad de ejecutar muchos programas distintos. Incluso, podemos ejecutar programas que no existían cuando la computadora fue fabricada! Los programas son, entonces, piezas de *software* que podemos poner y sacar de nuestros dispositivos. Esto descarta de cuajo que se correspondan con algún componente físico de la computadora".

Continuamos: "Ahora bien, hemos mencionado que la unidad central de procesamiento ejecuta una por una las instrucciones de un programa, pero... ¿cómo se entera la CPU de la instrucción que tiene que ejecutar? ¿Dónde están los programas?". Escuchamos las respuestas de los estudiantes y las discutimos entre todos. Guiamos el intercambio para llegar a la conclusión de que, para ejecutarse, los programas deben estar almacenados en algún lugar. Les explicamos, entonces, que se cargan en una porción reservada de la memoria (es decir, en una región separada de la que almacena datos), y es allí donde la unidad central de procesamiento va a buscar las instrucciones que tiene que seguir a medida que el programa se ejecuta. Completamos luego el esquema en el pizarrón, agregando los programas en la memoria.



Arquitectura de von Neumann

Repartimos la ficha de la actividad y les solicitamos a los estudiantes que completen las consignas. En la primera se pide completar una tabla con cinco máquinas que sean una computadora o contengan una y, para cada una de ellas, identificar algún dispositivo de entrada y alguno de salida. A continuación se propone una posible solución.

MÁQUINA QUE ES O QUE CONTIENE UNA COMPUTADORA	DISPOSITIVO DE ENTRADA	DISPOSITIVO DE SALIDA
Lavavajillas	Botonera para establecer el programa de lavado	Pantalla en la que se muestra cuánto tiempo resta para completar un lavado
Reloj inteligente	Pantalla (táctil) que se usa para cargar y controlar aplicaciones	Pantalla (táctil), donde se ven los datos que muestran las aplicaciones
Cajero automático	Botonera donde se ingresan claves, se seleccionan operaciones, etc.	Impresora, que imprime el ticket resultante de una operación
Aire acondicionado	Sensor de temperatura	Display que muestra la temperatura seleccionada
Máquina fotográfica digital	Sensor de luz	Pantalla

Possible solución de la primera consigna

En la segunda consigna se presenta una publicidad ficticia de una computadora, en la que se describen también sus características técnicas. Los estudiantes tienen que identificar qué componentes de la computadora corresponden a cada una de las partes del modelo conceptual de computadora presentado. De contar con una conexión, se les puede sugerir que consulten en Internet sobre las características de la computadora que no les resulten reconocibles.

A continuación se muestra el anuncio y la solución de la consigna.

# Bet&Rob Pro

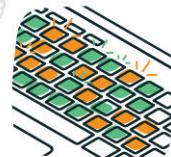
La nueva computadora *Bet & Rob Pro* cuenta con un **microprocesador Intel Core i7-7700HQ** que maneja cargas de trabajo pesadas en minutos. Este y la **RAM DDR2-667 de 32 Gigabytes** le permitirán trabajar con programas complejos y múltiples pestañas, incluso ejecutando varias aplicaciones a la vez.



Con la **pantalla táctil full HD** podrá comandar la máquina usando los diez dedos de la mano, y la **cámara frontal de 8 megapíxeles** le permitirá sacarse fotos cuando esté procrastinando.



TOUCHPAD HIPERSENSIBLE

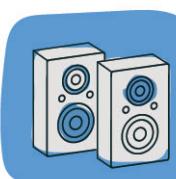
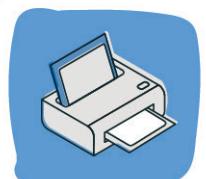


TECLADO LUMINOSO



LECTOR DE HUELLAS

Además, tiene **touchpad hipersensible**, teclado luminoso y **lector de huellas dactilares**.



**De regalo,** unos parlantes **holofónicos** y una **impresora láser color**.

Anuncio ficticio

<b>CPU</b>	Intel Core i7-7700HQ
<b>MEMORIA</b>	RAM DDR2-667 de 32 Gigabytes
<b>DISPOSITIVOS DE ENTRADA</b>	Pantalla (táctil) Cámara Touchpad Lector de huellas dactilares
<b>DISPOSITIVOS DE SALIDA</b>	Pantalla (táctil) Parlantes holofónicos Impresora láser color

Solución de la segunda consigna

### CIERRE

Les comentamos a los estudiantes que el modelo conceptual de computadora descrito, con una unidad central de procesamiento y una memoria en la que residen datos y programas, y que se comunica con el exterior mediante dispositivos de entrada y salida, fue ideado por el matemático de origen austro-húngaro John von Neumann en el año 1945. A pesar de haber pasado ya muchos años, este modelo describe cómo están organizadas la gran mayoría de las computadoras actuales. En su honor, recibe el nombre de *arquitectura de von Neumann*.

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# ¿QUÉ SON LAS COMPUTADORAS?

¿Qué caracteriza a las computadoras? ¿Las hay de diferentes formas y tamaños? ¿Qué cosas tienen en común todas ellas?

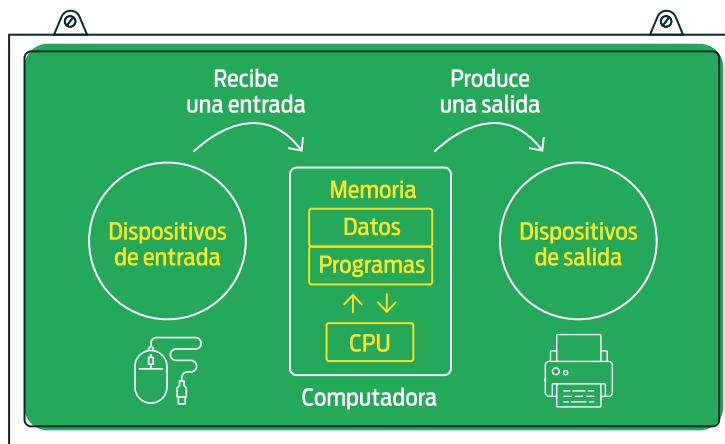
1. Completá la tabla con cinco computadoras que no sean ni las portátiles ni las de escritorio. Para cada una de ellas identificá, al menos, un dispositivo de entrada y uno de salida.



MÁQUINA QUE ES O QUE CONTIENE UNA COMPUTADORA	DISPOSITIVO DE ENTRADA	DISPOSITIVO DE SALIDA

## ARQUITECTURA DE VON NEUMANN

En 1945, el matemático de origen austrohúngaro John von Neumann presentó un modelo conceptual de arquitectura de computadoras, cuyo diseño sigue vigente en las computadoras modernas. En este modelo, conocido como *arquitectura de von Neumann*, una computadora está compuesta por una unidad central de procesamiento –que se encarga de ejecutar las instrucciones de los programas–, una memoria –en la que se almacenan los datos y los programas–, y dispositivos de entrada y salida que permiten el ingreso y egreso de datos.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

2. Explorá el siguiente anuncio. Relacioná los elementos de la computadora publicitada con los componentes descritos en el modelo de von Neumann.

# Bet&Rob Pro

La nueva computadora *Bet & Rob Pro* cuenta con un microprocesador Intel Core i7-7700HQ que maneja cargas de trabajo pesadas en minutos. Este y la RAM DDR2-667 de 32 Gigabytes le permitirán trabajar con programas complejos y múltiples pestañas, incluso ejecutando varias aplicaciones a la vez.



Con la pantalla táctil *full HD* podrá comandar la máquina usando los diez dedos de la mano, y la cámara frontal de 8 megapíxeles le permitirá sacarse fotos cuando esté procrastinando.



TOUCHPAD HIPERSENSIBLE

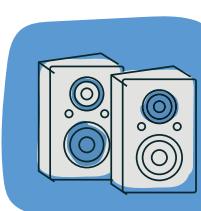
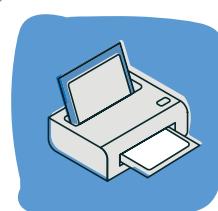


TECLADO LUMINOSO



LECTOR DE HUELLAS

Además, tiene *touchpad* hipersensible, teclado luminoso y lector de huellas dactilares.



De regalo,  
unos parlantes  
holofónicos y una  
impresora láser color.

CPU	
MEMORIA	
DISPOSITIVOS DE ENTRADA	
DISPOSITIVOS DE SALIDA	

## Actividad 2

### Una computadora a cielo abierto



GRUPAL (4)

#### OBJETIVOS

- Conocer una computadora por dentro.
- Identificar los componentes de una computadora.

#### MATERIALES



Computadoras de escritorio



Destornilladores



Ficha para estudiantes

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes desensamblen una computadora de escritorio para tener contacto directo con el *hardware* que la compone. En el camino, reconocerán sus componentes principales, como la unidad central de procesamiento y la memoria, y (esperamos) perderán el miedo que produce desarmar artefactos electrónicos.

Para desarrollar la actividad es necesario contar con computadoras de escritorio, preferentemente, una por cada cuatro estudiantes. Puede llevarse a cabo tanto en un laboratorio como en el aula. Si bien son artefactos robustos, es importante que, al desarmarlos, los alumnos sean cuidadosos y prolíjos. Por ejemplo, deben tener presente dónde dejan los tornillos que saquen para poder volver a ensamblar la computadora.

Les repartimos a los estudiantes la ficha, en donde encontrarán una guía para desarmar la computadora. Lo primero que observarán es el **gabinete**. Se trata de una caja, por lo general metálica, que contiene los principales componentes de la computadora, como la unidad central de procesamiento, la memoria, unidades de almacenamiento –por ejemplo, discos rígidos–, etc.



Gabinete cerrado

Lo primero que debemos hacer es asegurarnos de que todas las computadoras se encuentren apagadas. Si están encendidas, les pedimos a los estudiantes que las apaguen y las desenchufen del tomacorriente. Además, deben desconectar todos los cables que se encuentran enchufados al gabinete, tanto en el frente como en la parte posterior. De este modo, habrán desconectado todos los periféricos –como monitores, teclados, etc.–, lo cual facilitará la inspección del interior de la máquina.

En la parte trasera de un gabinete suele haber pequeños tornillos que hay que desatornillar para retirar las tapas laterales y acceder al interior. Una vez que todos las hayan abierto, les decimos: "Como pueden observar, la computadora está formada por muchos componentes que trabajan de forma integrada. Vamos a detenernos en algunos de ellos".



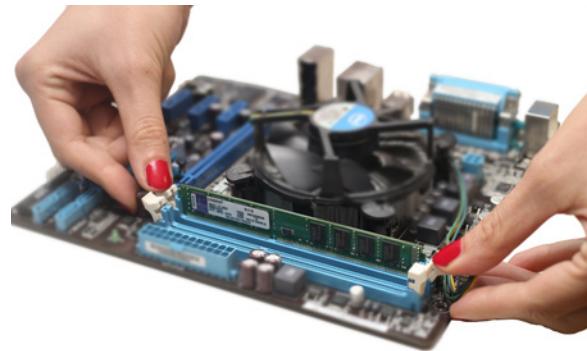
Gabinete abierto

Lo primero que tienen que identificar es la **placa madre** (o *motherboard*, en inglés). Se trata de una placa en la que se montan algunos de los componentes principales –como la CPU y la memoria–. Además, cuenta con muchos zócalos libres que permiten adosarle otros componentes y circuitos impresos que permiten que los distintos componentes se comuniquen. Comentamos: "Los circuitos impresos que observan permiten el flujo de datos entre los distintos componentes. Por ejemplo, entre la CPU y la memoria. La función principal de la placa madre es, justamente, que todos los subsistemas que forman parte de la computadora puedan comunicarse".



Placa madre

A continuación, tienen que identificar y retirar un módulo de memoria. Les contamos: "Habitualmente, cuando hablamos de **memoria** nos referimos a la memoria RAM, que es la memoria principal de la computadora. Allí es donde residen los programas mientras se ejecutan, y también donde están los datos que el programa procesa". A continuación les pedimos que retiren un módulo. Los módulos de memoria se encuentran en unas ranuras de la placa madre. Para sacar uno, primero se deben empujar hacia afuera los ganchos de retención que se encuentran en los extremos de las ranuras. Luego, se puede retirar tirando hacia arriba, sin hacer demasiada fuerza. A pesar de tratarse de un componente electrónico sofisticado, no son frágiles y resulta fácil volver a colocarlos, con lo que alentamos a los estudiantes a que lo retiren sin miedo.



Se retira un módulo de memoria

Una vez que todos los grupos hayan retirado un módulo de memoria, les preguntamos: "¿Para qué creen que son los contactos dorados que se encuentran en la parte inferior?". Escuchamos sus respuestas y, en caso de que no surja de algún estudiante, les decimos que a través de ellos ingresan y egresan datos a la memoria. Continuamos: "¿Saben qué son los sectores negros?". Allí es donde se almacena la información. Resaltamos: "Hasta ahora, al referirnos a la composición de una computadora, siempre hemos hablado de componentes tales como la CPU o la memoria, considerando a cada uno como una unidad indivisible. Sin embargo, como puede observarse, la memoria está formada por otros elementos más pequeños, que trabajan coordinados. Esto también es así en el resto de los componentes de la computadora".



Módulo de memoria RAM

Continuamos: “En general, para referirnos a la unidad central de procesamiento, decimos simplemente **procesador** ¿Ven dónde está?”. Es probable que no consigan identificarla, pues suele colocarse debajo de un ventilador que la cubre por completo. Les explicamos que, debido a que trabajan a gran velocidad, los procesadores suelen levantar mucha temperatura y, por lo tanto, hay que refrigerarlos para que no se quemen. Además, tanto el procesador como el zócalo en el que se encuentra son componentes muy frágiles, por lo que no es recomendable que intenten extraerlo.



Ventilador para procesadores

Les indicamos que observen la imagen del procesador que se encuentra en la ficha y comentamos: “El procesador tiene un aspecto muy simple en relación con la complejidad de su funcionamiento. Se lo encierra en una carcasa metálica para protegerlo porque es un componente muy delicado. Además, de este modo, se facilita la disipación de calor. ¿Se imaginan para qué son los contactos dorados que se encuentran en el reverso?”. Para el ingreso y egreso de datos, tanto cuando interactúa con la memoria como con dispositivos de entrada y salida.



Unidad central de procesamiento

Siguiendo las indicaciones de la ficha, los estudiantes tienen que identificar **dispositivos de almacenamiento**. Si bien existen otros externos –como dispositivos portátiles de almacenamiento o tarjetas SD, por ejemplo–, los que encontramos en el interior del gabinete son los **discos**. Estos no se encuentran montados directamente sobre la placa madre, sino que se conectan a ella mediante cables. Una vez que los hayan identificado comentamos: “En los discos se almacena información. Entonces, ¿qué los diferencia de la memoria RAM?”. Escuchamos sus respuestas y continuamos: “La memoria RAM es el componente en el que se cargan datos y programas para que un programa pueda ejecutarse. Sin embargo, para funcionar, la RAM necesita corriente eléctrica, con lo que no preserva su contenido cuando la computadora se apaga. Por tal motivo se dice que es una memoria volátil. Por el contrario, los dispositivos de almacenamiento conservan la información aun cuando la computadora se apaga.”

Es allí donde se almacenan de forma permanente los datos, como nuestras fotos, discos de música, programas instalados, etc. Como habrán notado, si prendemos y apagamos la computadora, los seguimos teniendo. En general, nos referimos a ellos como **discos duros o rígidos**".



Disco rígido

Les comentamos que existen distintas tecnologías de discos, y que los más habituales en la actualidad son los **discos rígidos rotacionales HDD** (por la sigla en inglés de *Hard Disk Drive*) y los **discos de estado sólido SSD** (por la sigla en inglés de *Solid State Drive*). Los discos rígidos rotacionales tienen un cabezal y platos magnéticos giratorios en los que se guarda información. Por su parte, los de estado sólido no tienen componentes mecánicos; su composición interna es más parecida a los de una memoria USB. Los discos de estado sólido son tecnológicamente más modernos que los rotacionales. En ellos, la velocidad a la que se puede leer o escribir información es sustancialmente mayor, y por eso también son considerablemente más caros.

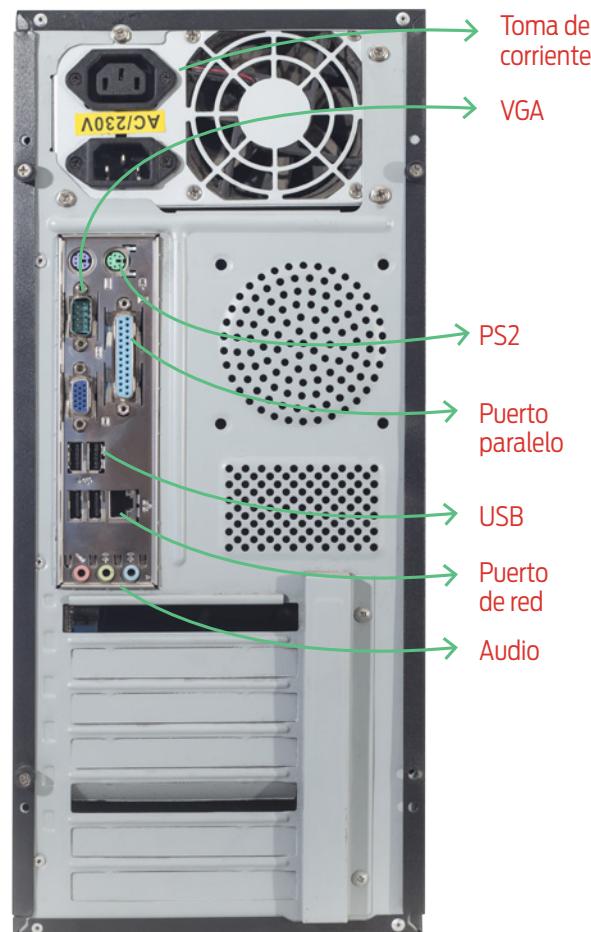


Discos HDD (parte inferior) y SDD

A continuación indagamos: “¿Dónde están los dispositivos de entrada y salida?”. Les comentamos que, aun cuando no existe un consenso sobre este punto, hay quienes consideran que los discos son periféricos de entrada y salida, ya que de allí se lee información que alimenta la memoria para hacer cómputos –por ejemplo, al cargar un documento en un procesador de texto–, y, también, en algunos casos, el resultado de procesar información allí se almacena –como al sacar una foto con una cámara incorporada en la computadora que se guarda como un archivo–.

Continuamos preguntando: “¿Y el resto de los componentes? ¿El ratón, el teclado, el monitor, etc.?”. Estos componentes son externos, nunca se encuentran dentro del gabinete de una computadora. Sin embargo, en un borde de la placa madre hay una serie de **puertos** a los que estos dispositivos se conectan. Habitualmente se encastran en la parte trasera del gabinete, que tiene perforaciones para dejar los puertos expuestos al exterior. Los puertos que encontrarán dependen de la computadora usada durante la actividad. Algunos que pueden encontrar son puertos VGA, DVI, HDMI –todos ellos para dispositivos de video, como un monitor–, PS2 –para conectar teclados y ratones–, etc.

Si bien no vamos a profundizar en esto, les mencionamos que hay componentes que, por su complejidad, suelen tener una placa propia que se conecta a la placa madre por cables. Por ejemplo, las placas de red, que tienen puertos que están expuestos al exterior del gabinete para poder enchufarlos a otros dispositivos, como routers o módems. O las placas de video, que transforman la información para que se pueda mostrar en dispositivos de salida, tales como monitores, proyectores, etc.<sup>1</sup>



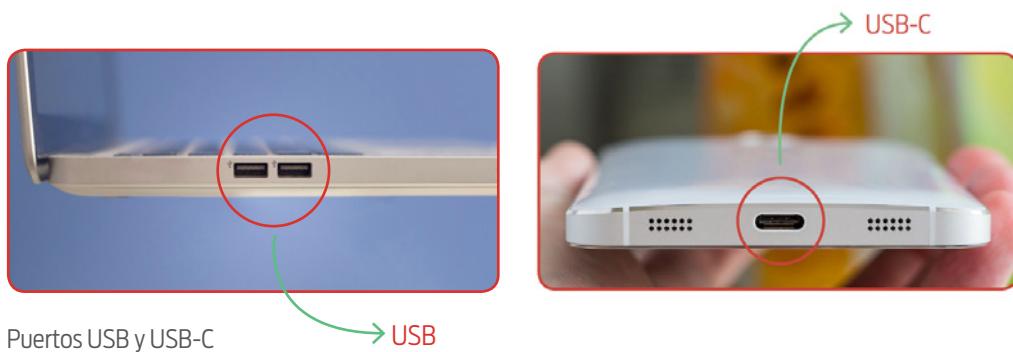
Puertos para componentes externos



Placa de video

<sup>1</sup> En rigor, una placa de video es en sí misma una computadora, ya que tiene un procesador, memoria propia, etc., aunque desde fuera del gabinete se vea un puerto como cualquier otro.

A continuación, les hacemos notar los puertos USB (del inglés *Universal Serial Bus*) y preguntamos: “¿Para qué sirven los puertos USB? ¿Qué conectamos en ellos?”. Actualmente, es habitual que las computadoras posean puertos USB que permiten conectar una gran cantidad de periféricos muy diversos, como ratones, discos externos, teléfonos, micrófonos, etc. Por lo tanto, estos puertos nos posibilitan ampliar los dispositivos de entrada y salida que interactúan con la computadora. La versatilidad provista por estos canales se basa en una técnica conocida como *enchufar y usar* (*plug and play*, en inglés). En la actualidad, las computadoras suelen venir equipadas con uno o varios de estos puertos, tanto en su versión original como en una más moderna llamada USB-C.



Finalmente, les pedimos que vuelvan a ensamblar la computadora y la dejen tal como estaba antes de comenzar la actividad.

### CIERRE

Hacemos notar a los estudiantes que entraron en contacto con algo muy concreto que tocaron con sus manos: el *hardware* de una computadora –es decir, sus partes tangibles–. Aun así, todo lo observado se ajusta perfectamente al modelo conceptual de von Neumann: una computadora formada por una unidad central de procesamiento, una memoria y dispositivos de entrada y salida.

NOMBRE Y APELLIDO:

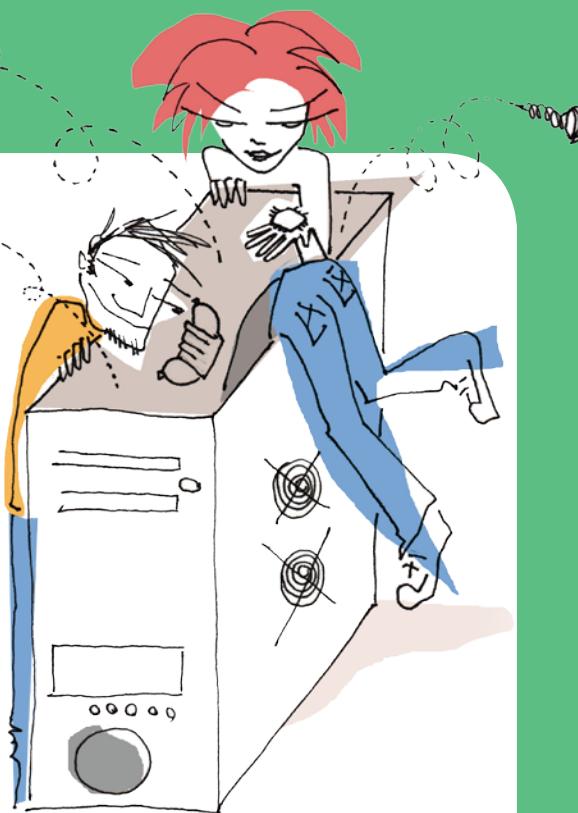
CURSO:

FECHA:

# UNA COMPUTADORA A CIELO ABIERTO

¿Cómo es una computadora por dentro? Seguí las indicaciones para investigarlo. ¡A arremangarse y meter mano, que así también se aprende!

1. Asegurate de que la computadora no esté conectada al tomacorriente y desenchufale todos los cables, de modo que quede solo el gabinete. Desatornillá todos los tornillos y retirá la tapa lateral. ¿Reconocés algún componente? ¿La memoria? ¿El procesador?



---

---

---

---



2. Retirá un módulo de memoria. ¿Para qué son los conectores dorados que hay en la parte inferior? ¿Y qué son los bloques negros?

---

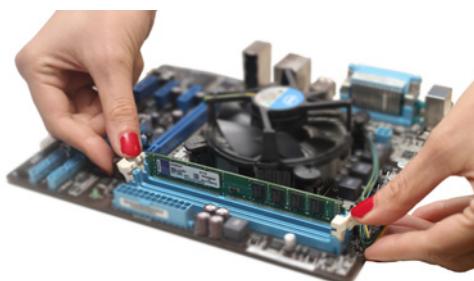
---

---



## PARA SACAR UN MÓDULO DE MEMORIA

Los módulos de memoria se encuentran en unas ranuras de la placa madre. Para sacar uno, primero se deben empujar hacia afuera los ganchos de retención ubicados en los extremos de las ranuras. Luego, se puede retirar tirando hacia arriba.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

3. ¿Podés ver la unidad central de procesamiento (o procesador)?

¿Por qué? Mirá la imagen de la derecha. ¿Para qué sirven los contactos dorados?

---

---

---



4. Observá qué unidad de almacenamiento encontrás. ¿De qué tipo es? ¿Está montada sobre la placa madre?

---

---

---



#### DISCOS

Existen distintas tecnologías de discos; los más habituales en la actualidad son los **discos rígidos rotacionales HDD** (por la sigla en inglés de *Hard Disk Drive*) y los **discos de estado sólido SSD** (por la sigla en inglés de *Solid State Drive*). Estos últimos son más modernos, más rápidos y más caros. Los rotacionales tienen un cabezal y platos magnéticos giratorios, en los que se guarda información. Por su parte, los de estado sólido no tienen componentes mecánicos; su composición interna es más parecida a los de una memoria USB.



5. ¿Encontraste algún dispositivo de entrada y salida?

---

---

---

6. ¿Para qué sirven los puertos que asoman por las perforaciones que tiene el gabinete?

¿Cuáles encontraste?

---

---

---

7. ¿En qué se diferencian los puertos USB del resto de los puertos que hay en el gabinete?

---

---

8. Volvé a ensamblar la computadora y dejala como al principio.



## Secuencia Didáctica 2

# LA UNIDAD CENTRAL DE PROCESAMIENTO Y LA MEMORIA

La materia prima con la que trabajan las computadoras es la información: la reciben y la transforman para producir información nueva. Hay dos componentes, entonces, que resultan indispensables para que una computadora pueda funcionar: uno que tenga la capacidad de procesar la información y otro que tenga capacidad de representarla.

Esta secuencia didáctica hace foco tanto en la unidad central de procesamiento como en los dispositivos de memoria. Las actividades propuestas conducirán a los estudiantes a analizar las características que determinan el rendimiento de la CPU y los volúmenes de información que se representan en la memoria, y les permitirán conocer un mecanismo que hace posible acelerar la interacción entre ambos componentes.

.....  
**OBJETIVOS**

- Identificar variables que determinan el rendimiento de la CPU.
  - Dimensionar los volúmenes de información que almacenan los dispositivos de memoria.
  - Conocer el mecanismo de *caching*.
- .....

## Actividad 1

### Zoom in a la unidad central de procesamiento



#### OBJETIVO

- Conocer las principales características de los procesadores.

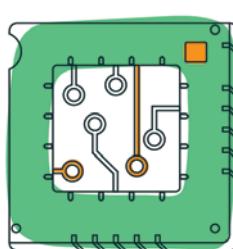
#### MATERIALES

- Computadora
- Internet
- Ficha para estudiantes

#### DESARROLLO

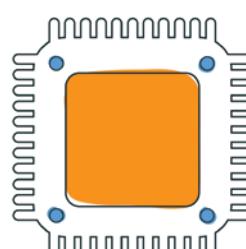
El objetivo de esta actividad es que los estudiantes conozcan las características que definen la capacidad y rendimiento de una unidad central de procesamiento.

Comenzamos repartiendo la ficha a los estudiantes y los invitamos a que completen la primera consigna. Encontrarán allí tres anuncios publicitarios de tres procesadores, en los que pueden observarse algunas especificaciones técnicas de cada uno. Los dos primeros son de procesadores como los que suelen usarse en computadoras de escritorio o servidores; el tercero es de un procesador para dispositivos móviles como, por ejemplo, un teléfono inteligente.



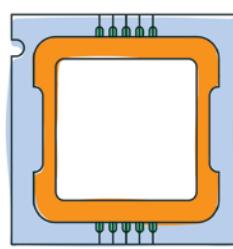
#### Intel® Core™ i7

- 6 núcleos 4.5 GHz
- Memoria compatible DDR3/DDR4.



#### AMD Ryzen 7 / 2700

La verdadera inteligencia, compuesta por 8 núcleos, una frecuencia de reloj de 4.1 GHz, compatible con memorias DDR4.



#### Qualcomm Snapdragon 835

Un 30% más fino y eficiente, para un presente de dobles cámaras y realidad virtual. Los 8 núcleos de 2.45 GHz se integran con memorias LPDDR4, lo que genera máxima eficiencia en dispositivos móviles.

Anuncios publicitarios de procesadores

Para resolver la consigna, los estudiantes tienen que escribir en una tabla las características de los procesadores de los anuncios. Además, deben buscar en Internet las especificaciones de otros dos procesadores para completar las últimas dos filas de la tabla.<sup>1</sup>

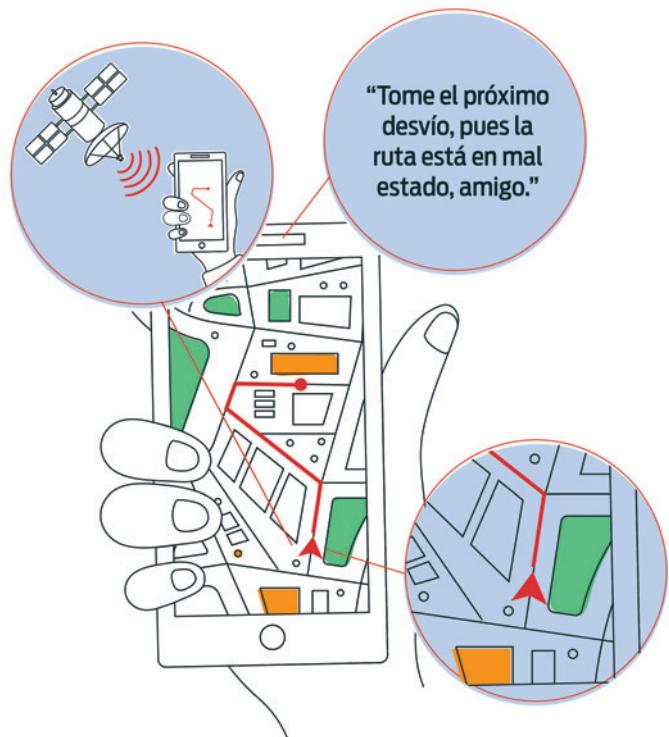
<sup>1</sup> Se les puede sugerir que realicen las búsquedas en sitios de venta en línea.

MARCA	MODELO	CANTIDAD DE NÚCLEOS	FRECUENCIA	COMPATIBILIDAD CON MEMORIA
Intel	i7	6	4.5 GHz	DDR3/DDR4
AMD	Ryzen 7 / 2700	8	4.1 GHz	DDR4
Qualcomm	Snapdragon 835	8	2.45 GHz	LPDDR4

Características de los procesadores de los anuncios

Una vez que hayan completado la tabla, les contamos que los procesadores de los anuncios de Intel y AMD se usan en computadoras de escritorio y servidores, y que el de Qualcomm se usa para dispositivos móviles.

Les preguntamos: “¿Recuerdan cuál es la función principal de una unidad central de procesamiento (o procesador)?”. En caso de que no surja la respuesta de parte de los estudiantes, les recordamos que su función principal es ejecutar una por una las instrucciones de un programa. Continuamos: “¿Qué quiere decir que un procesador tenga 2, 6, 8 o 16 **núcleos**?”. Entonces aclaramos: “Un procesador de múltiples núcleos está compuesto por dos o más unidades de procesamiento. Cada una tiene la capacidad de ejecutar instrucciones en forma independiente. Esto permite que el procesador pueda ejecutar varias instrucciones en núcleos separados al mismo tiempo, lo que, en general, aumenta la velocidad de ejecución de los programas. Esto resulta ventajoso, por ejemplo, cuando estamos ejecutando varios programas a la vez; o cuando un mismo programa realiza muchas tareas independientes en forma simultánea. Por ejemplo, los programas de navegación asistida que, simultáneamente, usan el GPS para determinar la ubicación del vehículo, al mismo tiempo que dibujan el mapa en la pantalla y dan indicaciones por el altavoz”.



Distintas instrucciones de un programa que se ejecutan en forma simultánea en distintos núcleos

A continuación les preguntamos a los estudiantes si saben qué es la frecuencia y les proponemos el siguiente ejercicio. Cada vez que nosotros aplaudimos, ellos deben pararse de sus asientos y volver a sentarse. Comenzamos aplaudiendo en intervalos regulares cada cinco segundos. Luego de un rato, lo hacemos más rápido (por ejemplo, cada dos segundos), cuidando en todo momento que el tiempo entre aplauso y aplauso sea aproximadamente el mismo. Una vez concluido el ejercicio, comentamos: “La frecuencia mide la cantidad de ocurrencias de un evento por alguna unidad de tiempo. Por ejemplo, al comenzar el ejercicio yo aplaudía 12 veces por minuto y, luego, 30. En este caso el evento es un aplauso y, la unidad de tiempo, un minuto”.

Les explicamos que un **hertz** (Hz) es la unidad de medida de frecuencia que indica que un cierto evento se produce cada un segundo. Un kilohertz (kHz) equivale a 1000 hertz –es decir que el evento ocurre 1000 veces por segundo–; un megahertz (MHz) equivale a 1000 kHz –ocurre un millón de veces por segundo–; y un gigahertz (GHz) equivale a 1000 MHz –el evento ocurre mil millones de veces por segundo–.

UNIDAD	Hertz (Hz)	Kilohertz (KHz)	Megahertz (MHz)	Gigahertz (GHz)
OCURRENCIAS DE UN EVENTO POR SEGUNDO	1	1.000	1.000.000	1.000.000.000

Unidades de medida de frecuencia

Preguntamos: “¿Por qué en los anuncios de procesadores se habla de frecuencia? ¿A qué se refiere este concepto?”. Escuchamos las respuestas y comentamos: “Internamente, un procesador está formado por muchos pequeños componentes (o partes). Para que el procesador funcione correctamente como un todo, esas partes tienen que trabajar juntas de manera sincronizada y coordinada. Todos los componentes internos están conectados a un **reloj** que emite una serie de pulsos eléctricos a intervalos constantes, como si fuera un director de orquesta marcando un ritmo. Cada componente interno comienza y termina su trabajo siguiendo algunas de estas marcas, también conocidas como *ticks* o *ciclos*. Este mecanismo es el que permite la sincronización entre las distintas partes”. Finalmente, para que dimensionen la velocidad a la que funcionan los procesadores, les hacemos notar que lo que se indica en los anuncios es la cantidad de ciclos de reloj por segundo: 1 GHz corresponde a mil millones de ciclos por segundo. Por lo tanto, un procesador de 4.5 GHz realiza cuatro mil quinientos millones de ciclos por segundo.

A continuación comentamos: “Los anuncios también mencionan que los procesadores tienen compatibilidad con algunas memorias. ¿A qué se refieren?”. Como sucede con casi cualquier componente electrónico, también de las memorias existen diferentes generaciones tecnológicas. Por ejemplo, en las memorias DDR, que en los anuncios son las compatibles con los procesadores Intel y AMD, el número (como 4, en DDR4) indica la generación a la que pertenece. Cada nueva generación suele tener mejor rendimiento que las anteriores. Con las memorias LPDDR –que son para dispositivos móviles, como teléfono celulares–, sucede lo mismo. Señalamos: “Si en algún momento piensan en comprar una memoria, deben asegurarse de que sea compatible con el procesador de sus computadoras”.

### CIERRE

A modo de conclusión, comentamos con los estudiantes que las características presentadas dan una noción del rendimiento de un procesador. Sin embargo, como en una computadora este no funciona en forma aislada, el rendimiento global del sistema dependerá también de muchos otros factores, tales como las características propias de otros componentes –memoria, placa madre, dispositivos de almacenamiento, etc.– y de la interacción entre ellos.

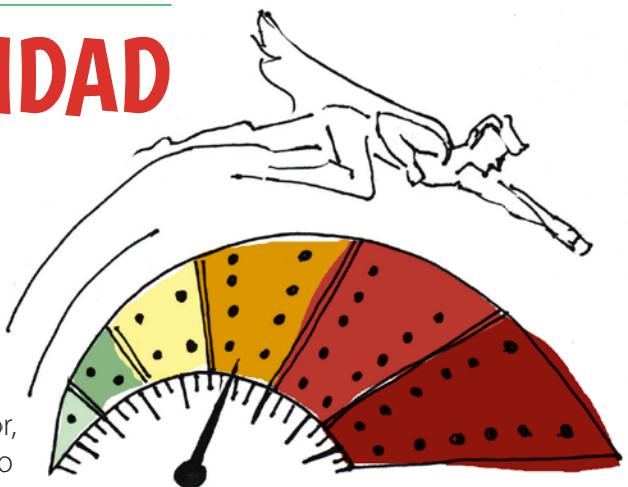
---

NOMBRE Y APELLIDO:

CURSO:

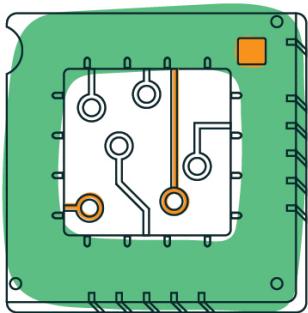
FECHA:

# ZOOM IN A LA UNIDAD CENTRAL DE PROCESAMIENTO



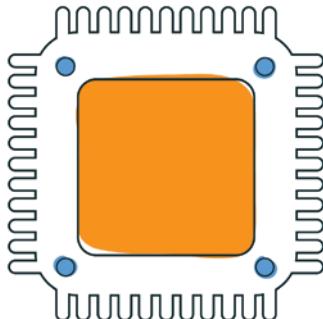
Cuando miramos el anuncio de venta de un procesador, ¿qué quieren decir los datos que se mencionan? ¿Cómo inciden en el rendimiento de nuestras computadoras?

1. Mirá los anuncios publicitarios y completá las tres primeras filas de la tabla que se encuentra en la siguiente página.



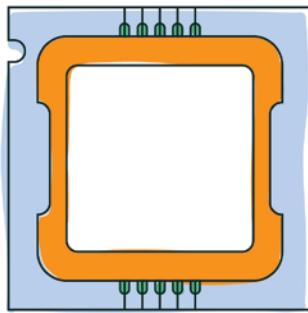
**Intel® Core™ i7**

- 6 núcleos 4.5 GHz
- Memoria compatible DDR3/DDR4.



**AMD Ryzen 7 / 2700**

La verdadera inteligencia, compuesta por 8 núcleos, una frecuencia de reloj de 4.1 GHz, compatible con memorias DDR4.



**Qualcomm Snapdragon 835**

Un 30% más fino y eficiente, para un presente de dobles cámaras y realidad virtual. Los 8 núcleos de 2.45 GHz se integran con memorias LPDDR4, lo que genera máxima eficiencia en dispositivos móviles.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

MARCA	MODELO	CANTIDAD DE NÚCLEOS	FRECUENCIA	COMPATIBILIDAD CON MEMORIA

**2.** Buscá en Internet dos anuncios más y completá las últimas dos filas.

**3.** ¿Cuál o cuáles de los procesadores de la tabla tiene capacidad de ejecutar más instrucciones en forma simultánea? ¿Por qué, de qué depende?

---

---

---

---

**4.** ¿Qué indica la frecuencia de un procesador? ¿Tiene alguna incidencia en la velocidad a la que funciona una computadora?

---

---

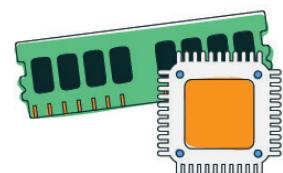
---

---

#### COMPATIBILIDAD CON LA MEMORIA

Para poder funcionar juntos, el procesador y la memoria deben ser compatibles.

Como sucede con casi cualquier componente electrónico, también entre las memorias existen diferentes generaciones tecnológicas. Un procesador, en general, admitirá memorias de una generación. Si querés ampliar la cantidad de memoria de tu computadora, asegurate de comprar una compatible.



## Actividad 2

### ¿Cuánto cabe en la memoria?



TODA LA CLASE

#### OBJETIVOS

- Presentar unidades de medida de información.
- Dimensionar el volumen de información que se almacena en un dispositivo de memoria.

#### MATERIALES



Anexo para estudiantes

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes dimensionen la cantidad de información que se puede almacenar en los dispositivos de memoria.

Comenzamos la clase preguntándoles a los alumnos: “¿Dónde se almacena la información que procesa una computadora?”. Es probable que algún estudiante conteste que se almacena en la memoria. Entonces, indagamos: “¿Qué es la memoria?”. Escuchamos sus respuestas y llegamos a la conclusión de que, cuando hablamos de memoria, nos referimos a una serie de componentes físicos que tienen la capacidad de representar información.

Continuamos: “¿Qué tipos de memoria conocen?”. Si no surgiera espontáneamente de los estudiantes, les recordamos que, mientras un procesador ejecuta un programa, lee datos de la memoria RAM para procesarlos y también escribe en ella el resultado de algunos de sus cálculos. Además, en un sector reservado, residen los programas mientras son ejecutados, y es ahí donde el procesador busca cuál es la siguiente instrucción que tiene que realizar mientras ejecuta un programa. Ya sean datos o programas, en la memoria RAM hay información. “Además de la RAM, ¿qué otros componentes pueden ser considerados memorias? Es decir, ¿qué otros componentes físicos tienen la capacidad de representar información?”. Si bien tienen características distintas, también los medios de almacenamiento –como discos, pendrives, etc.– son memorias, ya que pueden representar información.

Indagamos: “¿Qué quiere decir que una memoria sea de, por ejemplo, 8 GB?”. Es probable que algún estudiante responda que se trata del tamaño o, en forma más precisa, de la cantidad máxima de información que puede almacenar. Continuamos: “Ahora bien, ¿cómo se almacena la información en un dispositivo de memoria?”. Una memoria se puede representar como una gran tira de celdas contiguas. En cada una de las celdas se puede almacenar uno de dos valores: o bien un cero o bien un uno. Esta es la mínima unidad de información que una computadora puede representar y se llama **bit**.

Continuamos: “Así como para medir longitudes hay distintas unidades, como por ejemplo milímetros, centímetros y metros, lo mismo sucede cuando queremos medir cantidades de información. En general, no se piensa en términos de bits, sino que se usan unidades de medida más grandes. Por ejemplo, un **byte** equivale a 8 bits. Como cada bit puede tener un cero o un uno, si se cuentan todas las posibles combinaciones de ellos agrupados de a 8, en un byte se pueden almacenar 256 valores distintos, generalmente para representar los valores de 0 a 255”. Copiamos a continuación el siguiente gráfico en el pizarrón y les contamos que lo que observan en cada byte es la representación de un número en el sistema de numeración binario.

0	0	0	0	0	0	0	0	→ 0
0	0	0	0	0	0	0	1	→ 1
0	0	0	0	0	0	1	0	→ 2
0	0	0	0	0	0	1	1	→ 3
...								
1	1	1	1	1	1	1	0	→ 254
1	1	1	1	1	1	1	1	→ 255

Secuencias de bytes en representación binaria y decimal

Les explicamos: “Así como 8 bits son 1 byte, 1024 bytes son un Kilobyte (KB), 1024 KB son 1 Megabyte (MB), 1024 MB son 1 Gigabyte (GB) y 1024 GB son un Terabyte (TB).<sup>1</sup> Si hiciésemos las cuentas, podríamos ver que, por ejemplo, 8 GB son más de ocho mil millones de bytes. Cuando nos hablan de la capacidad de una memoria, se hace referencia a la cantidad de información que se puede almacenar en ella, medida en alguna de estas unidades”.

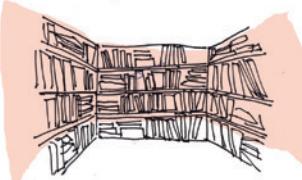
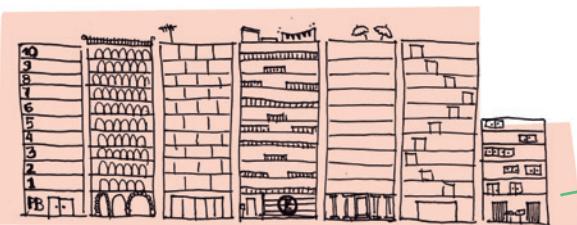
UNIDAD	Byte	Kilobyte (KB)	Megabyte (MB)	Gigabyte (GB)	Terabyte (TB)
TAMAÑO	8 bits	1024 bytes	1024 KB	1024 MB	1024 GB

Unidades de medida de información

Finalmente, para que dimensionen la cantidad de información que se puede guardar en los medios físicos, les presentamos la siguiente analogía: “Supongamos que cada letra de un libro ocupa 1 byte.<sup>2</sup> Entonces, una página de ese libro podría tener 1 KB de información, y un libro de 1024 páginas –un libro bastante gordo– sería el equivalente de 1 MB. Una habitación en la que hubiera 1024 libros distribuidas en bibliotecas contendría 1 GB de información. ¿Cuánto sería entonces 1 TB? Serían unas 1024 habitaciones... Si pensamos en un edificio de 10 pisos con 4 departamentos por piso y 4 habitaciones por departamento, en cada edificio hay 160 habitaciones. O sea, se precisan 6 edificios de 10 pisos y uno de 5 pisos, cada uno lleno de libros en todas las habitaciones para tener 1 TB de información. Y todo eso lo podemos guardar en un disco que cabe en la mochila. Impresionante, ¿no?”. A continuación, les entregamos el anexo de la actividad, que grafica lo expuesto.

<sup>1</sup>El uso del número 1024 como factor de multiplicación entre distintas unidades de medida se debe a que, por ser una potencia de 2 ( $2^{10} = 1024$ ), es de fácil manipulación para una computadora.

<sup>2</sup>Este no necesariamente es así; depende del sistema de codificación de texto que se use. En algunas codificaciones estándar, como ASCII, se usa un byte para representar un carácter, lo cual permite representar 256 símbolos. Otras más actuales, como UNICODE, llegan utilizar hasta 4 bytes, con los que permiten distinguir más de cuatro mil millones de símbolos.

CANTIDAD DE MEMORIA	PUEDE CONTENER...
1 byte	 Una letra de un libro
1 KB	 Una página de un libro
1 MB	 Un libro de 1024 páginas
1 GB	 Una habitación con 1024 libros
1 TB	 Los libros contenidos en 6 edificios de 10 pisos y uno de 5 pisos con cuatro departamentos de cuatro habitaciones por piso

Analogía para dimensionar volúmenes de información

Esto entra  
en un disco  
de un Terabyte.



## CIERRE

Les comentamos a los estudiantes que en 1980 se empezó a comercializar una de las primeras computadoras de uso hogareño en salir al mercado: la Commodore VIC-20, que tenía en total 5 KB de memoria RAM. Hoy en día, no es raro que un teléfono celular tenga alrededor de 3 GB. Algo así como 1600.000 veces más!

# ¿CUÁNTO CABE EN LA MEMORIA?

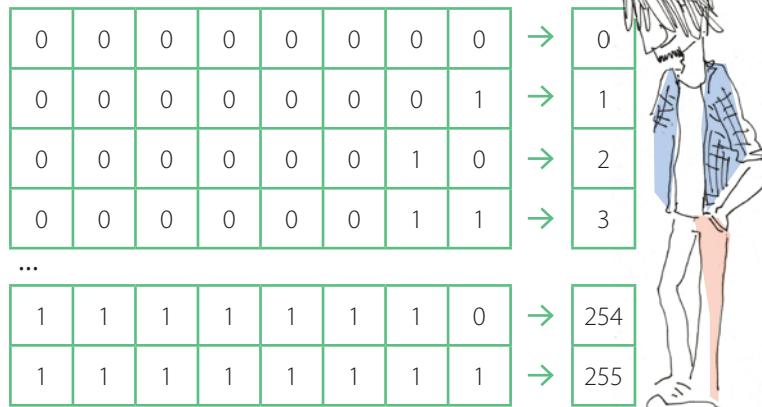
ANEXO



Cuando hablamos de memoria, nos referimos a una serie de componentes físicos –de *hardware*– que tienen la capacidad de representar información. Esto incluye tanto a la memoria RAM, como a los discos rígidos, los dispositivos portátiles de almacenamiento, etc.

## ¿CÓMO SE ALMACENA INFORMACIÓN EN UN DISPOSITIVO DE MEMORIA?

Una memoria se puede representar como una gran tira de celdas contiguas. En cada una se puede almacenar uno de dos valores: o bien un cero o bien un uno. Esta es la mínima unidad de información que una computadora puede representar y se llama **bit**. En general, no se piensa en términos de bits, sino que se usan unidades de medida más grandes. Por ejemplo, un **byte** equivale a 8 bits. Como cada bit puede tener un cero o un uno, al considerar todas las posibles combinaciones de ellos agrupados de a 8, podemos ver que en un byte se pueden almacenar 256 valores distintos.



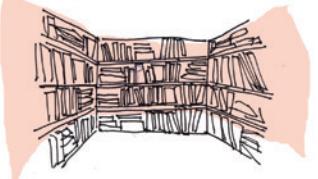
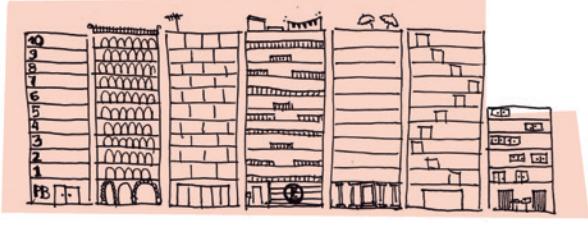
## BITS, BYTES Y LO DE MÁS ALLÁ

Así como para medir longitudes hay distintas unidades de medida, como por ejemplo milímetros, centímetros y metros, lo mismo sucede cuando queremos medir cantidades de información. Estas son algunas de ellas.

UNIDAD	Byte	Kilobyte (KB)	Megabyte (MB)	Gigabyte (GB)	Terabyte (TB)
TAMAÑO	8 bits	1024 bytes	1024 KB	1024 MB	1024 GB

El uso del número 1024 como factor de multiplicación entre distintas unidades de medida se debe a que, por ser una potencia de 2 ( $2^{10} = 1024$ ), es de fácil manipulación para una computadora.

Para dimensionar cuánta información cabe en un dispositivo de memoria, mirá la siguiente analogía.

CANTIDAD DE MEMORIA	PUEDE CONTENER...
1 byte	 Una letra de un libro
1 KB	 Una página de un libro
1 MB	 Un libro de 1024 páginas
1 GB	 Una habitación con 1024 libros
1 TB	 Los libros contenidos en 6 edificios de 10 pisos y uno de 5 pisos con cuatro departamentos de cuatro habitaciones por piso

### NOBLEZA OBLIGA

El espacio que se utiliza para codificar un carácter depende del sistema de codificación usado. Por ejemplo, algunas versiones de UNICODE llegan a usar cuatro bytes. Pero la codificación ASCII, que permite codificar el alfabeto latino, utiliza un solo byte para cada carácter. O sea que, si todos los libros están en castellano, ¡la analogía es precisa!

Esto entra  
en un disco  
de un Terabyte.



## Actividad 3

### La caché



INDIVIDUAL

#### OBJETIVOS

- Reconocer la idea de *caching*.
- Mostrar cómo se implementa el *caching*.
- Introducir jerarquía de memorias.

#### MATERIALES

- Ficha para estudiantes
- Teléfono inteligente
- Internet

#### DESARROLLO

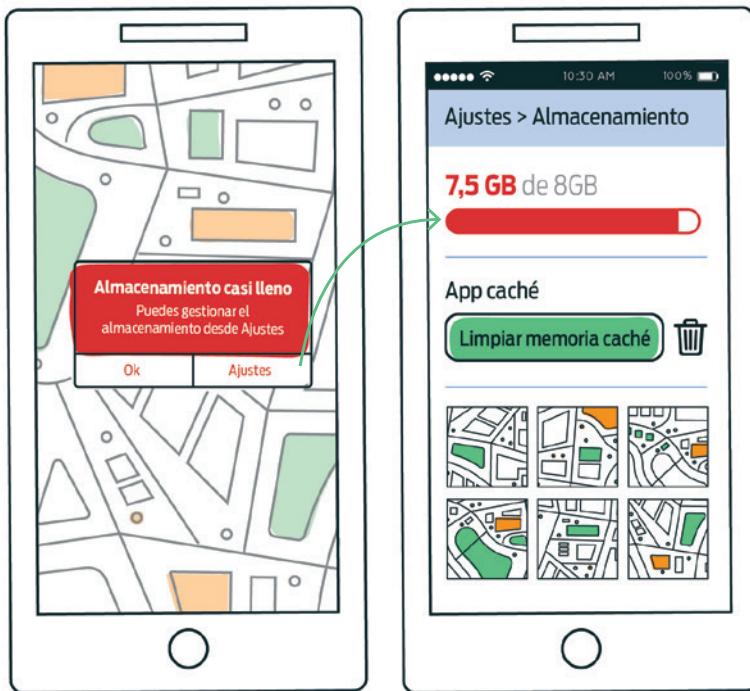
El objetivo de esta actividad es presentar el mecanismo de *caching*. Se trata de una estrategia utilizada por las computadoras para disminuir el tiempo de acceso a los datos con los que trabajan los programas. En primer lugar, se introduce la estrategia y, luego, se explica cómo hace una computadora para ponerla en práctica.

#### Idea de *caching*

Comenzamos planteando a los estudiantes la siguiente situación: "Supongan por un momento que en sus habitaciones tienen un escritorio que usan para estudiar y, unos cuantos pasos más allá, una biblioteca. Tanto sobre el escritorio como en los estantes de la biblioteca tienen carpetas y libros de estudio. Supongan, también, que hace dos días tuvieron un examen de Historia, dentro de dos días tienen uno de Matemática y dentro de dos semanas uno de Lengua y otro de Química. ¿Qué cosas imaginan que tendrían sobre el escritorio y qué cosas en la biblioteca?". Escuchamos las respuestas y las discutimos grupalmente. Es probable que los alumnos contesten que sobre el escritorio tendrían materiales de Matemática, posiblemente algunos de Historia que no se hubiesen encargado de guardar, y en la biblioteca los de Lengua y los de Química. "¿Por qué los tendrían de ese modo?".

Guiamos la discusión para arribar a la conclusión de que esa disposición se debe a que a los de Matemática los estarían consultando permanentemente, algunos de Historia porque todavía no se habrían ocupado de ordenarlos, y los restantes están en la biblioteca porque no necesitan consultarlos hasta dentro de un tiempo.

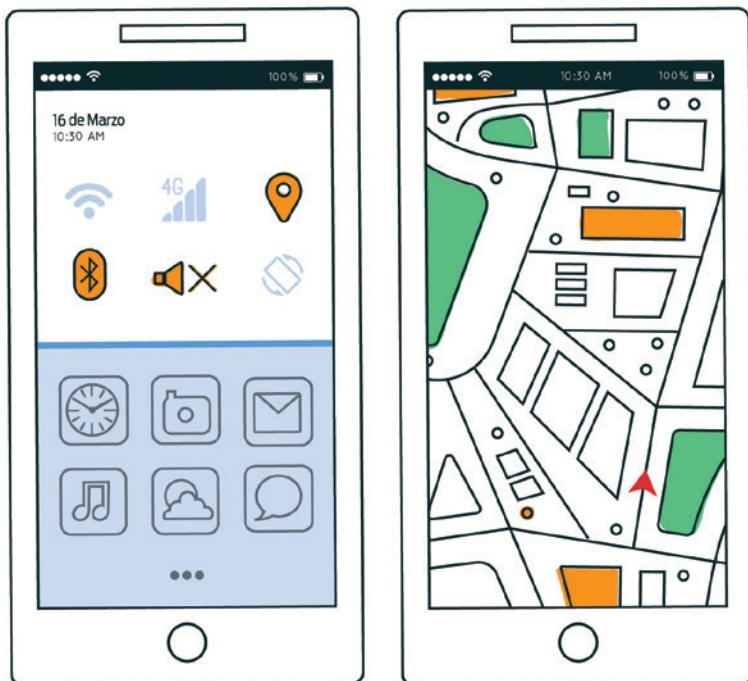
Luego, les preguntamos: "¿Alguna vez escucharon o leyeron la palabra *caché*? ¿Saben de qué se trata?". Seguramente, muchos la hayan visto aparecer en sus teléfonos. "Aun cuando no sepamos bien de qué se trata, ¿qué sugiere el teléfono hacer con la *caché*?". En general, los dispositivos y programas piden autorización para borrarla. "¿Cuándo aparece esta sugerencia?". Cuando se está agotando el espacio disponible en el almacenamiento interno del teléfono. "Muy bien, con esto podemos concluir que la *caché* es una porción del almacenamiento interno del teléfono que contiene información. Sin embargo, no pareciera tratarse de información indispensable para que el teléfono pueda funcionar. ¡Si así fuese no sugeriría borrarla!".



El teléfono pide autorización para borrar la caché

Continuamos: "Hoy en día es muy común el uso de aplicaciones de mapas, por ejemplo. Nos sirven, entre otras cosas, para obtener la ruta que nos permite ir de un lugar a otro, o para indicarnos qué medio de transporte tenemos que tomar para ir a algún sitio. Por supuesto, son muy útiles cuando queremos mirar mapas y, más aún, para ver fotografías de lugares que señalemos en ellos. ¿Cómo hace la aplicación para mostrarnos los mapas y las fotos? ¿De dónde los saca?". Es esperable que algún estudiante responda que la aplicación los baja de Internet.

Les proponemos entonces el siguiente experimento: que, en primer lugar, deshabiliten de sus teléfonos el uso de wifi y de transmisión de datos móviles y, a continuación, abran la aplicación de mapas que usan habitualmente. Verán entonces que en la pantalla aparece un mapa que, posiblemente, muestre el área en el que se encuentra la escuela. A continuación les indicamos que busquen la dirección de sus domicilios y verán, nuevamente, que en la pantalla aparece el mapa que muestra la ubicación de sus hogares. Preguntamos: "¿Cómo puede ser que veamos los mapas si no estamos conectados a Internet? ¿Dónde están guardados?". Escuchamos las consideraciones de los estudiantes y guiamos el intercambio para llegar a la conclusión de que, como el teléfono no está conectado a Internet (ni por una red wifi ni por la red de datos móviles de telefonía celular), los mapas necesariamente deben estar guardados en sus teléfonos.



Conexiones a Internet deshabilitadas y visualización de mapa sin conexión.

Les indicamos a continuación que, sin volver a conectar sus dispositivos a Internet, usen la aplicación para buscar el mapa de Kiev, capital de Ucrania. Verán entonces que el programa señala que no puede realizar la búsqueda por no estar conectado a Internet. “¿Por qué en este caso no pudimos ver el mapa? ¿Los anteriores estaban guardados en nuestros teléfonos y este no? ¿Por qué? ¿Qué pasa con Kiev?”. Prestamos atención a las observaciones de los estudiantes y guiamos la discusión para concluir que los primeros mapas se habían descargado antes de deshabilitar la conexión a Internet y habían sido almacenados en sus dispositivos. Contrariamente, el mapa de Kiev no había sido descargado previamente y, al no poder bajarlo de Internet, la aplicación no pudo mostrarlo. “¿Por qué unos sí y otros no? ¿Cuál es el criterio?”. Escuchamos las reflexiones de los estudiantes y continuamos: “El tiempo que le demanda a nuestros dispositivos acceder a la información que tienen almacenada localmente –es decir, en el almacenamiento interno–, es mucho menor que el que les requiere bajar la misma información de Internet. Es por eso que los datos que se usan con mucha frecuencia son conservados en el teléfono. Esto permite que la velocidad de respuesta, en numerosos casos, sea mucho mayor”.



El dispositivo no puede acceder a información no descargada previamente

Continuamos: "Pero, entonces, ¿por qué mejor no tener descargados en el teléfono todos los mapas del mundo, así podemos ver todos sin necesidad de conectarnos a Internet?". El tamaño del medio de almacenamiento de los teléfonos tiene una capacidad limitada. Actualmente, suelen tener no menos de 8 GB y, en algunos casos, llegan a 256 GB. En cualquier caso, esto no es suficiente para almacenar todos los mapas. Además, en el almacenamiento interno no solo se conservan los datos a los que se accede frecuentemente, sino que allí también están guardados los programas que tenemos instalados, las fotos que sacamos, etc. Solo una pequeña porción, que se denomina **caché**, está reservada para este propósito.

Retomamos entonces la analogía presentada al comenzar la actividad: "Al comenzar la clase notamos que sobre el escritorio de nuestra habitación teníamos los materiales de Matemática, porque eran los que usábamos para prepararnos para el inminente examen de esa materia. Además, si hubiésemos querido consultar algo de Lengua, hubiéramos tenido que pararnos, ir hasta la biblioteca, revisar los estantes, retirar el material buscado y volver al escritorio para leerlo. Esto, evidentemente, demanda más tiempo que buscar lo que tenemos a mano sobre nuestro escritorio. Por otro lado, tampoco podríamos tener los materiales de todas las materias sobre el escritorio, porque no es suficientemente grande. El escritorio, en este caso, hace las veces de **caché**: un lugar con tamaño acotado en el que conservamos lo que requerimos con mayor frecuencia para poder acceder a ello muy rápidamente".

Proseguimos luego: "También mencionamos que, posiblemente, sobre el escritorio habrían quedado algunos libros de Historia porque, hasta hace pocos días, requerimos ese material para estudiar y luego no nos encargamos de ordenarlo meticulosamente en la biblioteca. Además, una vez que nos hubiésemos sacado de encima la prueba de Matemática, tendríamos que ponernos a estudiar para la de Lengua y la de Química. A esa altura, lo más conveniente sería pasar a la biblioteca lo dispuesto hasta entonces sobre el escritorio y colocar allí los materiales de las materias que aún no rendimos. Esto quiere decir que aquello que consultamos con frecuencia no es siempre lo mismo, sino que varía con el tiempo. ¿No pasa algo parecido con las aplicaciones que usamos? ¿Qué pasa si nos mudamos? ¿Y si nos vamos de viaje a otra ciudad?". También en este caso irían cambiando los mapas a los que accedemos con mayor frecuencia. "La información que más frecuentemente consultamos es dinámica, va cambiando con el tiempo. Los teléfonos van llevando un registro de la información a la que se accede y tienen un política para determinar qué es lo que, en cada momento, conviene tener almacenado en la **caché**".



Kiev, Ucrania

## Implementación de *caching*

Comenzamos la segunda parte de la actividad contándoles a los estudiantes: “Hasta ahora hemos ejemplificado el mecanismo de *caching* tomando como referencia el tiempo que le demanda a un teléfono acceder a información guardada en el almacenamiento interno en contraste con el que le toma bajar la misma información de Internet, guardada en una computadora que no sabemos dónde se encuentra. Sin embargo, la misma estrategia puede aplicarse en otros niveles, considerando otros dispositivos de memoria. El acceso a la memoria RAM es muchísimo más veloz que el acceso a un disco rígido de una computadora. Si un programa está requiriendo muchas veces un archivo guardado en el disco, posiblemente convenga copiarlo a la RAM para poder acceder a él más rápidamente. Otro ejemplo: hoy en día es habitual que existan módulos intermedios de memoria entre el procesador y la RAM, habitualmente llamados *memoria caché*, a los que se accede considerablemente más rápido que a la RAM. También estos se usan para almacenar información que se emplea con mucha frecuencia.”

Por último, los procesadores tienen pequeñas memorias internas llamadas *registros*, cuyo acceso es casi inmediato, y se utilizan (entre otras cosas) con este mismo propósito.<sup>1</sup> Los componentes más veloces son electrónicamente más complejos, además de ser mucho más caros”. Copiamos a continuación el siguiente gráfico de jerarquía de memorias en el pizarrón y seguimos: “Lo importante es entender que, en cualquier caso, se trata de acelerar los tiempos de acceso a información de uso frecuente, manteniéndola en un dispositivo de memoria mucho más veloz que aquel en el que originalmente se encontraba”.

	Velocidad	Capacidad
Registros		
Memoria caché		
Memoria RAM		
Discos		

Jerarquía de dispositivos de memoria

Continuamos comentando: “Caché es una palabra francesa que significa ‘oculto’. No es casual que se haya escogido este nombre: la *caché* funciona como una memoria escondida entre el procesador y un dispositivo de memoria más lento. Por ejemplo, supongamos que un programa requiere ciertos datos almacenados en la RAM. Antes de que efectivamente se realice el pedido a la memoria RAM, la memoria *caché* intercepta el pedido y chequea si tiene “cacheados”<sup>2</sup> los datos solicitados. Si así fuere, se los manda directamente al procesador; en caso contrario, el requerimiento continúa su curso hasta la RAM”.

<sup>1</sup>Las computadoras actuales suelen tener no más de 32 registros de 32 bytes cada uno.

<sup>2</sup>Jerga utilizada en computación.

Les repartimos entonces la ficha a los estudiantes y les indicamos que resuelvan la consigna. Allí encontrarán una serie de viñetas que muestran los posibles caminos que recorren los pedidos de datos por parte del procesador a la memoria cuando se cuenta con un mecanismo de *caching*. Ellos deben indicar qué está sucediendo en cada instancia, desde que el procesador realiza el pedido, hasta que obtiene la información.

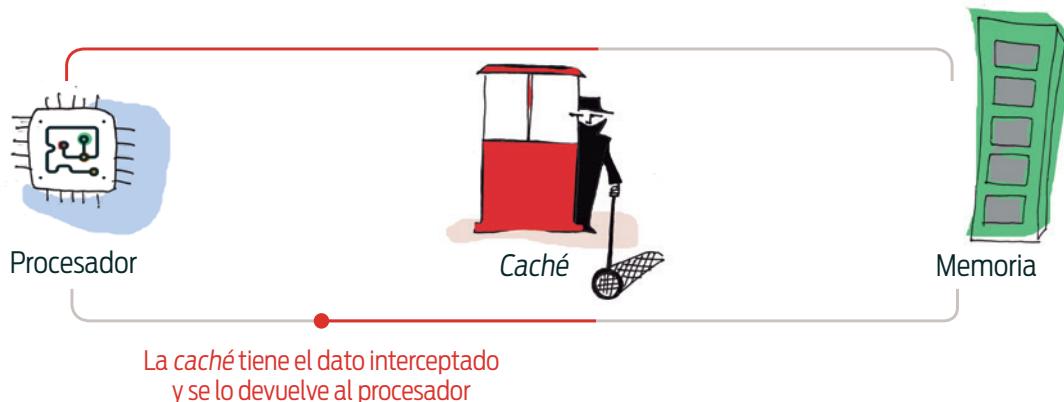
Lo primero que ocurre es que el procesador, mientras ejecuta las instrucciones de un programa, requiere datos que se encuentran almacenados en la memoria.



Luego, antes de que el pedido llegue a la memoria, es interceptado por la caché.



En esta instancia pueden suceder dos cosas: o bien el dato está en la *caché* o bien no lo está. En el primer caso, es devuelto directamente a la computadora sin que el requerimiento llegue a la memoria.



La caché tiene el dato solicitado y se lo devuelve al procesador

En caso de que el dato no se encuentre en la *caché*, el requerimiento es enviado a la memoria y, a continuación, el dato recorrerá el camino inverso hasta llegar al procesador.





Se solicita el dato a la memoria y este viaja hasta el procesador

Una vez que todos hayan completado la consigna, hacemos una puesta en común y continuamos: “¿Qué les parece que pasa cuando el dato de la memoria llega a la *caché*? ¿Debería la *caché* mantenerlo con el fin de tenerlo a mano para resolver futuros requerimientos?”. Si hay lugar disponible, el dato se copia en la *caché*. “¿Y si no hay espacio disponible?”. Lo que suceda en este caso dependerá de la política de *caching* que se use. Hay algunas implementaciones en las que siempre se conserva el último dato solicitado. En ese caso, deberá seleccionarse qué borrar de la *caché* para poder guardar el nuevo dato (posiblemente, lo menos solicitado en un cierto período de tiempo). En otras implementaciones, conservarlo o no puede depender de otros factores, como por ejemplo cuántas veces se lo solicitó en un cierto lapso de tiempo.

Finalmente preguntamos: “¿Y qué sucede cuando el procesador requiere escribir (en lugar de leer) un dato en la memoria de la computadora? ¿Además de conservarse en la *caché* (si es que esa fuese la política de *caching*), tendría que escribirse en la memoria?”. Escuchamos las respuestas y concluimos: “Siempre es necesario escribirlo en la memoria. Si así no se hiciese y, tiempo después, ese dato fuese borrado de la *caché* para escribir otro, posteriores requerimientos de ese dato que haga el procesador llegarían a la memoria, y el valor devuelto al procesador se encontraría desactualizado”.

### CIERRE

Para cerrar la actividad reforzamos la idea de que las estrategias de *caching* pueden aplicarse entre dos dispositivos de almacenamiento cualesquiera: uno de menor tamaño al que se accede a mayor velocidad y uno de mayor tamaño cuyo acceso demanda más tiempo. Por ejemplo, los navegadores de Internet hacen *caching* en los discos de las computadoras, el procesador utiliza registros internos para realizar menos accesos a la memoria RAM, etc. Finalmente, concluimos que, al borrar la *caché*, no estamos perdiendo la posibilidad de acceder a los datos sino que, simplemente, futuros accesos demandarán más tiempo.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# LA CACHÉ

¿Alguna vez tu teléfono inteligente te sugirió "borrar la caché"? Todo muy lindo, pero... ¿qué es la caché?

- 1.** Frecuentemente utilizamos aplicaciones de mapas en los teléfonos inteligentes. Cuando buscamos algún lugar, ¿cómo hace el teléfono para mostrarnos el mapa? ¿De dónde saca la información?

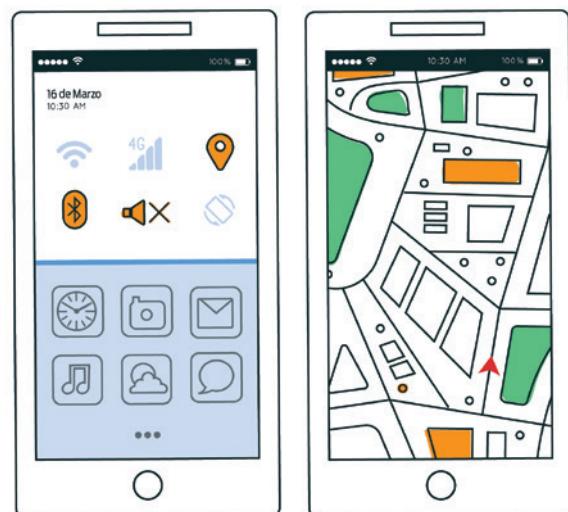


- a.** Deshabilitá de tu teléfono la conexión wifi y el acceso a la red de datos de telefonía celular. Después abrí la aplicación de mapas que uses habitualmente y buscá la dirección de tu casa. ¿Apareció el mapa en la pantalla? ¿Dónde están guardados esos mapas?

---

---

---



- b.** Ahora buscá una ciudad distante, que nunca hayas visitado. ¿Apareció el mapa en la pantalla? ¿Cuál es la diferencia de esta búsqueda con la del punto anterior? ¿Dónde está guardado el mapa de esta ciudad?

---

---

---

## CACHING

La idea de *caching* es que, entre el procesador y un dispositivo de memoria, haya otra memoria más pequeña a la que se pueda acceder a mucha mayor velocidad. En ella se conservan los datos más frecuentemente solicitados por los programas y, de esta manera, se reduce significativamente el tiempo que espera el procesador entre que pide esos datos y los obtiene. Algunos ejemplos son un disco para algo almacenado en la nube, la memoria RAM para algo almacenado en el disco, etc.



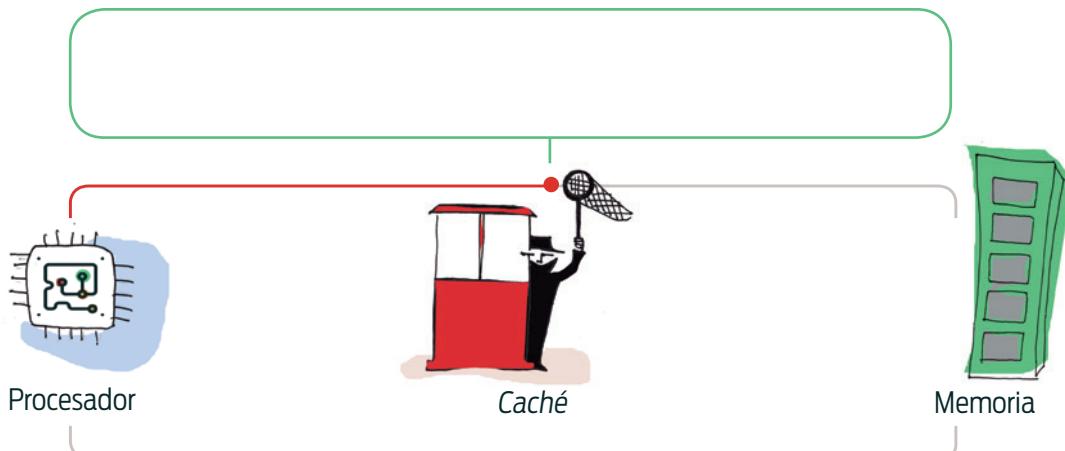
NOMBRE Y APELLIDO:

CURSO:

FECHA:

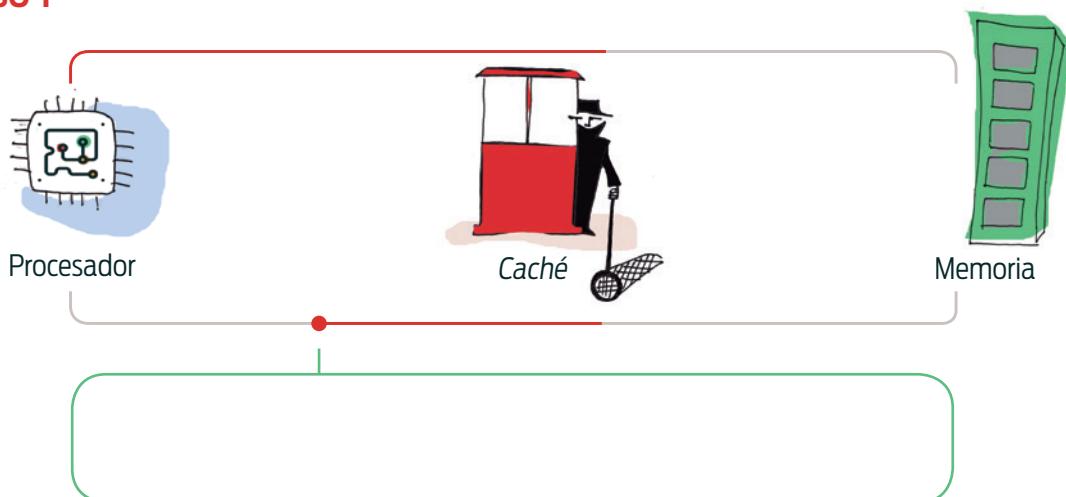
2. Cuando un programa solicita un dato, pueden suceder dos cosas: o bien el dato está en la caché o bien no lo está.

- a. Completá en el siguiente esquema lo que va sucediendo desde que el procesador requiere un dato de la memoria hasta que lo obtiene.



En esta instancia pueden suceder dos cosas: o bien el dato está en la caché o bien no lo está.

### Caso 1



NOMBRE Y APELLIDO:

CURSO:

FECHA:

## Caso 2



NOMBRE Y APELLIDO:

CURSO:

FECHA:

**b.** Cuando un programa escribe un dato que es interceptado y almacenado en la caché, ¿debe también copiarse en la memoria? ¿Por qué?

---

---

---

---

---

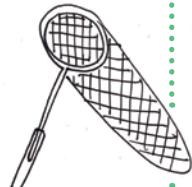
#### ¿LA PEGA O LA PIFIA?

Cuando un dato solicitado se encuentra en la caché se dice que la caché la pegó (o más comúnmente *caché hit*, por su forma en inglés). En cambio, cuando esto no sucede y hay que ir a buscarlo a la memoria, que la caché la pifió (o *caché miss*).



#### ¿SABÍAS QUÉ...

...los procesadores tienen (muy) pequeños módulos de memoria internos llamados *registros* a los que puede acceder a altísima velocidad? En la actualidad, un procesador no suele tener más de 32, cada uno de 32 bytes. ¡En ellos se "cachea" información de la memoria RAM!





## Secuencia Didáctica 3

# LA COMPUTADORA EN ACCIÓN

Escribimos programas usando diferentes lenguajes de programación. En general, nos valemos de lenguajes cuyo poder expresivo nos permite razonar en términos del problema que queremos resolver, sin prestar atención a aquello que hace el *hardware* de la computadora. ¿Cómo es que, efectivamente, el *hardware* lleva a cabo operaciones cuyo efecto es el que describimos en nuestros programas?

Las actividades de esta secuencia didáctica establecen el puente que existe entre los programas que escribimos y las operaciones que la computadora lleva a cabo. Siguiendo los ejercicios, los estudiantes descubrirán que sus programas necesitan transformaciones antes de poder ser ejecutados. Además, observarán qué es lo que efectivamente sucede cuando se ejecutan las instrucciones de los programas.

### ..... **OBJETIVOS**

- Observar cómo la computadora ejecuta programas en código de máquina.
  - Presentar la categorización de lenguajes de alto y bajo nivel.
  - Comprender las traducciones que hacen falta para que un programa pueda ejecutarse.
- .....

## Actividad 1

### Bajamos hasta el lenguaje de máquina



TODA LA CLASE

#### OBJETIVOS

- Conocer la compilación de programas.
- Diferenciar lenguajes compilados y lenguajes interpretados.
- Distinguir entre lenguajes de alto y bajo nivel.

#### MATERIALES



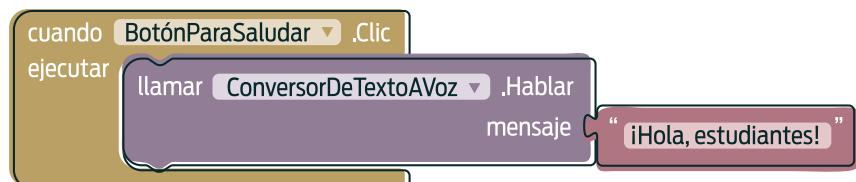
Anexo para estudiantes

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes comprendan que, para que una computadora pueda ejecutar los programas que construimos, previamente estos deben traducirse a un lenguaje que pueda ser interpretado por una máquina.

Comenzamos la clase diciéndoles a los estudiantes: “*Hello, students!*”. Observamos las reacciones y a continuación preguntamos: “¿Entendieron lo que les dije recién? Y entonces, ¿por qué no me contestaron?”. En caso de que no comprendan el idioma, les aclaramos que lo que acabamos de hacer es saludarlos en inglés: “Les dije *iHola, estudiantes!* en inglés. ¿Saben cómo se dice en francés? *Bonjour les étudiants!* En italiano sería *Ciao, studenti!* Si bien recién hablé en cuatro idiomas diferentes, ¿qué tienen en común todas las frases?”. Escuchamos sus respuestas y concluimos que todas las frases, si bien expresadas en lenguajes distintos, tienen el mismo significado. Es decir, son semánticamente equivalentes.

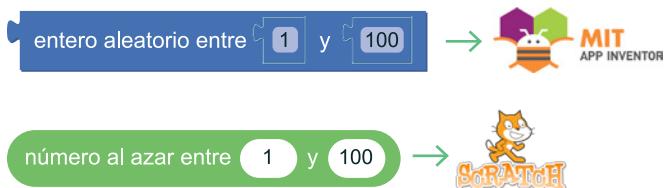
A continuación, copiamos en el pizarrón el fragmento de un programa escrito en el lenguaje de bloques App Inventor. Continuamos: “Hemos construido distintos programas encastrando bloques, ¿recuerdan? ¿Qué les parece que hará el programa que observamos en el pizarrón?”. Escuchamos sus respuestas y concluimos que las instrucciones indican que, cuando un cierto botón se presione, el teléfono dirá: “*iHola, estudiantes!*”.



Fragmento de un programa en un lenguaje de bloques

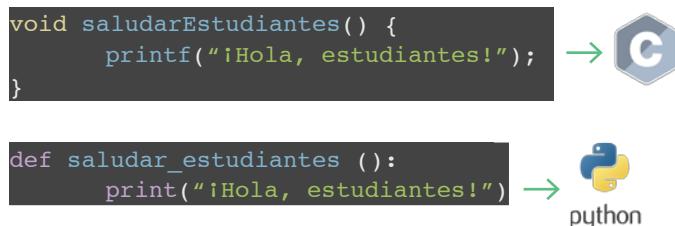
Continuamos: “También hemos aprendido que una máquina, al ejecutar un programa, solo realiza simples operaciones aritméticas y lógicas. ¿Cómo hace, entonces, para ejecutar un programa construido por nosotros?”. Alentamos a los estudiantes a que expresen sus ideas y guiamos el intercambio para concluir que, para que una máquina pueda ejecutar el programa, es necesario traducir el programa a un lenguaje que la máquina sea capaz de interpretar.

A continuación copiamos en el pizarrón los bloques que se muestran en la figura. Luego les contamos: “Existen muchos lenguajes basados en bloques, casi todos ellos diseñados para enseñar a programar. Por ejemplo App Inventor y Scratch. Aun cuando en todos haya bloques que sirven para hacer lo mismo, suelen tener distinto aspecto e incluso llamarse de diferente modo. Aquí pueden ver los bloques que se usan para obtener un número al azar, en App Inventor y en Scratch”.



Bloques de App Inventor y Scratch para obtener un número al azar

Luego, copiamos los dos fragmentos de los dos programas que se muestran en la figura que sigue, el primero escrito en C y el segundo en Python. Comentamos: “Además de los lenguajes basados en bloques, hay otra familia de lenguajes que se denominan *lenguajes textuales*. Para construir programas con ellos, en lugar de encastrar bloques hay que escribir las instrucciones como texto. Algunos ejemplos son C y Python. Si bien permiten hacer lo mismo, cada uno tiene una sintaxis propia.”



Fragmentos de programas en C y en Python

A continuación les explicamos: “Si bien son distintos, todos estos lenguajes tienen en común que, para ejecutar programas construidos en ellos, estos deben traducirse a un lenguaje que una máquina sea capaz de interpretar. El proceso de traducción no es igual en todos: hay uno para los llamados *lenguajes compilados* y otro para los llamados *lenguajes interpretados*.

### Lenguajes compilados

Comentamos: “Uno de los lenguajes compilados más difundidos se llama C. En el caso de estos lenguajes, el proceso de traducción lo lleva a cabo un programa llamado **compilador**, que tiene como entrada un programa expresado en un lenguaje de este tipo y generan como salida otro programa, semánticamente equivalente, que se corresponde con las acciones que puede realizar el *hardware* de la computadora. Por supuesto, cada lenguaje requiere un compilador distinto porque el lenguaje de origen es diferente, del mismo modo que para traducir de inglés a español es necesario un traductor diferente al que hace falta para traducir de francés a español.”.

Luego, copiamos en el pizarrón el siguiente fragmento de un programa escrito en **lenguaje ensamblador** y les decimos: "Esto es una parte de un programa. ¿Entienden qué hace?". Es probable que nadie pueda comprender su significado. "Lo que ven es una porción muy pequeña del resultado de una compilación. La salida que genera un compilador es un programa en lenguaje ensamblador (más comúnmente llamado *assembler*) de una computadora particular. Se trata de un lenguaje cuyas instrucciones se corresponden con lo que los componentes electrónicos del *hardware* de la computadora pueden realizar. Aquí hay, por ejemplo, instrucciones para sumar y comparar números, realizar operaciones lógicas y copiar datos de un lado a otro. Como podrán imaginar, escribir programas directamente en un *lenguaje ensambladores* muy difícil!"

```

movl $0xFF001122, %eax
addl %ecx, %edx
xorl %esi, %esi
pushl %ebx
movl 4(%esp), %ebx
leal (%eax, %ecx, 2), %esi
cmpl %eax, %ebx
jnae foo
retl

```

Fragmento de un programa en un lenguaje ensamblador

Seguimos: "Además, como solo cuentan con pocas operaciones muy rudimentarias, los programas contienen muchísimas más instrucciones que las que usamos cuando programamos nosotros. En los lenguajes compilados que se usan habitualmente, para conseguir que un teléfono diga una frase alcanza con escribir unas pocas líneas, mientras que en *assembler* serían literalmente miles de instrucciones!".

Les comentamos: "Los distintos procesadores tienen circuitos electrónicos diferentes. Por lo tanto, las operaciones que puede hacer cada uno son distintas. Esto quiere decir que cada modelo de procesador tiene su propio conjunto de instrucciones y, en consecuencia, su propio lenguaje ensamblador".

Continuamos: "Ahora bien, ya hemos visto que, al ejecutarse, las instrucciones de los programas se encuentran en la memoria.<sup>1</sup> También hemos estudiado que la memoria es un componente con una larga tira de celdas en cada una de las cuales hay o bien un cero o bien un uno.<sup>2</sup> Sin embargo, hasta aquí seguimos viendo las instrucciones del programa como texto. ¿Qué está faltando, entonces, para que un programa pueda cargarse en la memoria para ejecutarse?".

Resta, por último, traducir el programa del lenguaje ensamblador a un código binario, en el que los números resultantes codifican instrucciones que puedan ser interpretadas, ahora sí, por el *hardware* de la máquina. Nuevamente, el código resultante debe ser semánticamente equivalente al programa original. Esta traducción la realiza un programa llamado **ensamblador**, que toma como entrada un programa en lenguaje ensamblador y genera como salida un programa en **lenguaje de máquina**. Este, sí, puede cargarse en la memoria y, al leerlo, el procesador será capaz de ejecutar las instrucciones codificadas.

<sup>1</sup>Ver actividad "¿Qué son las computadoras?" de este capítulo.

<sup>2</sup>Ver actividad "¿Cuánto cabe en la memoria?" de este capítulo.

1	0	1	1	1	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1
1	1	0	0	1	0	1	0	0	0	1	1	0	0	0	1

Fragmento de un programa en lenguaje de máquina

Para anclar la idea de lo que hacen tanto los compiladores como los ensambladores presentamos la siguiente analogía: “En los lenguajes compilados, tanto el compilador como el ensamblador traducen programas de un lenguaje a otro, de forma similar a como lo hace un traductor de libros. Un traductor recibe un libro (o un programa) en un idioma y lo reescribe en otro”.

### Lenguajes interpretados

Les contamos a los estudiantes: “En la actualidad, los lenguajes más utilizados (como App Inventor o Python) son lenguajes interpretados. A diferencia de los lenguajes compilados, los programas que construimos nosotros son ejecutados por máquinas que no son físicas, son virtuales! ¿Se imaginan qué quiere decir esto?”. Escuchamos sus consideraciones y continuamos: “Aquí, las máquinas son programas!”.

Continuamos: “Una máquina virtual es un programa que emula la ejecución de otros programas. Al ejecutar una máquina virtual se le indica cuál es el programa que queremos que emule. Hasta ahora, nosotros, hemos construido programas en App Inventor, que fueron emulados (o ejecutados) por la máquina virtual de App Inventor. ¡Ya sé, parece un trabalenguas!”.

En el caso de los lenguajes interpretados, lo que realmente se carga en la memoria y es ejecutado por el *hardware* de la computadora es el programa de la máquina virtual. Les contamos: “Las máquinas virtuales, que nos son provistas por quienes diseñan y construyen estos lenguajes de programación, son programadas usando lenguajes compilados. De otro modo, sería imposible cargarlas en la memoria de la computadora para que la unidad central de procesamiento ejecute sus instrucciones”.

### CIERRE

Les entregamos el anexo a los estudiantes y les comentamos que los lenguajes con los que habitualmente programamos se conocen como **lenguajes de alto nivel**. Ellos nos permiten abstraernos de las operaciones que internamente llevan a cabo las máquinas y concentrarnos en el problema que buscamos resolver. Contrariamente, los **lenguajes de bajo nivel** proveen instrucciones que se relacionan directamente con lo que las máquinas pueden realizar.

# BAJAMOS HASTA EL LENGUAJE DE MÁQUINA

Cada modelo de computadora tiene componentes electrónicos que son capaces de llevar a cabo operaciones muy simples. Por ejemplo, operaciones aritméticas (como sumar y multiplicar), operaciones lógicas, leer un dato en la memoria, escribir un dato en la memoria, etc. Al programar, si por suerte no nos hace falta pensar en esto! En su lugar, usamos lenguajes que son mucho más expresivos, llamados **lenguajes de alto nivel**, que nos permiten razonar en términos del problema que queremos resolver.

Ahora bien, ¿cómo se ejecutan estos programas si la computadora hace tan pocas y rudimentarias operaciones?

## COMPILEADOR

Un **compilador** es un programa que toma como entrada un programa escrito en un lenguaje de programación de alto nivel y genera como salida otro programa en lenguaje ensamblador que es semánticamente equivalente; es decir, que hace exactamente lo mismo.

## ENSAMBLADOR

Un programa en lenguaje ensamblador sigue siendo un archivo con texto. Para ejecutar un programa, lo primero que hay que hacer es cargarlo en la memoria, de donde el procesador irá leyendo sus instrucciones una por una. Sin embargo, en la memoria solo se guardan bits. El **ensamblador** es un programa que toma como entrada un programa en lenguaje ensamblador y genera como salida otro equivalente en **lenguaje de máquina**.

## COMPILEADOR

```
movl $0xFF001122, %eax
addl %ecx, %eax
xori %esi, %esi
pushl %ebx
movl 4(%esp), %ebx
leal (%eax,%ecx,2), %esi
cmpl %eax, %ebx
jnae foo
retl
```

## ENSAMBLADOR

1	0	1	1	1
0	0	0	1	0
1	1	1	1	1
1	1	0	0	1

## LEN GUJE DE MÁQUINA

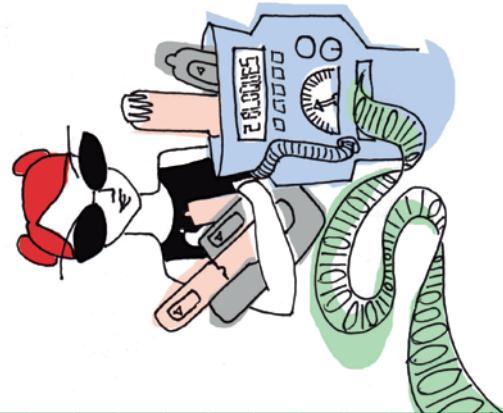
El **lenguaje de máquina** está formado por una serie de instrucciones codificadas con números binarios que pueden tanto cargarse en la memoria como ser interpretadas por el hardware de la computadora.

## LEN GUJE ENSAMBLADOR

Cada computadora tiene componentes que realizan algunas operaciones simples: aritméticas, lógicas, de lectura y escritura en la memoria, y algunas más. El **lenguaje ensamblador** de una computadora tiene instrucciones que se corresponden con lo que sus componentes de hardware son capaces de llevar a cabo.

## LEN GUJE DE ALTO NIVEL

Habitualmente, al construir un programa se utiliza un **lenguaje de alto nivel**. Estos lenguajes permiten que un programador rase sobre el problema que quiere resolver, abstrayendo por completo el funcionamiento interno de los componentes de hardware.



## Actividad 2

### Ejecución de programas

 DE A DOS

#### OBJETIVOS

- Reproducir la ejecución de un programa en lenguaje de máquina.
- Identificar los componentes internos del procesador.
- Conocer el ciclo de instrucción.

#### MATERIALES

-  Anexo de la actividad
-  Ficha para estudiantes

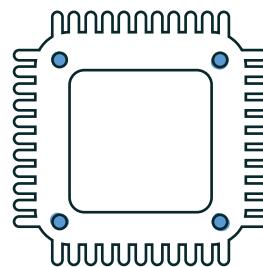
#### DESARROLLO

El objetivo de esta actividad es que los estudiantes comprendan cómo hace una computadora para ejecutar un programa. Por tratarse de una actividad extensa, se encuentra dividida en tres partes. En la primera se hace foco en los subcomponentes de la unidad central de procesamiento que intervienen en la ejecución de las instrucciones de un programa y se presenta el **ciclo de instrucción**; en la segunda, se muestra cómo un programa se almacena en la memoria y se introduce el **puntero de instrucción**; finalmente, en la tercera, se propone un lenguaje ensamblador ficticio y se muestra paso a paso la ejecución de un programa.

#### Los componentes de la CPU y el ciclo de instrucción

Comenzamos comentando: “Como ya hemos visto, para que un programa que construimos pueda ejecutarse, en primer lugar se compila y se obtiene un programa semánticamente equivalente en lenguaje ensamblador, es decir, con instrucciones que se corresponden con las operaciones que los circuitos del *hardware* de la máquina pueden realizar; en segundo lugar, se ensambla el programa para obtener el código de máquina –instrucciones codificadas en un sistema binario que pueden almacenarse en la memoria de la computadora–”.

Preguntamos luego: “¿Recuerdan cuál es la función de la unidad central de procesamiento?”. Guiamos el intercambio para concluir que su función principal es ejecutar una por una las instrucciones de los programas. Comentamos: “Las operaciones que realiza, que son las descritas por las instrucciones del lenguaje ensamblador, son operaciones muy simples, como sumar números, leer datos de la memoria, etc.”. Dibujamos, entonces, un esquema del procesador, al que le iremos incorporando elementos en el transcurso de la actividad.

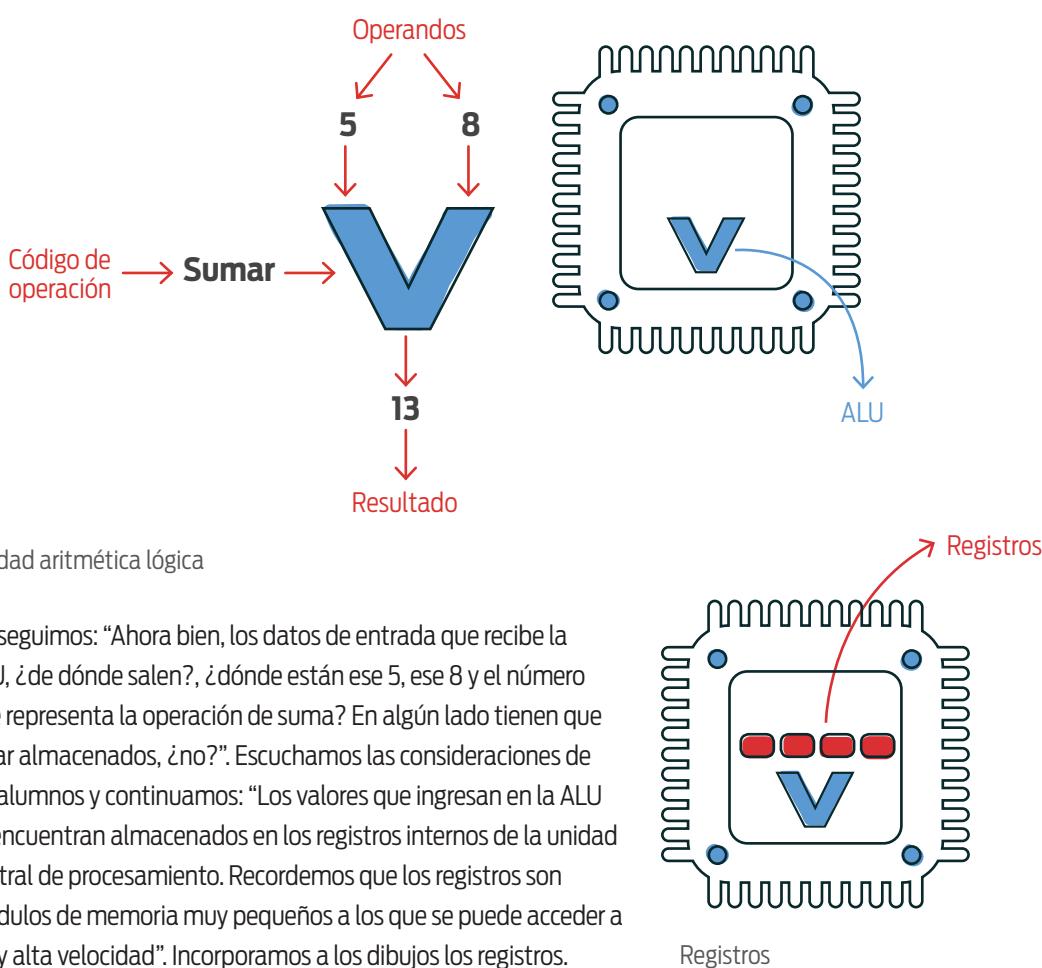


Esquema de una unidad central de procesamiento

Continuamos: “Ahora bien, ¿cómo hacen ustedes cuando quieren realizar una operación aritmética? Por ejemplo, si quisieran multiplicar dos números.”. Es esperable que algún estudiante responda que usaría una calculadora, como las que traen instaladas los teléfonos inteligentes. “Supongamos que una instrucción de un programa indica multiplicar dos valores, ¿saben cómo hace una computadora para llevar a cabo la operación? Los procesadores también tienen una especie de calculadora interna, cuyos circuitos le permiten sumar, multiplicar, etc. Este componente se llama **unidad aritmética lógica o ALU**.<sup>1</sup>

<sup>1</sup> Acrónimo en inglés de *Arithmetic Logic Unit*.

En esencia, una ALU tiene tres canales de entrada y uno de salida. Por uno de los canales de entrada recibe el **código de operación** del cómputo que se requiere realizar; por ejemplo, un código que denota la operación de suma de dos números. Los otros dos canales de entrada son para los **operандos** de la operación en cuestión: siguiendo con el ejemplo, los dos números que se quieren sumar, que podrían ser 5 y 8. El canal de salida, por su parte, es por donde sale el **resultado** de la operación (13, en el ejemplo).<sup>1</sup> Dibujamos en el pizarrón el esquema de la ALU y, también, la incorporamos dentro del dibujo de la CPU.

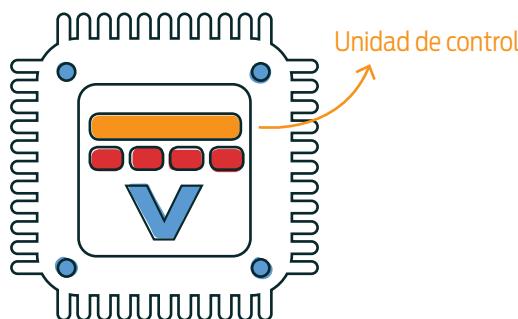


Seguimos luego el intercambio con los estudiantes: “¿Cómo les parece que se llena el contenido de los registros, que finalmente ingresa a la ALU? Me refiero tanto al que contiene el código de operación como los de los operandos. Siguiendo con el ejemplo, ¿quién escribe el código de operación de una suma en un registro, el número 5 en otro y el 8 en otro?”. En caso de que no haya respuestas, preguntamos: “Mientras

<sup>1</sup>Las ALU incluyen más canales, que, en general, se usan para comunicar el estatus que resulta de una operación llevada a cabo. Este podría indicar, por ejemplo, que todo salió según lo esperado; o que el resultado de una operación aritmética es mayor que los números que puede representar la computadora, lo que indica un error (entre muchos otros ejemplos). Sin embargo, a efectos de que la explicación resulte clara, obviamos mencionar estos canales.

un programa se ejecuta, ¿dónde están almacenadas sus instrucciones (por ejemplo, la instrucción de sumar dos números) y los datos (en el ejemplo, los valores 5 y 8)?”.

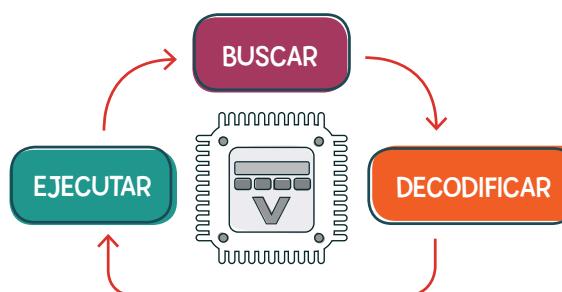
Escuchamos sus reflexiones y continuamos: “Como ya hemos estudiado, mientras se ejecuta un programa, sus instrucciones están almacenadas en la memoria principal, al igual que los datos con los que trabaja. Hay un componente interno de la unidad central de procesamiento que se llama **unidad de control**, cuya función es coordinar el funcionamiento de la CPU. En esencia, lo que hace es (i) ir a **buscar** a la memoria la siguiente instrucción del programa que debe ejecutarse; (ii) **decodificar** de qué instrucción se trata (por ejemplo, sumar dos números); y (iii) establecer las condiciones para **ejecutar** la instrucción –siguiendo el ejemplo, escribir en el registro que corresponde el código de la operación *suma*, y mandar una señal a la ALU para que realice la operación”. El conjunto de estos tres pasos, que se realizan para ejecutar cada instrucción de un programa, se llama **ciclo de instrucción**. La unidad central de procesamiento repetirá ciclos de instrucción para ejecutar todas las instrucciones de un programa, hasta que este finalice.



Unidad de control

#### CICLO DE INSTRUCCIÓN

Un ciclo de instrucción comprende las tres etapas que lleva a cabo la unidad central de procesamiento para ejecutar una instrucción del lenguaje de máquina: (i) **buscar** la instrucción en la memoria; (ii) **decodificar** de qué instrucción se trata; y (iii) **ejecutar** la instrucción. Usualmente, a estas



etapas se las llama por sus nombres en inglés: *fetch*, *decode* y *execute*. La unidad central de procesamiento repite este ciclo para cada instrucción que tiene que realizarse mientras un programa se ejecuta. Es decir, lo repite hasta que finalice su ejecución.

Continuamos: “Recién exemplificamos el ciclo de instrucción considerando que la instrucción a ejecutar era una operación aritmética. Sin embargo, la instrucción podría ser de otra naturaleza, como leer el contenido de un dato almacenado en la memoria y escribirlo en un registro. En este caso, la etapa de ejecución no será realizada por la ALU, sino que intervendrán los circuitos que comunican la CPU con la memoria. En definitiva, lo que suceda durante la etapa de ejecución dependerá de la instrucción decodificada por la unidad de control”.

### Programas en memoria y puntero de instrucción

Comenzamos la segunda parte de la actividad preguntando a los estudiantes: “¿Recuerdan cuál es la organización más básica de la memoria?”. Escuchamos sus respuestas y, en caso de que no surja de alguno de ellos, les recordamos: “La memoria está organizada como una larga tira de bits que solemos representar con los dígitos 0 y 1”. Copiamos a continuación la siguiente tabla, que representa un fragmento de memoria.

→ Una secuencia de bits

1	0	1	1	1	0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
1	1	0	0	1	0	1	0	0	0	1	1	0	0	0	0	1

Representación de un fragmento de memoria como secuencia de bits

Continuamos: “En general, para referirnos al contenido de la memoria, los bits se agrupan de a 8 y, por lo tanto, se la concibe como una secuencia de bytes (como se recordará, 8 bits equivalen a un byte). El fragmento anterior, entonces, puede reescribirse del siguiente modo”. Copiamos a continuación la siguiente representación del mismo fragmento de la memoria, ahora como secuencia de bytes.

→ Un byte

184	34	17	0	255	1	202	49
-----	----	----	---	-----	---	-----	----

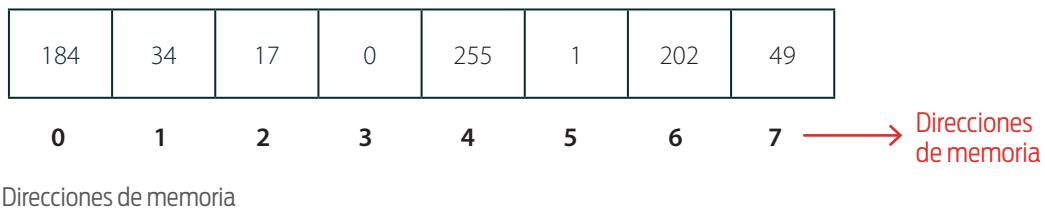
Una secuencia de bytes

Representación de un fragmento de memoria como secuencia de bytes

Les explicamos: “Lo que ahora ven en las celdas es una representación de la memoria como secuencia de bytes, en lugar de bits. El contenido de cada celda es la representación decimal del número que, en el dibujo previo, veíamos en un sistema binario”.

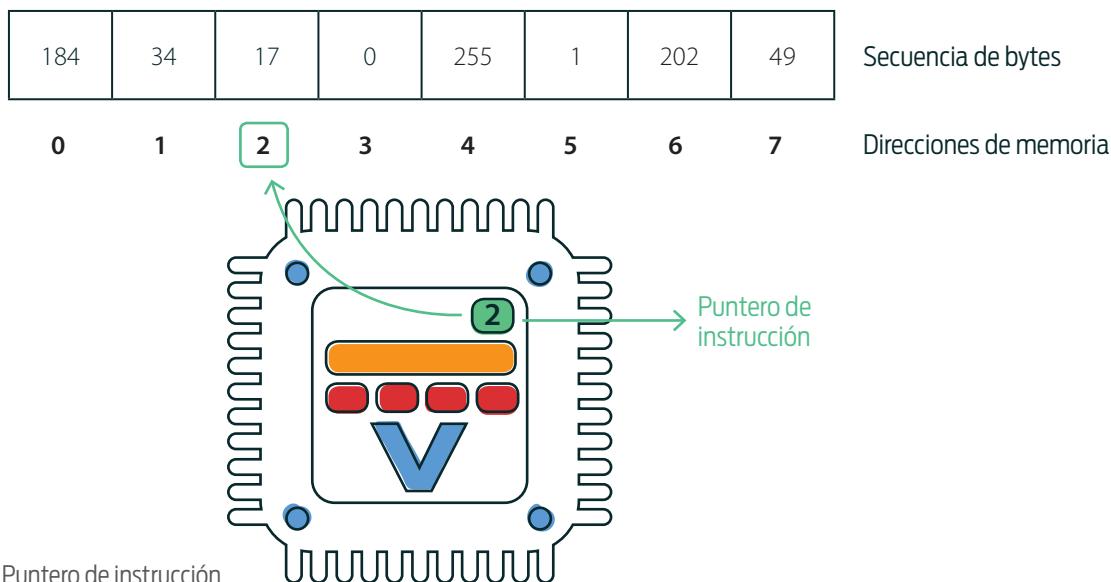
Continuamos: “Cada byte de la memoria puede identificarse con un número, que se corresponde con la ubicación en la que se encuentra, comenzando a contar desde 0. A este número se lo conoce como **dirección de memoria**. En este caso, en la dirección de memoria 0 hay un 184, en la 1 un 34, en la 2 un 17, y así

siguiendo. Cuando el procesador necesite un dato de la memoria, lo localizará usando estas direcciones". Entonces, agregamos al gráfico previo las direcciones de memoria.

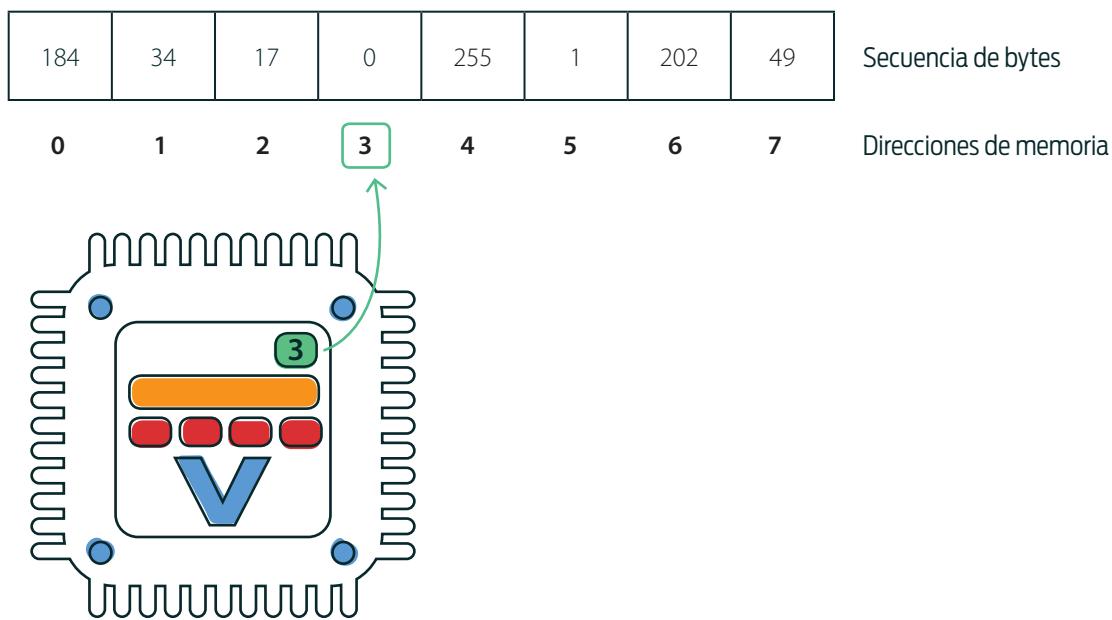


Luego de asegurarnos de que todos los estudiantes comprendieron la explicación, seguimos: "Ya sabemos que, para ejecutarse, un programa se traduce a lenguaje de máquina –es decir, a números que codifican instrucciones– y esos números se cargan en la memoria. Tranquilamente, entonces, lo que observamos podría tratarse de la representación de parte de un programa, en la que en cada byte hay una instrucción. Por ejemplo, el número 17 que está en la dirección de memoria 2 podría codificar la operación suma. Eventualmente, la CPU la buscará de la memoria (**buscar**), la unidad de control decodificará que se trata de sumar dos números (**decodificar**), y cargará en el registro que corresponde el código de la operación para que la ALU realice la suma (**ejecutar**), completando de ese modo un ciclo de instrucción".

Entonces preguntamos: "Ahora bien, hasta aquí siempre mencionamos que la CPU va a buscar a la memoria la próxima instrucción que tiene que ejecutarse, pero... ¿cómo sabe cuál es esa instrucción? Es decir, ¿de dónde saca la dirección de memoria de la próxima instrucción a ejecutar?". Guiamos el intercambio para arribar a la conclusión de que hay un registro interno de la unidad central de procesamiento que se llama **puntero de instrucción** (o *contador de programa*), que siempre tiene almacenada la dirección de memoria a la que hay que ir a buscar la próxima instrucción que tiene que ejecutarse. Este registro se utiliza únicamente para esto y no está disponible para que un programa escriba en él un valor o lea su contenido.



A continuación preguntamos: “¿Qué tiene que pasar con el puntero de instrucción al finalizar un ciclo de instrucción? ¿Qué valor debe contener?”. Antes de comenzar el siguiente ciclo de instrucción, la unidad de control actualiza el valor del puntero de instrucción para que contenga la dirección de la próxima instrucción del programa que hay que ir a buscar a la memoria. “Siguiendo con el ejemplo, luego de la suma hay que ejecutar la instrucción inmediatamente posterior del programa; es decir, la que se encuentra almacenada en la posición 3 de la memoria. Si bien en la mayoría de los casos la siguiente instrucción es la que está en la siguiente posición de la memoria, esto no siempre es así. Recuerden, por ejemplo, cómo usaron las sentencias condicionales en los programas que armaron: de acuerdo a si una condición era verdadera o falsa, se ejecutaban unas instrucciones u otras; no era siempre la misma. Como podrán imaginarse, al comenzar la ejecución de un programa, el valor del puntero de instrucción será la dirección de memoria en la que se encuentra la primera instrucción”.



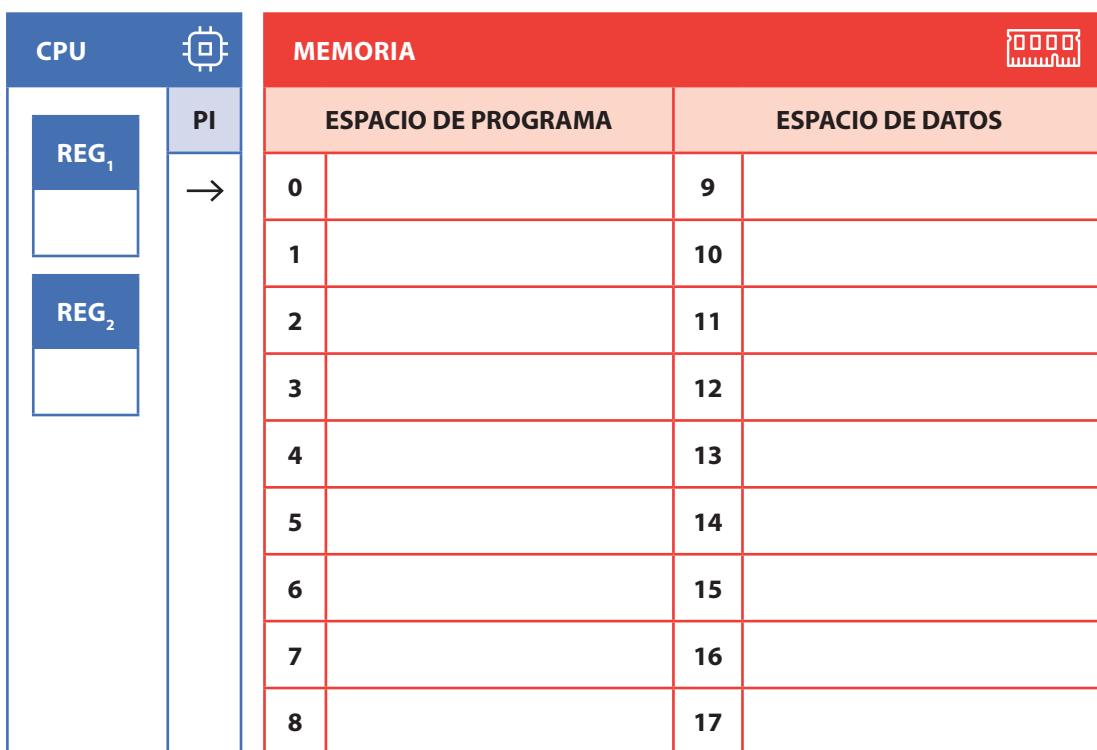
Actualización del puntero de instrucción

### Ejecución de un programa

Al comenzar la tercera y última parte de la actividad, les repartimos a los estudiantes el anexo y la ficha, les pedimos que se ubiquen en parejas y les comentamos: “Ahora veremos cómo es la ejecución de un programa. Vamos a suponer que tenemos una computadora ficticia, que se llama *Bet & Rob air*, que tiene una memoria de 18 bytes, de los cuales los primeros 9 están reservados para programas y los restantes se usan para almacenar datos.”.

Continuamos: “El procesador es muy sencillo: además del puntero de instrucción, tiene dos registros, **REG<sub>1</sub>** y **REG<sub>2</sub>**, que los programas pueden usar tanto para escribir como para leer valores. Su lenguaje ensamblador tiene instrucciones para: (i) establecer el contenido de un registro; (ii) incrementar y decrementar en 1 el contenido de un registro; (iii) copiar en un registro un dato contenido en la memoria y copiar en

la memoria un dato contenido en un registro; (iv) modificar el flujo de ejecución de instrucciones; y (v) finalizar la ejecución de un programa". Les indicamos, entonces, que lean la descripción del *hardware* de la computadora que se encuentra en la ficha, en la que se describe las instrucciones disponibles en el lenguaje que figuran a continuación.



Unidad central de procesamiento de la *Bet & Rob air*

Memoria de la *Bet & Rob air*

**ASIGNAR [r] [n]**: escribe en el registro *r* el número *n*.

**INCREMENTAR [r]**: incrementa en 1 el valor contenido en el registro *r*.

**DECREMENTAR [r]**: decrementa en 1 el valor contenido en el registro *r*.

**MOVER\_DE\_MEMORIA\_A\_REGISTRO [r] [n]**: escribe en el registro *r* el contenido de la celda de memoria cuya dirección es *n*.

**MOVER\_DE\_REGISTRO\_A\_MEMORIA [r] [n]**: lee el contenido del registro *r* y lo escribe en la posición *n* de la memoria.

**SALTO\_CONDICIONAL [r] [n] [i]**: lee el contenido del registro *r*, lo compara con el número *n* y, si son iguales, establece que la siguiente instrucción a ejecutar es la *i*-ésima (es decir, la que está en la posición *i* en el programa); si no, el flujo de ejecución continúa con la instrucción que se encuentra inmediatamente a continuación.

**SALTO\_INCONDICIONAL [i]**: establece que la siguiente instrucción a ejecutar es la *i*-ésima (es decir, la que está en la posición *i* en el programa).

**DETENER**: detiene la ejecución del programa.

A pesar de que, para que un programa se cargue en la memoria de una computadora, este debe estar en lenguaje de máquina (es decir, como una secuencia de números), en este caso escribiremos las instrucciones en lenguaje ensamblador, ya que resultará más claro. Además, hacerlo de este modo no representa un cambio conceptual importante: tanto las instrucciones en lenguaje de máquina como en lenguaje ensamblador se corresponden con lo que las piezas de *hardware* pueden realizar. La diferencia es que, en el lenguaje ensamblador, en lugar de codificarlas con números, las codifican con texto.

En la ficha se presentan tres programas para que los estudiantes analicen, con cada uno de ellos, cómo es la evolución del sistema a medida que se ejecutan –es decir, cómo se va modificando el contenido de la memoria y de los registros mientras se repite una y otra vez el ciclo de instrucción–. El primero, para que entiendan la dinámica, lo resolvemos entre todos. Como se verá, el programa multiplica un número por 2. Copiamos en el pizarrón, entonces, el estado del sistema antes de la ejecución de la primera instrucción. A la izquierda se encuentran los registros internos de la CPU: **REG<sub>1</sub>** y **REG<sub>2</sub>** y el puntero de instrucción. Para que resulte visualmente claro, en el caso del puntero de instrucción, en lugar escribir la dirección de memoria de la próxima instrucción que tiene que ejecutarse, dibujaremos una flecha que apunte a tal posición. Al comenzar, se encuentra apuntando a la dirección 0, que contiene la primera instrucción del programa. A la derecha se encuentra la memoria principal. El espacio reservado para el programa comprende las celdas entre las direcciones 0 y 8 inclusive; el espacio de datos, las de la 9 a la 17. Salvo en la celda cuya dirección es 9 (que tiene un 2), en todas las restantes celdas de datos hay un 0.



Estado del sistema antes de iniciar la ejecución del programa

En lo que sigue, iremos analizando junto a los estudiantes lo que sucede a medida que avanza la ejecución del programa. Cada vez que se modifique un valor de la memoria o uno de un registro, lo reflejaremos en el esquema del pizarrón.

Al comenzar la ejecución, la unidad central de procesamiento leerá de la memoria la primera instrucción que, tal como lo indica el puntero de instrucción, es la que está en la posición 0 de la memoria (**búsqueda**). Entonces, la unidad de control decodificará que se trata de escribir en **REG<sub>1</sub>** el contenido de la dirección 9 de la memoria (**decodificación**) y, en consecuencia, activará las señales correspondientes para que la operación se lleve a cabo (**ejecución**).<sup>1</sup> El contenido de **REG<sub>1</sub>**, entonces, pasará a ser 2. Una vez completada la operación, la unidad de control hará apuntar al puntero de instrucción a la siguiente posición de memoria.

<sup>1</sup> Si bien el ciclo de instrucción se repetirá al ejecutar cada instrucción del programa, en lo que sigue no nos detendremos para identificar cada una de sus tres etapas. En cambio, haremos foco en el efecto de las instrucciones sobre el estado del sistema.



Estado del sistema luego de ejecutar la primera instrucción

A continuación, la CPU leerá la instrucción de la posición 1 de la memoria. Una vez decodificada, se encargará de escribir un 0 en el registro REG<sub>2</sub> y, acto seguido, hará avanzar el puntero de instrucción a la siguiente posición de la memoria.



Estado del sistema luego de ejecutar la segunda instrucción

La siguiente instrucción es un salto condicional, lo que quiere decir que, en caso de que la condición evaluada sea verdadera, se altera el flujo de ejecución del programa. En este caso, si el contenido de **REG<sub>1</sub>** fuese 0, la siguiente instrucción a ejecutar sería la que está en la posición 7 de la memoria. Como el valor de **REG<sub>1</sub>** es 2, la condición es falsa y, en consecuencia, no se produce el salto. Por lo tanto, el único efecto de la instrucción es que el puntero de instrucción pasa a apuntar a la posición 3 de la memoria.



Estado del sistema luego de ejecutar la tercera instrucción

Como puede observarse, las instrucciones de las posiciones 3 y 4 incrementan en 1 el valor de **REG<sub>2</sub>** y la de la posición 5 decremente el de **REG<sub>1</sub>**. Se muestra a continuación el estado del sistema luego de la ejecución de estas tres instrucciones.



Estado del sistema luego de ejecutar la sexta instrucción

La instrucción siguiente es un salto incondicional que indica que, a continuación, debe ejecutarse la instrucción de la dirección 2 de la memoria.



Estado del sistema luego del salto incondicional

Nuevamente, tal como lo indica el puntero de instrucción, la siguiente instrucción a ejecutar es el salto condicional. Como el valor contenido en **REG<sub>1</sub>** es, otra vez, distinto de cero, la ejecución continuará con la instrucción de la posición 3 de la memoria. Además, como puede observarse, lo que sigue es, otra vez, incrementar dos veces el valor de **REG<sub>2</sub>**, decrementar en uno el valor de **REG<sub>1</sub>** y, con el salto incondicional de la dirección 6 de memoria, volver a establecer que la siguiente instrucción es la que está en la segunda posición:



Estado del sistema luego del segundo salto incondicional

En esta ocasión, la condición evaluada en el salto condicional es verdadera: el valor de **REG<sub>1</sub>** es 0. Por lo tanto, se establece que la siguiente instrucción que tiene que ejecutarse es la que está en la posición 7 de la memoria.



Estado del sistema luego del segundo salto incondicional

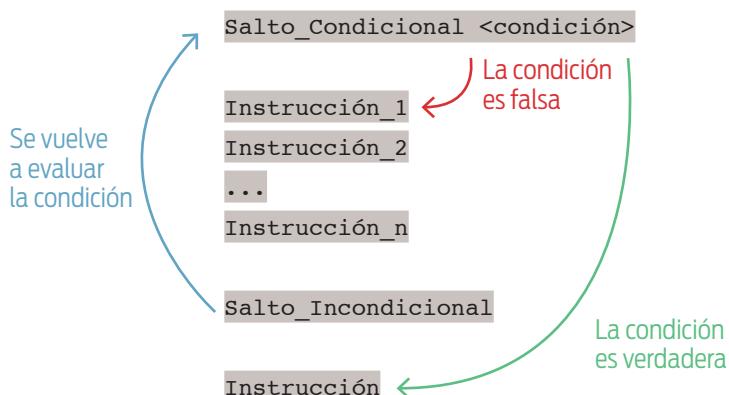
El efecto de la instrucción **MOVER\_DE\_REGISTRO\_A\_MEMORIA REG<sub>2</sub> 10** es que en la posición 10 de la memoria quede el valor contenido en el registro **REG<sub>2</sub>**; en este caso, un 4. Además, adelanta el puntero de instrucción a la siguiente posición de la memoria, lo que indica que la ejecución del programa finaliza.



Estado del sistema al finalizar la ejecución del programa

Una vez que hayamos finalizado de analizar la ejecución del programa, les decimos a los estudiantes: "Hemos visto cómo fue evolucionando, paso a paso, el estado del sistema al ejecutar el programa. ¿Qué sucedió con el contenido de la memoria?". Probablemente algún estudiante conteste que, salvo en la posición 10 en el que se escribió un 4, el resto permaneció inalterado. "Muy bien, pero ¿qué hubiese sucedido si antes de comenzar la ejecución en la posición 9 –la primera del espacio de datos– hubiese habido un 3 en lugar de un 2?". Les damos tiempo para que piensen al respecto.

Continuamos: "Es interesante, en primer lugar, observar que con un salto condicional y un salto incondicional se consigue el mismo efecto que con una repetición, como las que usamos al programar en un lenguaje de alto nivel. El primero chequea una cierta condición que, si no se cumple, provoca la ejecución de todas las instrucciones que haya hasta alcanzar el salto incondicional. Además, este último redirige el flujo de ejecución para que la condición se chequee nuevamente. Cuando finalmente la condición sea verdadera, se continúa con la instrucción inmediatamente posterior al salto incondicional, lo que provoca el fin de la repetición".



Repetición condicional usando saltos

En el programa analizado, el salto condicional es la instrucción de la posición 2 de la memoria y verifica si el contenido de REG<sub>1</sub> es igual a 0; el salto incondicional es la instrucción de la dirección 6; y las instrucciones que se repiten, las que están en el medio de ambas.

<b>0</b>	MOVER_DE_MEMORIA_A_REGISTRO REG <sub>1</sub> 9
<b>1</b>	ASIGNAR REG <sub>2</sub> 0
<b>2</b>	<b>SALTO_CONDICIONAL REG<sub>1</sub> 0 7</b>
<b>3</b>	<b>INCREMENTAR REG<sub>2</sub></b>
<b>4</b>	<b>INCREMENTAR REG<sub>2</sub></b>
<b>5</b>	<b>DECREMENTAR REG<sub>1</sub></b>
<b>6</b>	<b>SALTO_INCONDICIONAL 2</b>
<b>7</b>	MOVER_DE_REGISTRO_A_MEMORIA REG <sub>2</sub> 10
<b>8</b>	DETENER

Instrucciones que delimitan la repetición (en azul) e instrucciones que se repiten (en amarillo)

A continuación les decimos: “La primera instrucción del programa cargó en REG<sub>1</sub> el valor contenido en la dirección 9 de la memoria. En este caso fue 2, pero podría haber sido 3, 4, 5 o cualquier otro número, dependiendo del estado de la memoria al iniciar la ejecución del programa. Luego, la condición del salto condicional chequeó si ese valor era 0, pero como no era así, se ejecutaron las instrucciones que seguían hasta el salto incondicional. Ahora bien, ¿qué hacían las instrucciones que se repitieron con el contenido de REG<sub>1</sub>?”. La única que alteró el contenido de REG<sub>1</sub> es la de la posición 5, que decrementó en 1 su valor. “Entonces, ¿cuántas veces se repetirían estas instrucciones? O, lo que es lo mismo, ¿cuántas veces hubo que decrementar REG<sub>1</sub> para que se cumpla la condición del salto condicional? (es decir, que REG<sub>1</sub> llegue a 0)”. Tantas como el número que originalmente había en la posición 9 de la memoria. “En este caso fueron 2, pero si originalmente en la memoria hubiese habido un 4, habrían sido 4.”.

<b>0</b>	<b>MOVER_DE_MEMORIA_A_REGISTRO REG<sub>1</sub> 9</b>
<b>1</b>	ASIGNAR REG <sub>2</sub> 0
<b>2</b>	<b>SALTO_CONDICIONAL REG<sub>1</sub> 0 7</b>
<b>3</b>	INCREMENTAR REG <sub>2</sub>
<b>4</b>	INCREMENTAR REG <sub>2</sub>
<b>5</b>	<b>DECREMENTAR REG<sub>1</sub></b>
<b>6</b>	<b>SALTO_INCONDICIONAL 2</b>
<b>7</b>	MOVER_DE_REGISTRO_A_MEMORIA REG <sub>2</sub> 10
<b>8</b>	DETENER

Instrucciones en las que intervienen REG<sub>1</sub> (en rojo)

Continuamos: “La segunda instrucción que se ejecuta en el programa establece el contenido de REG<sub>2</sub> en 0. ¿Qué hacen las instrucciones que se repiten con el contenido de este registro?”. Lo incrementan dos veces. “Entonces, si las instrucciones se repiten tantas veces como el número contenido en la posición 9 de la memoria, y cada vez el registro se incrementa 2 veces, ¿cuál será su valor al terminar la repetición?”. El doble del número contenido originalmente en la posición 9 de la memoria. “¿Y qué sucede una vez que se interrumpe la repetición?”. Se copia el contenido de REG<sub>2</sub> en la posición 10 de la memoria. “Es decir, el programa multiplica por 2 un valor y escribe en la memoria el resultado”.

<b>0</b>	MOVER_DE_MEMORIA_A_REGISTRO REG <sub>1</sub> 9
<b>1</b>	<b>ASIGNAR REG<sub>2</sub> 0</b>
<b>2</b>	SALTO_CONDICIONAL REG <sub>1</sub> 0 7
<b>3</b>	INCREMENTAR REG <sub>2</sub>
<b>4</b>	INCREMENTAR REG <sub>2</sub>
<b>5</b>	DECREMENTAR REG <sub>1</sub>
<b>6</b>	SALTO_INCONDICIONAL 2
<b>7</b>	<b>MOVER_DE_REGISTRO_A_MEMORIA REG<sub>2</sub> 10</b>
<b>8</b>	DETENER

Instrucciones en las que intervienen REG<sub>2</sub> (en verde)

En la ficha, se presentan otros dos programas para que los estudiantes los analicen de manera similar. El primero intercambia los valores almacenados en las posiciones 9 y 10 de la memoria. El segundo escribe, en cada celda del espacio de datos, el número de la dirección de la celda (en la celda de la dirección 9 se escribe un 9, en la de la dirección 10 un 10, y así siguiendo). Es interesante notar que, diferencia de los otros dos programas, en este se usa el número contenido en un registro para indicar una dirección de memoria.

<b>0</b>	MOVER_DE_MEMORIA_A_REGISTRO REG <sub>1</sub> 9
<b>1</b>	MOVER_DE_MEMORIA_A_REGISTRO REG <sub>2</sub> 10
<b>2</b>	MOVER_DE_REGISTRO_A_MEMORIA REG <sub>1</sub> 10
<b>3</b>	MOVER_DE_REGISTRO_A_MEMORIA REG <sub>2</sub> 9
<b>4</b>	DETENER

<b>0</b>	ASIGNAR REG <sub>1</sub> 9
<b>1</b>	SALTO_CONDICIONAL REG <sub>1</sub> 18 5
<b>2</b>	MOVER_DE_REGISTRO_A_MEMORIA REG <sub>1</sub> REG <sub>1</sub>
<b>3</b>	INCREMENTAR REG <sub>1</sub>
<b>4</b>	SALTO_INCONDICIONAL 1
<b>5</b>	DETENER

#### (a) Intercambia valores de la memoria

Programas propuestos en la ficha de estudiantes

#### (b) Escribe en cada celda su dirección

CIERRE

Les comentamos a los alumnos que la computadora ficticia que presentamos en la actividad, aun cuando solo tiene instrucciones para realizar operaciones muy simples y rudimentarias, ipuede hacer exactamente lo mismo que lo que hacen las que ellos utilizan cotidianamente! Esto quiere decir que con un conjunto de instrucciones muy simples alcanza para realizar los cálculos que hacen, por ejemplo, sus teléfonos inteligentes para ejecutar aplicaciones. En definitiva, lo que hacen las computadoras, es realizar operaciones muy sencillas a muy alta velocidad. Eso es todo.

# BIEN ADENTRO DE LA COMPUTADORA

¿Cómo hace una computadora para ejecutar cada instrucción de un programa? ¿Qué componentes internos intervienen? Metámonos bien adentro para ver qué es lo que pasa por allá abajo.



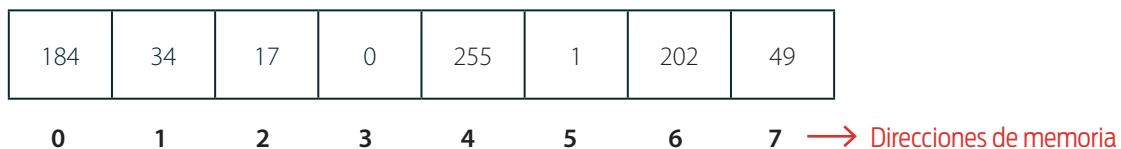
## RECORDATORIO

Para que un programa pueda ejecutarse, en primer lugar se compila y se obtiene un programa semánticamente equivalente en lenguaje ensamblador, es decir, con instrucciones que se corresponden con las operaciones que los circuitos del *hardware* de la máquina pueden realizar; en segundo lugar, el programa se ensambla para obtener código de máquina –instrucciones codificadas en un sistema binario que pueden almacenarse en la memoria de la computadora–.



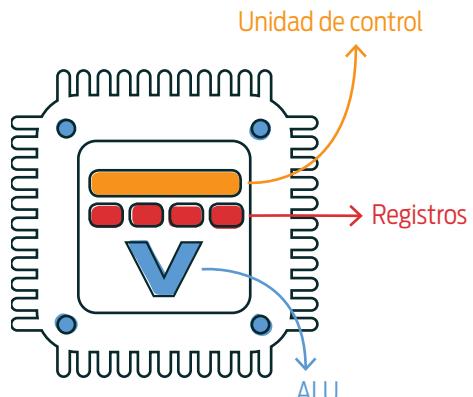
## DIRECCIONES DE MEMORIA

En la memoria, los bits se agrupan de a 8 y, por lo tanto, se la concibe como una secuencia de bytes. Cada byte de la memoria puede identificarse con un número, que se corresponde con la ubicación en la que se encuentra, comenzando a contar desde 0. A este número se lo conoce como **dirección de memoria**.



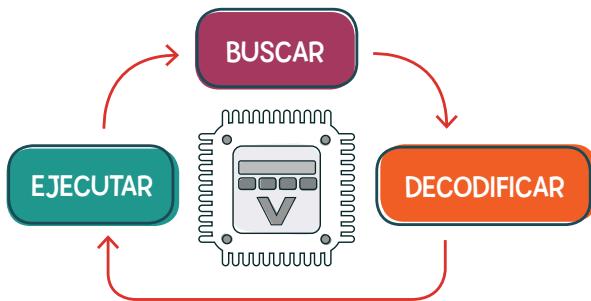
## LA CPU POR DENTRO

La unidad central de procesamiento, internamente, tiene: (i) una unidad aritmética lógica (ALU), que cuenta con circuitos para realizar operaciones aritméticas y lógicas simples; (ii) registros, que son pequeñas unidades de memoria, en los que almacena datos; y (iii) una unidad de control, que dirige el proceso de ejecución de las instrucciones de un programa.



## CICLO DE INSTRUCCIÓN

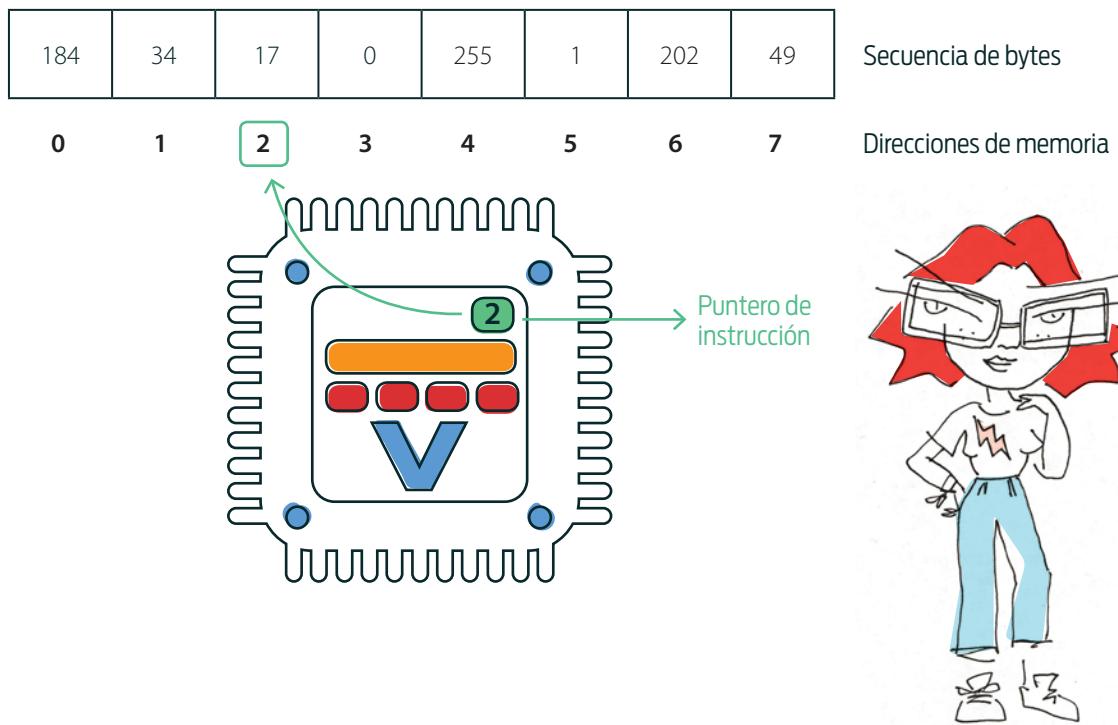
Un ciclo de instrucción comprende las tres etapas que lleva a cabo la unidad central de procesamiento para ejecutar una instrucción del lenguaje de máquina. La unidad de control se encarga de (i) **buscar** la instrucción en la memoria; (ii) **decodificar** la instrucción; y (iii) enviar las señales necesarias para que se pueda **ejecutar** la instrucción (dependiendo de cuál sea la instrucción, esto puede ser activar la ALU, cargar datos en registros, enviar señales a los circuitos que comunican la CPU con la memoria, etc.).



La unidad central de procesamiento repite este ciclo para cada instrucción que tiene que ejecutarse, hasta que el programa finalice su ejecución.

## ¿CÓMO SABE LA CPU CUÁL ES “LA PRÓXIMA INSTRUCCIÓN”?

Hay un registro interno de la unidad central de procesamiento, que se llama **puntero de instrucción**, que siempre tiene almacenada la dirección de memoria a la que hay que ir a buscar la próxima instrucción que debe ejecutarse. Antes de comenzar un nuevo ciclo de instrucción, la unidad de control actualiza su valor de modo que pase a contener la dirección de la próxima instrucción del programa que hay que ir a buscar a la memoria. En general, este valor será la siguiente posición de la memoria, aunque esto no siempre es así; por ejemplo, cuando se está en presencia de una alternativa condicional o una repetición.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

# LA COMPUTADORA EN ACCIÓN

Cada vez que interactuamos con una computadora hacemos que se ejecuten programas: usamos una aplicación de mensajería instantánea, una que muestra fotos, un reproductor de música, etc. ¿Cómo hace la computadora para ejecutarlos? En esta actividad iremos a fondo para ver cómo se ejecutan los programas en lenguaje de máquina. ¡Salió la ultra fina *Bet & Rob air!* Mirá las especificaciones de hardware:

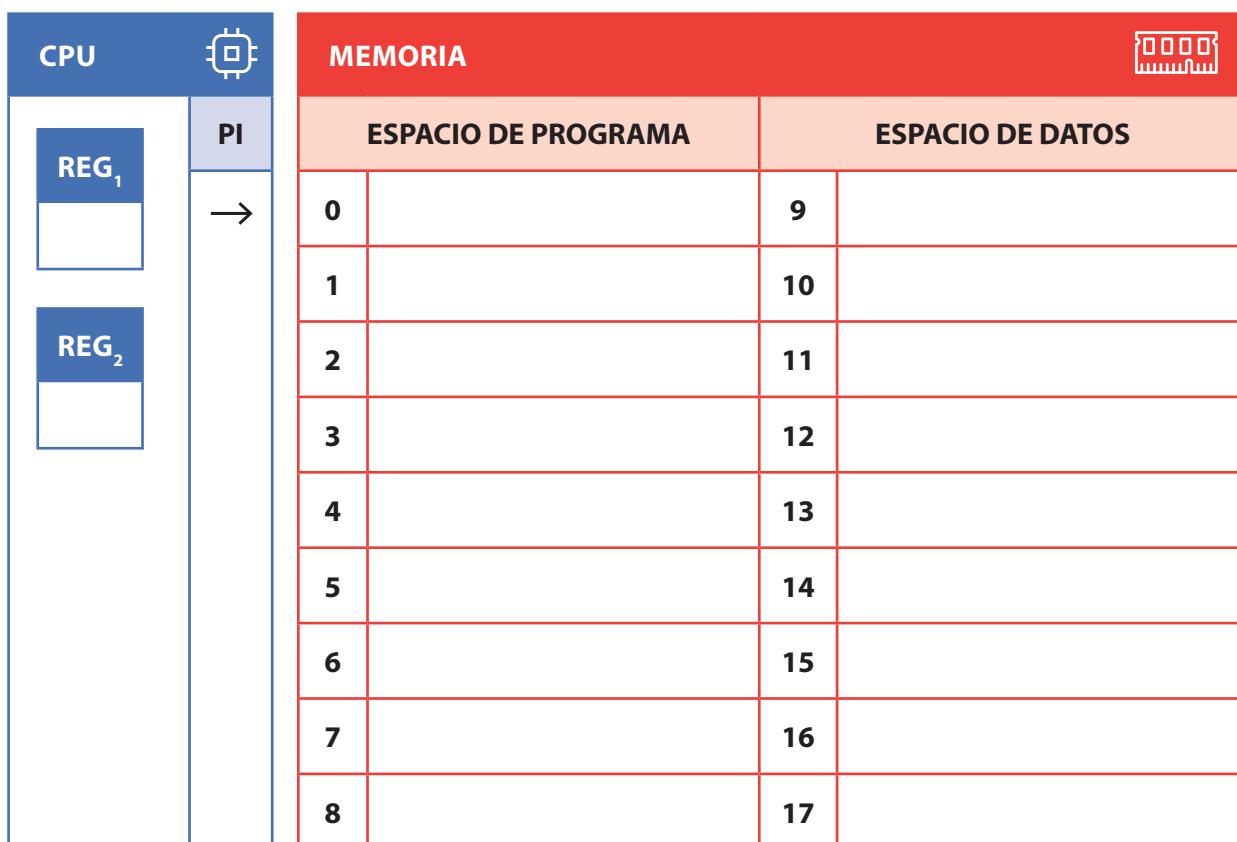


## Memoria

Tiene una memoria de 18 bytes; los primeros 9 están reservados para programas y los restantes se usan para almacenar datos.

## Registros de la unidad central de procesamiento

Posee 2 registros internos, **REG<sub>1</sub>** y **REG<sub>2</sub>**, que los programas usan para escribir y leer datos. Además, como cualquier otra computadora, también tiene un registro que actúa de puntero de instrucción: **PI**. Este, sin embargo, no puede ser mencionado en los programas.



NOMBRE Y APELLIDO:

CURSO:

FECHA:

## Lenguaje ensamblador

A continuación está el conjunto de instrucciones de su lenguaje ensamblador:

**ASIGNAR [r] [n]**: escribe en el registro  $r$  el número  $n$ .

**INCREMENTAR [r]**: incrementa en 1 el valor contenido en el registro  $r$ .

**DECREMENTAR [r]**: decrementa en 1 el valor contenido en el registro  $r$ .

**MOVER\_DE\_MEMORIA\_A\_REGISTRO [r] [n]**: escribe en el registro  $r$  el contenido de la celda de memoria cuya dirección es  $n$ .

**MOVER\_DE\_REGISTRO\_A\_MEMORIA [r] [n]**: lee el contenido del registro  $r$  y lo escribe en la posición  $n$  de la memoria.

**SALTO\_CONDICIONAL [r] [n] [i]**: lee el contenido del registro  $r$ , lo compara con el número  $n$  y, si son iguales, establece que la siguiente instrucción a ejecutar es la  $i$ -ésima (es decir, la que está en la posición  $i$  en el programa); si no, el flujo de ejecución continúa con la instrucción que se encuentra inmediatamente a continuación.

**SALTO\_INCONDICIONAL [i]**: establece que la siguiente instrucción a ejecutar es la  $i$ -ésima (es decir, la que está en la posición  $i$  en el programa).

**DETENER**: detiene la ejecución del programa.

1. Seguí paso a paso cómo evoluciona el estado del sistema a medida que se ejecuta el siguiente programa. Los valores iniciales de la memoria y los registros se observan a continuación.

CPU		PI	MEMORIA		
REG <sub>1</sub>	REG <sub>2</sub>	→	ESPACIO DE PROGRAMA		ESPACIO DE DATOS
0	9	0	MOVER_DE_MEMORIA_A_REGISTRO REG <sub>1</sub> 9	9	2
1	0	1	ASIGNAR REG <sub>2</sub> 0	10	0
2	7	2	SALTO_CONDICIONAL REG <sub>1</sub> 0 7	11	0
3	0	3	INCREMENTAR REG <sub>2</sub>	12	0
4	1	4	INCREMENTAR REG <sub>2</sub>	13	0
5	0	5	DECREMENTAR REG <sub>1</sub>	14	0
6	0	6	SALTO_INCONDICIONAL 2	15	0
7	0	7	MOVER_DE_REGISTRO_A_MEMORIA REG <sub>2</sub> 10	16	0
8	0	8	DETENER	17	0

NOMBRE Y APELLIDO:

CURSO:

FECHA:

¿Qué hubiese sucedido si, inicialmente, en la posición 9 de la memoria hubiese habido un 3? ¿Y si hubiera un 10?

---

---

---

---

¿Qué hace el programa?

---

---

---

---

2. Realizá el seguimiento de la ejecución de este otro programa.

CPU	PI	MEMORIA			
		ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
		0	MOVER_DE_MEMORIA_A_REGISTRO REG <sub>1</sub> 9	9	2
		1	MOVER_DE_MEMORIA_A_REGISTRO REG <sub>2</sub> 10	10	4
		2	MOVER_DE_REGISTRO_A_MEMORIA REG <sub>1</sub> 10	11	0
		3	MOVER_DE_REGISTRO_A_MEMORIA REG <sub>2</sub> 9	12	0
		4	DETENER	13	0
		5		14	0
		6		15	0
		7		16	0
		8		17	0

NOMBRE Y APELLIDO:

CURSO:

FECHA:

¿Qué hace el programa?

---

---

---

---

¿Tiene alguna importancia el valor inicial de los registros? ¿Qué pasaría si al iniciar la ejecución del programa, el contenido de las direcciones de memoria 9 y 10 fuese otro?

---

---

---

3. Ahora fijate qué pasa con este.

CPU		PI	MEMORIA		
		→	ESPACIO DE PROGRAMA		ESPACIO DE DATOS
	REG <sub>1</sub>		0	ASIGNAR REG <sub>1</sub> 9	9 2
	REG <sub>2</sub>		5	1 SALTO_CONDICIONAL REG <sub>1</sub> 18 5	10 4
				2 MOVER_DE_REGISTRO_A_MEMORIA REG <sub>1</sub> REG <sub>1</sub>	11 0
				3 INCREMENTAR REG <sub>1</sub>	12 0
				4 SALTO_INCONDICIONAL 1	13 0
				5 DETENER	14 0
				6	15 0
				7	16 0
				8	17 0

NOMBRE Y APELLIDO:

CURSO:

FECHA:

¿Cómo queda la memoria al finalizar la ejecución del programa?

---

---

---

---

¿Tiene alguna incidencia el contenido inicial de la memoria? ¿Y el valor de los registros?

---

---

---

---

# 07

# SISTEMAS OPERATIVOS

Un sistema operativo (SO) comprende una gran cantidad de piezas de *software* que se ubican entre los programas que usamos habitualmente y el *hardware* de la computadora. Se ocupa, por ejemplo, de la interacción de los programas con los periféricos de entrada y salida. Es también el responsable de la administración de los principales componentes de una computadora –como la unidad central de procesamiento y la memoria–, de la organización de los archivos en los que guardamos nuestros datos y de generar las condiciones para que los programas puedan ejecutarse. En un sistema de computación es, en definitiva, el gran director de orquesta que permite que todo acontezca.

## SECUENCIA DIDÁCTICA 1

- PRESENTACIÓN, TIEMPO Y ESPACIO
- Con ustedes, ¡el sistema operativo!
- Administración del tiempo
- Administración del espacio

En este capítulo presentamos a los sistemas operativos y abordamos algunos de los problemas que silenciosamente resuelven para que podamos usar nuestros dispositivos del modo en que lo hacemos habitualmente.



## Secuencia Didáctica 1

# PRESENTACIÓN, TIEMPO Y ESPACIO

Con el uso intensivo de las computadoras, hemos naturalizado muchas de las opciones y posibilidades que ellas nos proveen. No es usual que nos preguntemos por qué al encender nuestros dispositivos vemos lo que vemos en la pantalla, qué son realmente los archivos o cómo es que al presionar un botón un documento se imprime en una impresora. Esta secuencia didáctica comienza planteando preguntas para desnaturalizar lo que habitualmente damos por dado, y esboza respuestas que surgen de identificar algunos servicios provistos por el sistema operativo.

Al ejecutar varios programas al mismo tiempo, es necesario administrar el uso de los dos componentes más importantes de una computadora: la unidad central de procesamiento y la memoria. ¿Qué proceso dispone de la CPU en cada momento? ¿Qué porción de la memoria está disponible para cada uno de ellos? Las dos últimas actividades de la secuencia abordan los problemas que surgen al tener que compartir el tiempo y el espacio cuando hay más de un proceso en ejecución, y presenta las soluciones que brinda el sistema operativo para cada uno de ellos.

### ..... **OBJETIVOS**

- Comprender los sistemas operativos como un conjunto de piezas de *software*.
  - Reconocer los principales servicios que provee un sistema operativo.
  - Observar cómo los sistemas operativos administran el uso de la CPU y la memoria al ejecutarse más de un programa en forma concurrente.
- .....

## Actividad 1

### Con ustedes, iel sistema operativo!



TODA LA CLASE

**OBJETIVOS**

- Reconocer a los sistemas operativos como un conjunto de componentes de software.
- Identificar algunos servicios que brindan de los sistemas operativos.

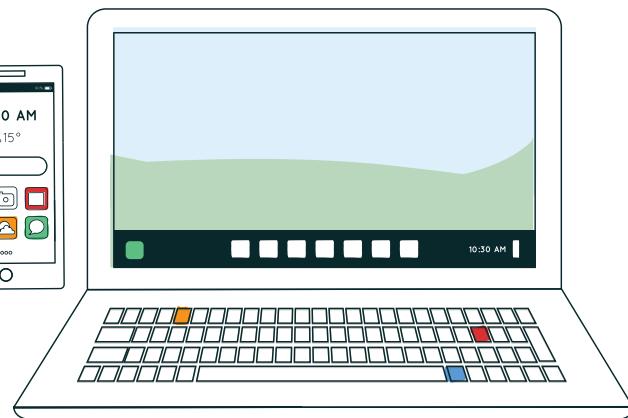
**MATERIALES**

Anexo para estudiantes

**DESARROLLO**

El objetivo de esta actividad es que los estudiantes reconozcan que los sistemas operativos son piezas de *software* que brindan interfaces y abstracciones que permiten que tanto usuarios como programas hagan uso de los recursos de una computadora.

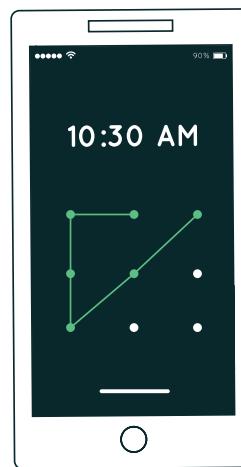
Comenzamos preguntando: “¿Qué sucede cuando encienden sus teléfonos?”. Es probable que surjan respuestas tales como “Se prende y está listo para que lo use” o “Aparecen en la pantalla los programas que tengo instalados”. Continuamos: “Efectivamente, al prender nuestros teléfonos vemos aparecer muchos íconos en la pantalla. Algunos de ellos nos permiten ejecutar programas que vienen instalados en el teléfono; otros sirven para ejecutar programas que instalamos nosotros; y otros, por ejemplo, acceder a las opciones de configuración de nuestros dispositivos. Además, en muchos casos, también se muestra otra información, como la hora, la temperatura, etc. En general, podemos decir que aparece un entorno que nos permite operar el teléfono. ¿Sucede lo mismo cuando prenden sus computadoras de escritorio o sus portátiles?”. Sí, también en este caso se habilita un entorno para operar la computadora.



Pantallas de inicio

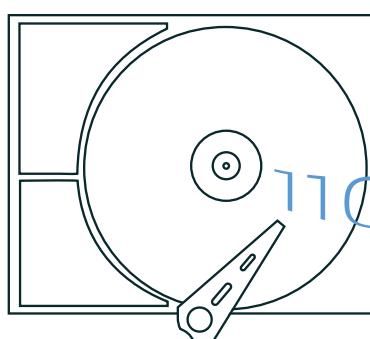
Luego preguntamos: “Todo esto que observamos, ¿cómo es que llega a mostrarse en la pantalla?”. Escuchamos las respuestas de los estudiantes y proseguimos: “Ustedes han construido programas para que, por ejemplo, se muestre un mensaje en la pantalla cuando se presiona un botón, ¿no es cierto? En este caso sucede algo parecido: si presionamos el ícono de la calculadora, veremos aparecer en la pantalla una aplicación que nos permite hacer cuentas. En ambos casos, de algún modo, hubo una respuesta del teléfono frente a un evento externo. En el primer caso, fue gracias a lo que programaron ustedes; ¿y en el segundo?”. Escuchamos las observaciones de los estudiantes y guiamos el intercambio para concluir que, cuando una computadora se enciende, comienza a ejecutarse una serie de progra-

mas –que alguien programó– que nos permite manejar la computadora. Al conjunto de estos programas –que en general vienen instalados de fábrica– se los conoce como **sistema operativo**. “Los sistemas operativos, en realidad, se encargan de un montón de cosas que, hasta aquí, a lo mejor pensábamos que sucedían porque sí, sin cuestionarlas. Por ejemplo, que al presionar un ícono se cargue un programa; que al mover el ratón, el puntero se mueva por la pantalla; que podamos ejecutar muchas aplicaciones al mismo tiempo; o que al poner una clave podamos desbloquear nuestros teléfonos para usarlos”.

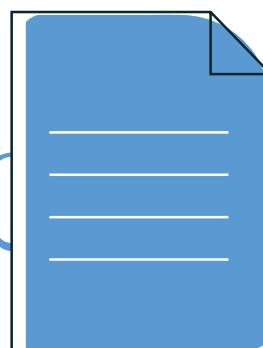


Programa para desbloquear un teléfono

Continuamos: “¿Qué son para ustedes los archivos?”. Es posible que entre las respuestas se haga referencia a archivos de imágenes, de texto, de video, etc. “Ciertamente, una foto, un texto o un video se *puede guardar* en un archivo, pero nuevamente, ¿qué es un archivo?”. Escuchamos sus respuestas e indagamos: “¿Dónde se guardan los archivos?”. Algún estudiante podría mencionar los discos rígidos o los pendrives. “Así es, los **archivos** los conservamos en dispositivos de almacenamiento, como por ejemplo un disco rígido. Sin embargo, un disco no sabe nada de archivos; lo único que hace es usar un cabezal para escribir y leer *bits* en los platos magnéticos que tiene en su interior. Qué porciones del disco representan un archivo, cómo está organizado, etc., es algo que excede sus competencias. Los archivos son abstracciones que nos provee el sistema operativo para que nosotros (y los programas) podamos agrupar lógicamente información que nos resulte de interés y manipularla”.



1101010110...



Los archivos son abstracciones provistas por los sistemas operativos

Luego preguntamos: “¿Cómo organizan ustedes sus archivos en la computadora?”. Probablemente los estudiantes mencionen que los agrupan en carpetas, siguiendo algún criterio que les permita luego localizarlos con facilidad. “Una función clave de un sistema operativo es proveer una interfaz limpia y clara para que podamos crear, leer y modificar archivos, además de organizarlos en carpetas. Al conjunto de programas que permite administrar y manipular archivos en medios físicos se lo conoce como **sistema de archivos**. El sistema operativo tiene registro sobre en qué lugar del disco está cada archivo, cómo es la organización en carpetas, etc”.



Sistema de archivos

A continuación les decimos: “No es extraño que un manual que describa cómo leer y escribir datos en una tarjeta de memoria a nivel físico –es decir, interactuando con el *hardware*– tenga más de 500 páginas. ¡Ningún programador en su sano juicio querría leerlo para hacer algo tan básico como guardar un archivo! Además, la forma de manipular cada marca y modelo de tarjeta de memoria es distinta. Aun si programamos la interacción con un medio de almacenamiento, si cambiáramos la tarjeta, el programa dejaría de funcionar!”.

Les comentamos: “Por suerte, hay piezas de software llamadas *controladores* –o *drivers*, en inglés–, que se ocupan de la interacción con el *hardware*. Los controladores proporcionan una interfaz sencilla para comunicarnos con los componentes físicos de una computadora, sin entrar en detalles. Los sistemas operativos contienen varios de ellos y, además, cuando incorporamos nuevos dispositivos a nuestras computadoras –como impresoras, monitores, etc.–, también es posible agregarles controladores para estos. Así, nos permiten manejarlos de manera sencilla, ilo que de otro modo sería un mundo lleno de dificultades!”.

Finalmente preguntamos: “¿Qué sistemas operativos conocen?”. En caso de que no surgieran algunos nombres, comentamos que para computadoras de escritorio y portátiles los más difundidos son *Windows*, *Linux* y *macOS*; y para teléfonos celulares y tabletas, *Android*, *iOS* y *Windows Mobile*.

## CIERRE

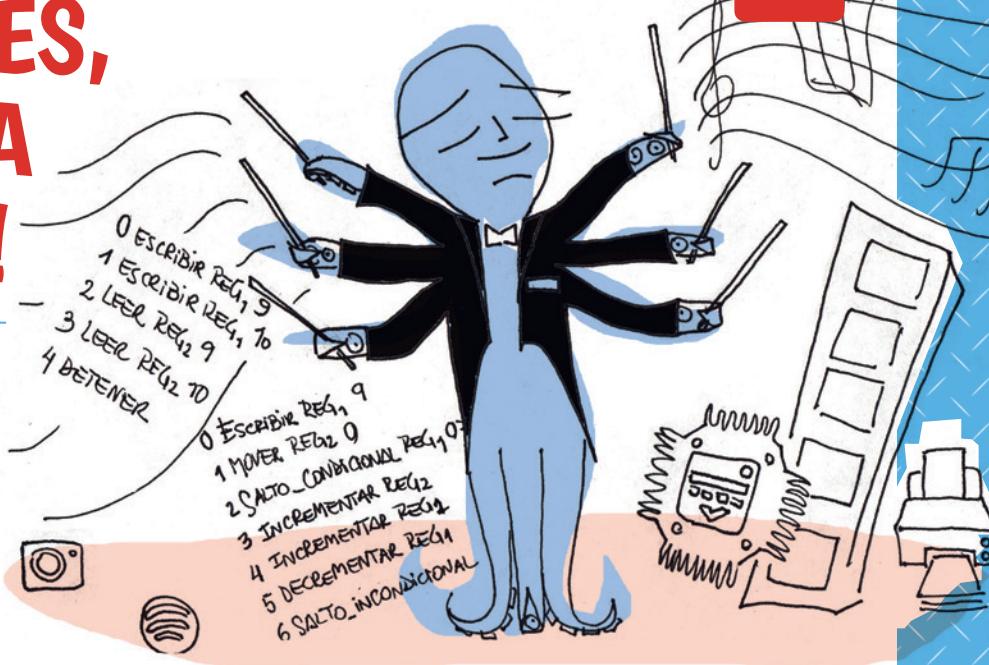
Les comentamos a los alumnos que, además de brindar un entorno para manejar nuestras computadoras (como usuarios) e interactuar con el *hardware* (como programadores), los sistemas operativos se ocupan de muchas otras cosas. Por ejemplo, de que podamos ejecutar varios programas en forma simultánea, de administrar la memoria, de garantizar la seguridad de nuestros dispositivos, etc. Concluimos que un sistema operativo es como un director de orquesta que permite que efectivamente acontezca todo lo que sucede en una computadora.

---

# CON USTEDES, ¡EL SISTEMA OPERATIVO!

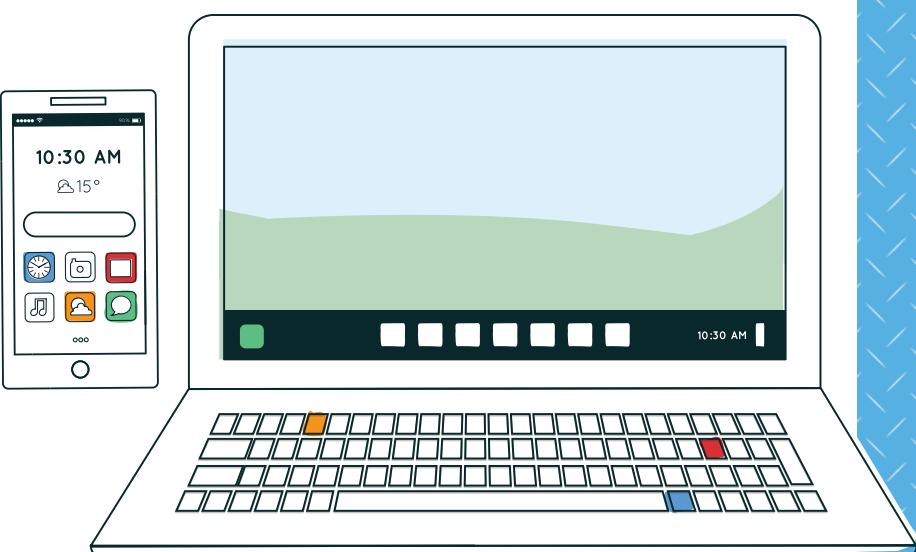
¿Por qué, cuando encendemos una computadora, aparece una interfaz que nos permite operarla? ¿Quién se ocupa de permitirnos organizar información en archivos y carpetas? ¿Cómo se lleva a cabo la interacción entre los programas y los dispositivos de *hardware*? ¿Por qué podemos ejecutar muchos programas al mismo

tiempo? Aunque solemos dar por sentadas estas cosas sin cuestionarlas, hay piezas de software que se ocupan de que todo esto sea posible: en su conjunto se las conoce como sistema operativo. El sistema operativo es, de algún modo, ¡el director de orquesta que hace que en una computadora todo acontezca!



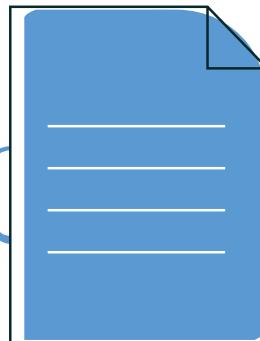
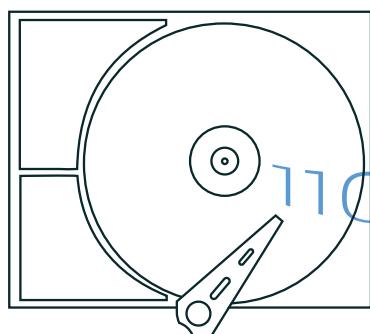
## ENTORNO DE TRABAJO

Cuando prendemos nuestros dispositivos nos encontramos con muchos íconos, botones e información –entre otras cosas–, que nos permiten manejarlos. Ya se trate de portátiles, de tabletas o teléfonos inteligentes, tenemos la posibilidad de ejecutar programas, configurar opciones y leer información. ¡Esto no sucede porque sí! Hay un conjunto de programas llamado sistema operativo –que, dicho sea de paso, fue programado por personas–, que comienza a correr no bien encendemos nuestras computadoras y que a nosotros, como usuarios, nos brinda un entorno agradable para poder operarlas.



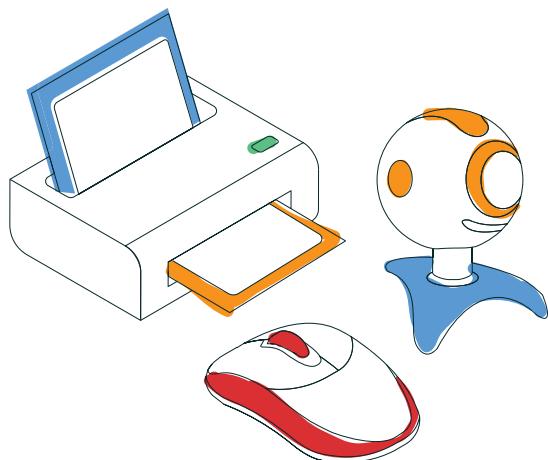
## ¿QUÉ ES UN ARCHIVO?

Con mucha frecuencia, usamos una computadora para ver fotos, escribir textos, escuchar música, ver videos, etc., que están guardados en archivos en algún medio de almacenamiento, como por ejemplo un disco rígido. Pero los discos, usando un cabezal, solo saben leer, escribir y borrar bits en los platos que hay en su interior. ¡No tienen la menor idea de qué es un archivo! Entonces, ¿qué son los archivos? Los **archivos** no son más que **abstracciones** que nos provee el sistema operativo para que nosotros (y los programas) podamos agrupar lógicamente información que nos resulte de interés y manipularla. Ni más ni menos.



## SISTEMA DE ARCHIVOS

Una función clave de un sistema operativo es proveer una interfaz limpia y clara para que podamos crear, leer y modificar archivos, además de organizarlos en carpetas. Al conjunto de programas que nos permite hacerlo se lo conoce como *sistema de archivos*. El sistema operativo tiene registro sobre en qué lugar del disco está cada archivo, cómo es la organización en carpetas, etc.



## CONTROLADORES DE HARDWARE

No es extraño que un manual que describa cómo interactuar con un dispositivo de *hardware* –como un disco, una impresora, etc.– tenga más 500 páginas. ¡Ningún programador en su sano juicio querría leerlo para hacer algo tan básico como crear un archivo o mandar a imprimir un documento! Por suerte, hay piezas de *software* llamadas *controladores* –o *drivers*, en inglés–, que se ocupan de la interacción con el *hardware*. Los controladores proporcionan una interfaz sencilla para comunicarnos con los componentes físicos de una computadora, sin necesidad de entrar en detalles. Los sistemas operativos contienen varios de ellos y, además, cuando incorporamos nuevos dispositivos a nuestras computadoras –como, monitores, escáneres, etc.– también es posible agregar controladores para estos. De este modo, al programar, ¡podemos interactuar con el *hardware* de un modo claro, simple y lindo, lo que, de otro modo, sería oscuro, difícil y monstruoso!

## Actividad 2

### Administración del tiempo

 TODA LA CLASE

#### OBJETIVOS

- Comprender la noción de proceso.
- Conocer los sistemas operativos multitareas.
- Observar que los programas no se ejecutan en forma simultánea.

#### MATERIALES

 Anexo de la actividad

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes comprendan que los llamados sistemas operativos multitarea –prácticamente todos los usados en la actualidad– generan la impresión de que muchos programas pueden ejecutarse en forma simultánea, cuando en realidad no es así. Lo que realmente sucede es que las instrucciones de los distintos programas se ejecutan intercaladamente, por turnos, de modo que cada programa avance un poco en un período de tiempo relativamente corto. La muy alta velocidad a la que funcionan los procesadores es la que genera la ilusión de que el avance de la ejecución de los programas se produce en forma concurrente.

La actividad está dividida en dos partes. En la primera se introduce la diferencia entre programa –descripción del comportamiento esperado de una computadora– y proceso –los programas en ejecución–. En la segunda, se trabaja sobre el modo en que se genera la impresión de que distintos procesos se ejecutan simultáneamente.

#### Programas y procesos

Comenzamos preguntando a los estudiantes: “¿A alguno de ustedes le gusta cocinar?”. Escuchamos sus respuestas y continuamos: “Imaginen que quieren preparar por primera vez esos deliciosos ñoquis caseros que cocinan sus abuelitas. La nona, la bobe o como sea que ustedes la llamen, sabiendo que probablemente fracasarán, tiene el enorme gesto de facilitarles la receta. Allí figuran tanto los ingredientes –papas, harina, un huevo, aceite, sal, etc.– como los pasos necesarios para llevarla a cabo –pelar las papas, ponerlas a hervir, sacarlas y hacer un puré agregándole el aceite, incorporar un huevo y mezclar, agregar la harina, etc.–. ¿En qué les hace pensar esto? ¿Alguna otra cosa además de abrirles el apetito?”. Guiamos la discusión para arribar a la conclusión de que la receta es, en definitiva, una descripción paso a paso de cómo preparar la pasta.

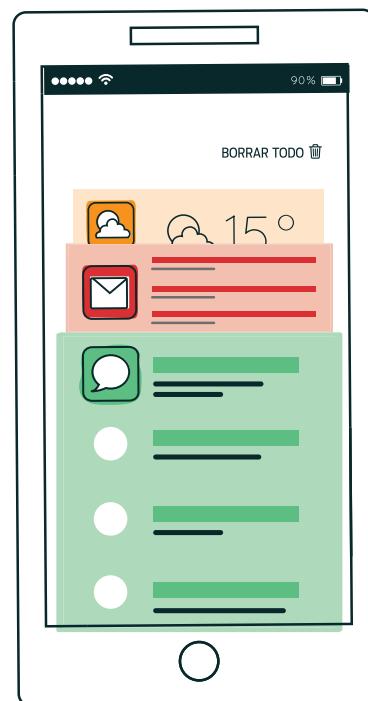
Continuamos: “Ahora llegó el momento de cocinar... ¿qué hacemos?”. Es probable que alguien responda que tenemos que seguir, una a una, las instrucciones de la receta. “Es cierto, eso es lo mejor que podemos hacer. Nada de inventar cosas raras, ¿eh?“.

Luego comentamos: “Tenemos dos cosas que podemos diferenciar: por una lado, la receta, que es una descripción paso a paso de cómo cocinar un plato de comida; por otro, nosotros cocinando, que es una actividad que comienza, se desarrolla y termina. Si en lugar de recetas y preparación de comida pensamos en términos de programas y computadoras, ¿cómo sería la analogía?”. Escuchamos sus respuestas y continuamos: “Por un lado, están los programas, que describen

(en algún lenguaje de programación) un comportamiento esperado por parte de una computadora. Por otro, los programas en ejecución, que siguen paso a paso las instrucciones de los programas. Estos últimos, a los que se llama **procesos**, pueden ser comprendidos como entidades “vivas”: un proceso “nace” –comienza a ejecutarse–, “se desarrolla” –avanza en su ejecución– y “muere” –termina de ejecutarse–. Por su parte, los programas son solo descriptivos; son entidades estáticas”.

### Ejecución de muchos programas

Les preguntamos a los estudiantes: “Al usar sus teléfonos inteligentes, ¿algunas veces tienen abiertos dos o más aplicaciones al mismo tiempo? Por ejemplo, ¿alguna vez contestan un mensaje usando un servicio de mensajería instantánea al mismo tiempo que escuchan una canción o miran un video?”. Guiamos el intercambio para arribar a la conclusión de que tanto el programa de mensajería instantánea como el reproductor de música y el reproductor de video son tres aplicaciones distintas; posiblemente todos (o casi) alguna vez hayan usado sus teléfonos para más de una cosa a la vez. “De hecho, en general, los teléfonos tienen una opción que nos permite observar todas las aplicaciones que se encuentran abiertas. ¿Lo habían notado?”. Continuamos preguntando: “Con las computadoras de escritorio y las portátiles, ¿pasa lo mismo?”. Sí, también en ellas se pueden ejecutar muchos programas a la vez. Por ejemplo, un navegador de Internet, un reproductor de música, un entorno de programación, etc.

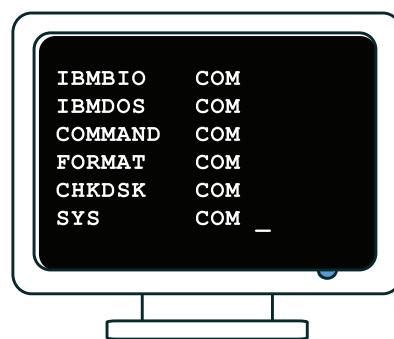


Muchas aplicaciones abiertas

“Para que sea posible ejecutar muchos programas a la vez, la computadora debe tener instalado lo que se conoce como **sistema operativo multitarea** –o *multitasking*, en inglés–. Hoy en día, casi todos lo son, pero esto no era así en las primeras computadoras personales, hace solo 40 años. En aquel entonces, para correr dos programas, primero se ejecutaba uno y, recién cuando este terminaba, se ejecutaba el otro”.

### UN SISTEMA MONOTAREA

En 1980, Microsoft lanzó el sistema operativo MS-DOS (acrónimo de *Microsoft Disk Operating System*) para computadoras personales que fue, en sus primeras versiones, un SO **monotarea**. A pesar de no destacarse por sus cualidades técnicas, logró imponerse en el mercado por un acuerdo con IBM, ya que se ofreció a un precio mucho más bajo que los de sus competidores. Recién en la versión 4.0, liberada en 1986, se convirtió en un SO **multitarea**.



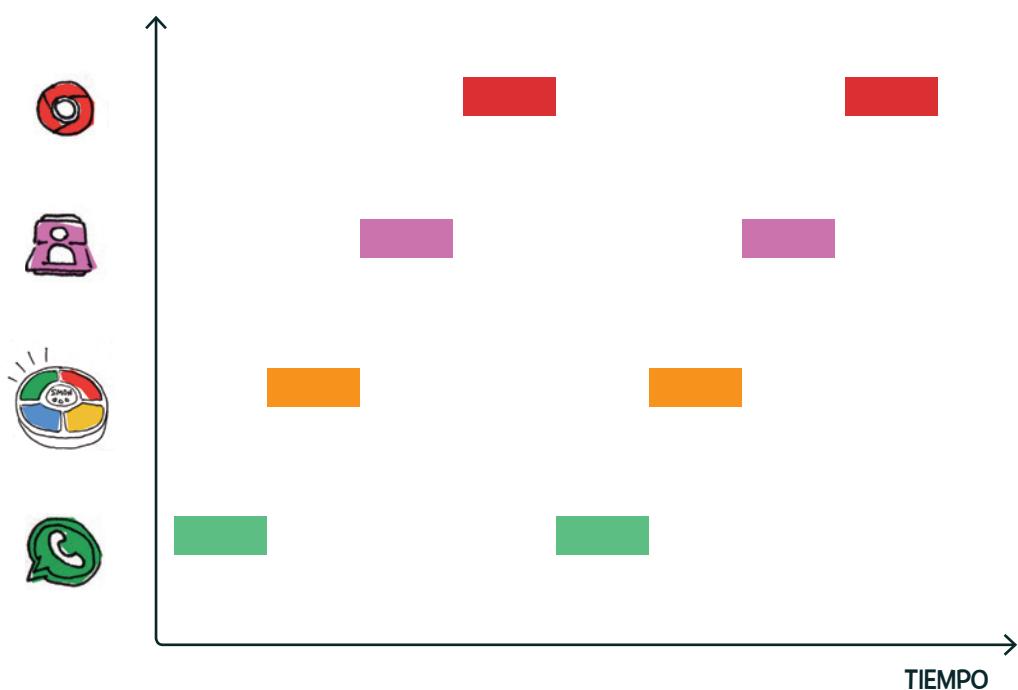
Continuamos: “Como ya hemos visto, la unidad central de procesamiento es el componente de la computadora que se encarga de ejecutar una a una las instrucciones de los programas. ¿Se les ocurre cómo lleva a cabo esta tarea cuando son muchos los programas que se están ejecutando?”.

Retomamos entonces el ejemplo de los ñoquis: “Volviendo a la receta de la abuela, supongan que mientras están cocinando, antes de tirar los ñoquis en la olla, un mosquito les empieza a zumbar cerca del oído. Pero no un mosquito que viene y se va, ¿eh? Uno que insiste, insiste e insiste. Muy pero muy molesto. ¿Qué harían?”. Es probable que algún estudiante conteste que, antes de tirar la pasta al agua, intentaría matarlo o, al menos, iría a buscar un repelente para ahuyentarlo. “No podría estar más de acuerdo ¡A mí los mosquitos me vuelven loco! Una vez resuelto este pequeño inconveniente, entonces sí, tiramos los ñoquis en la olla, ¿verdad?”.

Luego les decimos: “Detengámonos un segundo en lo que acabamos de describir. Estábamos llevando a cabo una tarea –cocinar– que en cierto momento interrumpimos para realizar otra –ocuparnos del mosquito– y que, luego, retomamos. ¿Cocinamos y nos ocupamos del mosquito al mismo tiempo?”. Escuchamos las reflexiones de los estudiantes y concluimos que, en realidad, dedicamos una porción de tiempo para llevar a cabo cada tarea pero, sin embargo, no hicimos las dos cosas a la vez en ningún momento: mientras cocinamos no nos ocupamos del mosquito y mientras nos ocupamos del mosquito no cocinamos.

A continuación retomamos la pregunta: “¿Qué creen que hacen las computadoras cuando hay varios programas corriendo –o, lo que es lo mismo, varios procesos en ejecución–?”. Es esperable que, a esta altura, algún estudiante conteste que la computadora fracciona el tiempo para que las instrucciones de los distintos programas se ejecuten intercaladamente. “¡Así es! Los programas no se ejecutan realmente al mismo tiempo. Cada uno avanza un poco cuando el procesador se dedica a ejecutar sus instrucciones. Lo que nos da la apariencia de que todos se ejecutan al mismo tiempo es la altísima velocidad a la

que funciona la CPU.<sup>1</sup> Adivinen quién se encarga de fraccionar el tiempo y decidir qué proceso se ejecuta en cada momento". En caso de que nadie conteste, les indicamos que se trata del sistema operativo. Para dar una idea gráfica de lo descrito, copiamos en el pizarrón el siguiente gráfico:



Las aplicaciones se ejecutan en forma intercalada

### CIERRE

Les comentamos a los estudiantes que, en los sistemas operativos multitarea, hay un programa llamado ***scheduler***, que es el que se encarga de dividir el tiempo de uso del procesador y determinar qué proceso se ejecuta en cada turno. Hay muchos algoritmos de *scheduling* distintos que priorizan distintas variables: el tiempo de espera promedio de los procesos para disponer del procesador, la equidad del tiempo disponible del procesador para cada proceso, la ejecución de aquellos procesos que sean más críticos, etc.

<sup>1</sup> Las computadoras actuales suelen tener más de un núcleo, por lo que sí pueden ejecutar distintas instrucciones simultáneamente. Sin embargo, la explicación es válida para cada núcleo cuando se ejecutan más programas que la cantidad de núcleos que tiene la computadora. Por ejemplo, si ejecutamos 8 programas en un dispositivo que posee dos núcleos, para que todos los procesos puedan avanzar será indispensable que se vayan turnando en el uso de los dos núcleos.

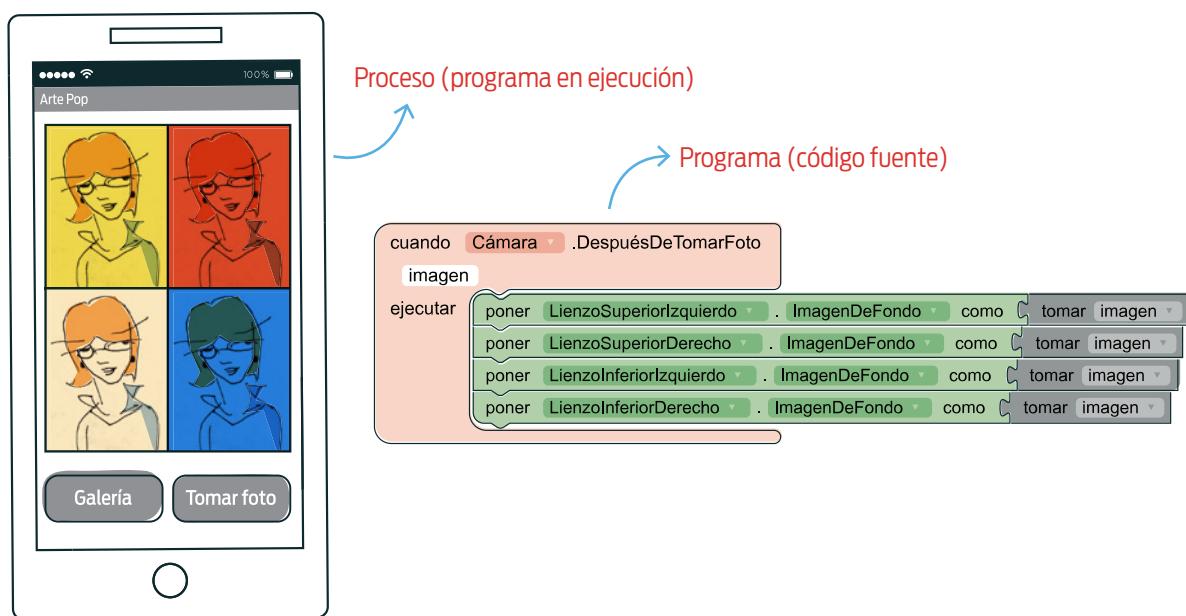
# ADMINISTRACIÓN DEL TIEMPO



Es habitual que usemos muchos programas al mismo tiempo. Por ejemplo, un navegador de Internet, un reproductor de música, un programa para chatear, etc. ¿Cómo es posible que esto suceda? ¿Realmente se ejecutan todos al mismo tiempo?

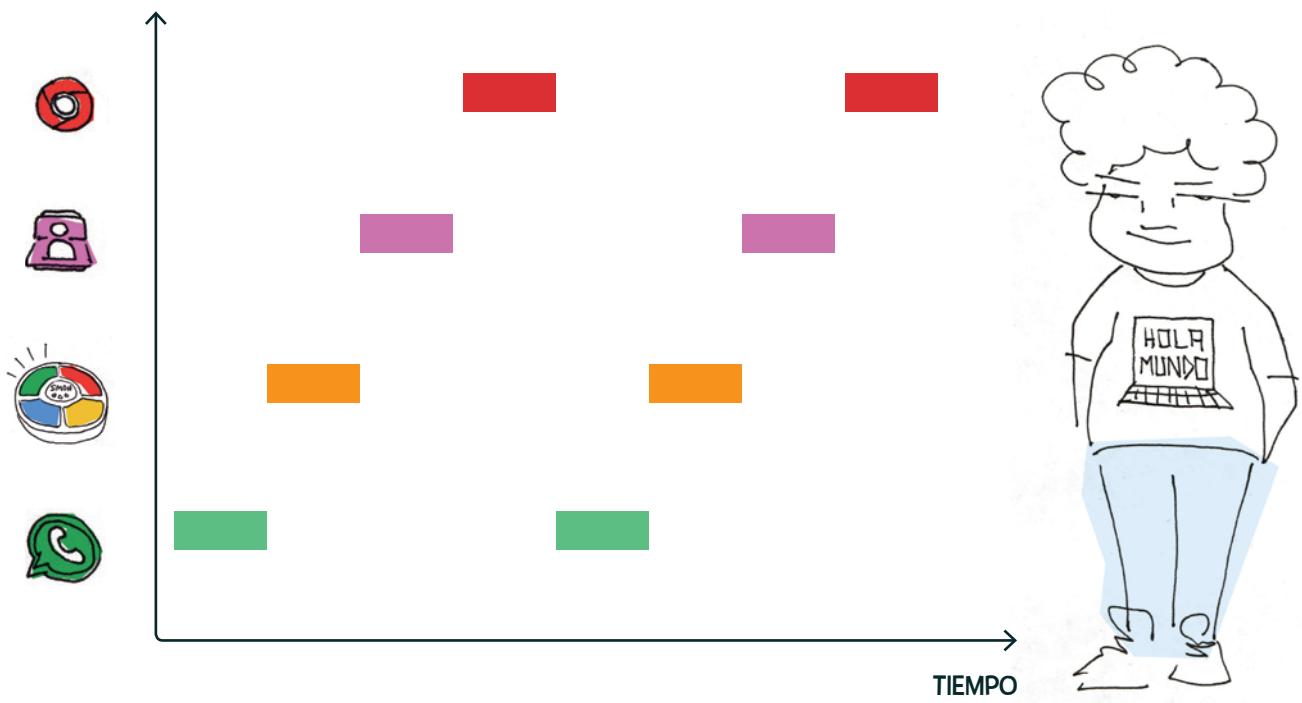
## PROGRAMAS Y PROCESOS

Los programas, escritos en algún lenguaje de programación, describen el comportamiento esperado por parte de una computadora. A su vez, cuando se ejecutan, se siguen paso a paso las instrucciones del programa. A los **programas** en ejecución se los llama **procesos**. Los procesos son semejantes a entidades “vivas”: un proceso “hace” –comienza a ejecutarse–, “se desarrolla” –avanza en su ejecución– y “muere” –termina de ejecutarse–. Por su parte, los programas son solo descriptivos; son entidades estáticas.



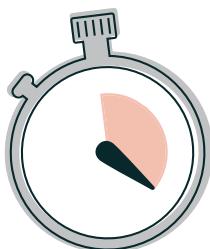
## ¿MUCHOS PROCESOS AL MISMO TIEMPO?

Cuando corremos muchos programas al mismo tiempo, las instrucciones de cada uno de ellos se ejecutan intercaladamente, por turnos, de modo que cada proceso puede avanzar un poco en un período de tiempo relativamente corto. La muy alta velocidad a la que funcionan los procesadores es la que genera la ilusión de que el avance de la ejecución de los programas se produce en forma simultánea.



## SCHEDULER

En los sistemas operativos, hay un programa llamado **scheduler**, que es el que se encarga de dividir el tiempo de uso del procesador y determinar qué proceso se ejecuta en cada turno. Hay muchas estrategias de *scheduling* distintas que priorizan diferentes variables: el tiempo de espera promedio de los procesos para disponer del procesador, la equidad del tiempo disponible del procesador para cada proceso, la ejecución de aquellos procesos que sean más críticos, etc.



## ¿Y SI TENEMOS MUCHOS NÚCLEOS?

Las computadoras actuales suelen tener más de un núcleo, por lo que sí pueden ejecutar distintas instrucciones simultáneamente. Sin embargo, cuando se ejecutan más programas que la cantidad de núcleos que tiene la computadora, no pueden ejecutarse todos al mismo tiempo. Por ejemplo, si ejecutamos 8 programas en un dispositivo que posee dos núcleos, resulta indispensable que se vayan turnando para que todos puedan avanzar. También en este caso es el *scheduler* el que se encarga de dividir el tiempo de uso de los núcleos e ir administrando los turnos entre los distintos procesos.



## Actividad 3

### Administración del espacio

 DE A DOS

#### OBJETIVOS

- Identificar el problema de interferencia entre procesos.
- Conocer los espacios de memoria de uso exclusivo.

#### MATERIALES

-  Ficha para estudiantes

#### DESARROLLO

El objetivo de esta actividad es que los estudiantes reconozcan que cuando varios procesos se ejecutan simultáneamente, para que no exista interferencia entre ellos, es necesario que cada uno tenga un espacio de uso exclusivo en la memoria.

Comenzamos repartiendo la ficha a los estudiantes y les comentamos: "Hoy trabajaremos con una antigua computadora que venía con una impresora integrada. Se trata de la vieja y querida *Bet & Rob compuimpresora*. En su momento fue muy de avanzada y se la llamó 'la dos en uno', por obvias razones. Era, en esencia, una computadora muy sencilla que podía programarse con dos instrucciones: una que permitía escribir una letra en una posición de la memoria, y otra que, dado un rango de direcciones de memoria, mandaba a imprimir el contenido de tales celdas". Les indicamos, entonces, que lean la descripción de las instrucciones de la computadora.



#### Lenguaje de la Bet & Rob compuimpresora

**ESCRIBIR [c, n]**: escribe el carácter *c* en la posición *n* de la memoria, donde *c* y *n* se reemplazan por un carácter y un número, respectivamente, cada vez que la instrucción es invocada.

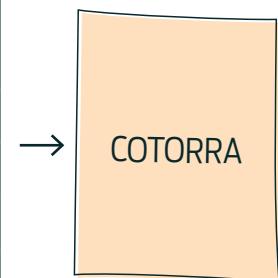
**IMPRIMIR [n<sub>1</sub>, n<sub>2</sub>]**: imprime los caracteres que se encuentran entre las posiciones *n<sub>1</sub>* y *n<sub>2</sub>* de la memoria, donde *n<sub>1</sub>* y *n<sub>2</sub>* se reemplazan por un número cada vez que la instrucción es invocada.

Una vez que les hayan quedado claras las instrucciones de esta computadora, les pedimos que resuelvan la primera consigna. Allí observarán, en primer lugar, la organización interna de la memoria: tiene 32 posiciones, de las cuales las primeras 16 están reservadas para programas y las últimas 16 para almacenar datos. Se les pide, entonces, que describan el resultado de la ejecución de dos programas: *Cotorra volá* y *Lará lará laringe*. El primero escribe en la memoria la palabra *cotorra* y, luego, la imprime en una hoja; el segundo hace lo propio con la palabra *laringe*.

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	
1	ESCRIBIR ["O", 17]	17	
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8		24	

→

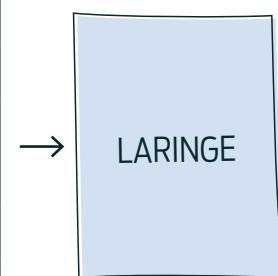
MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	C
1	ESCRIBIR ["O", 17]	17	O
2	ESCRIBIR ["T", 18]	18	T
3	ESCRIBIR ["O", 19]	19	O
4	ESCRIBIR ["R", 20]	20	R
5	ESCRIBIR ["R", 21]	21	R
6	ESCRIBIR ["A", 22]	22	A
7	IMPRIMIR [16, 22]	23	
8		24	



MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["L", 16]	16	
1	ESCRIBIR ["A", 17]	17	
2	ESCRIBIR ["R", 18]	18	
3	ESCRIBIR ["I", 19]	19	
4	ESCRIBIR ["N", 20]	20	
5	ESCRIBIR ["G", 21]	21	
6	ESCRIBIR ["E", 22]	22	
7	IMPRIMIR [16, 22]	23	
8		24	

→

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["L", 16]	16	L
1	ESCRIBIR ["A", 17]	17	A
2	ESCRIBIR ["R", 18]	18	R
3	ESCRIBIR ["I", 19]	19	I
4	ESCRIBIR ["N", 20]	20	N
5	ESCRIBIR ["G", 21]	21	G
6	ESCRIBIR ["E", 22]	22	E
7	IMPRIMIR [16, 22]	23	
8		24	



Resultado de las ejecuciones de Cotorra volá y Lará lará laringe

Una vez que todos hayan completado la consigna, hacemos una puesta en común y continuamos: “Aunque parezca muy rudimentaria, iesta computadora trae incorporado un sistema operativo multitarea! Es decir, nos permite ejecutar muchos programas sin tener que esperar que termine uno para empezar con el siguiente”. Recordamos entonces que, en los sistemas operativos multitarea, los distintos programas no ejecutan sus instrucciones en forma simultánea; es el *scheduler* quien divide el tiempo de uso del procesador en turnos y permite que cada programa avance un poco en cada turno.

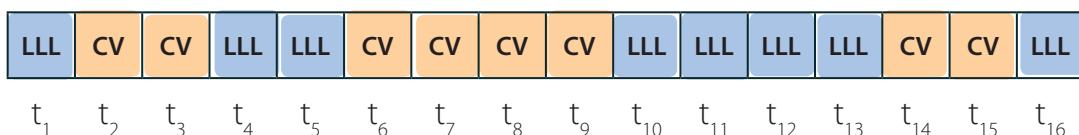
A continuación, les indicamos que resuelvan la segunda consigna de la actividad. Allí se presenta un escenario en el que ambos programas se cargan en la memoria para ser ejecutados. “Como ya hemos visto, al correr dos programas se van intercalando las ejecuciones de cada proceso. La forma en la que se organizan los turnos y qué proceso dispone del procesador en cada turno es algo que decide el *scheduler* que, como ustedes ya saben, es uno de los programas que forma parte del sistema operativo. En la consigna pueden observar, además de los programas, el orden dispuesto por el *scheduler* para la ejecución de las instrucciones de ambos programas”.

MEMORIA		
	ESPACIO DE PROGRAMA	ESPACIO DE DATOS
0	ESCRIBIR ["C", 16]	16
1	ESCRIBIR ["O", 17]	17
2	ESCRIBIR ["T", 18]	18
3	ESCRIBIR ["O", 19]	19
4	ESCRIBIR ["R", 20]	20
5	ESCRIBIR ["R", 21]	21
6	ESCRIBIR ["A", 22]	22
7	IMPRIMIR [16, 22]	23
8	ESCRIBIR ["L", 16]	24
9	ESCRIBIR ["A", 17]	25
10	ESCRIBIR ["R", 18]	26
11	ESCRIBIR ["I", 19]	27
12	ESCRIBIR ["N", 20]	28
13	ESCRIBIR ["G", 21]	29
14	ESCRIBIR ["E", 22]	30
15	IMPRIMIR [16, 22]	31

Cotorra volá y Lará lará laringe en la memoria para ser ejecutados

Lará lará laringe

Los estudiantes deberán, en este caso, indicar en la ficha la evolución del estado de la memoria teniendo en cuenta que las instrucciones de los programas fueron intercaladas por el *scheduler* de la siguiente manera para su ejecución:<sup>1</sup>



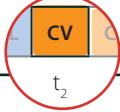
Orden en el que se ejecutan las instrucciones de ambos procesos

A continuación se muestra la solución a la que deben arribar los estudiantes:

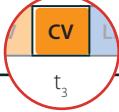
MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	
1	ESCRIBIR ["O", 17]	17	
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	L
1	ESCRIBIR ["O", 17]	17	
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	

<sup>1</sup> Para simplificar la lectura, se abrevia *Cotorra volá* como *CV* y *Lará lará laringe* como *LLL*.

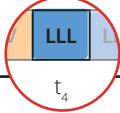


MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	C
1	ESCRIBIR ["O", 17]	17	
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	

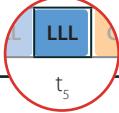


MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	C
1	ESCRIBIR ["O", 17]	17	O
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	

→



MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	C
1	ESCRIBIR ["O", 17]	17	A
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	



MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	C
1	ESCRIBIR ["O", 17]	17	A
2	ESCRIBIR ["T", 18]	18	R
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	

→

**MEMORIA**

ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	A
2	ESCRIBIR ["T",18]	18	T
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

**MEMORIA**

ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	A
2	ESCRIBIR ["T",18]	18	T
3	ESCRIBIR ["O",19]	19	O
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

**MEMORIA**

ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	A
2	ESCRIBIR ["T",18]	18	T
3	ESCRIBIR ["O",19]	19	O
4	ESCRIBIR ["R",20]	20	R
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	A
2	ESCRIBIR ["T",18]	18	T
3	ESCRIBIR ["O",19]	19	I
4	ESCRIBIR ["R",20]	20	R
5	ESCRIBIR ["R",21]	21	R
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

→

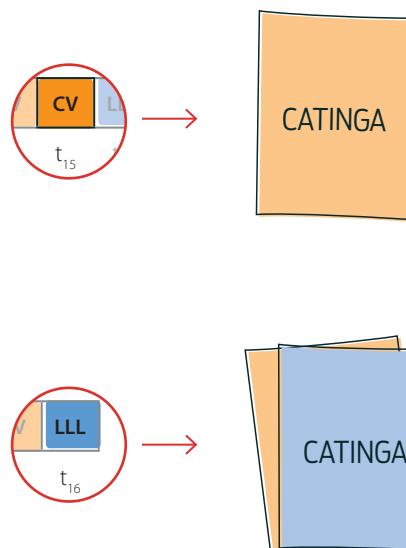
MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	A
2	ESCRIBIR ["T",18]	18	T
3	ESCRIBIR ["O",19]	19	I
4	ESCRIBIR ["R",20]	20	N
5	ESCRIBIR ["R",21]	21	R
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	A
2	ESCRIBIR ["T",18]	18	T
3	ESCRIBIR ["O",19]	19	I
4	ESCRIBIR ["R",20]	20	N
5	ESCRIBIR ["R",21]	21	G
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

→

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	A
2	ESCRIBIR ["T",18]	18	T
3	ESCRIBIR ["O",19]	19	I
4	ESCRIBIR ["R",20]	20	N
5	ESCRIBIR ["R",21]	21	G
6	ESCRIBIR ["A",22]	22	E
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	C
1	ESCRIBIR ["O", 17]	17	A
2	ESCRIBIR ["T", 18]	18	T
3	ESCRIBIR ["O", 19]	19	I
4	ESCRIBIR ["R", 20]	20	N
5	ESCRIBIR ["R", 21]	21	G
6	ESCRIBIR ["A", 22]	22	A
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	



Una vez que todos hayan descubierto que al finalizar la ejecución de los programas se habrán impresos dos hojas con el texto *CATINGA*, les preguntamos: “El propósito de uno de los programas era imprimir la palabra *cotorra* y, el del otro, *laringe*. Ninguno logró su cometido. ¿Qué pasó? ¿Por qué la impresora terminó sacando dos hojas con *catinga*?”. Guiamos la discusión de forma tal de arribar a la conclusión de que, al escribir en la misma porción de la memoria, los programas interfirieron entre sí.

Continuamos: “Dos procesos, en principio, no deberían poder escribir en las mismas celdas de memoria. Si así lo hiciesen, el resultado de sus ejecuciones dependería de cómo el *scheduler* intercale las instrucciones. ¿Se les ocurre una solución a este problema?”. Es esperable que algún estudiante responda que una solución consiste en que cada programa tenga una porción de memoria de uso exclusivo.<sup>1</sup> Copiamos entonces el siguiente esquema en el pizarrón:

<sup>1</sup> Hay circunstancias en las que dos o más procesos requieren comunicarse y lo hacen compartiendo segmentos de la memoria. Sin embargo, en general los procesos son independientes y cada uno tiene su propio espacio de memoria de uso exclusivo, tal como se presenta en la actividad. Por simplicidad, sugerimos evitar hablar sobre comunicación entre procesos.

MEMORIA			
COTORRA VOLÁ		LARÁ LARÁ LARINGE	
Programa	Datos	Programa	Datos

Cada programa tiene un espacio de memoria de uso exclusivo

Luego les comentamos: “Cuando presionamos el ícono de una aplicación en nuestros teléfonos o hacemos doble clic sobre uno en nuestras computadoras, el sistema operativo detecta que hay una solicitud para que un programa comience a ejecutarse. En ese momento, define en qué porción de la memoria se cargará el programa y cuál será su espacio de datos. Recién luego, una vez cargado en la memoria, comienzan a ejecutarse sus instrucciones”.

### CIERRE

Les contamos a los estudiantes que la porción de la memoria que se le asigna a un proceso se conoce como **espacio de direcciones** y delimita el rango de direcciones que está disponible para el proceso que comienza a ejecutarse. De este modo, todas las direcciones de memoria que sean referenciadas desde el programa serán relativas a este espacio: por ejemplo, la dirección 16 de un proceso no será la misma que la 16 de otro proceso, y así se evita la interferencia.

---

NOMBRE Y APELLIDO:

CURSO:

FECHA:

# ADMINISTRACIÓN DEL ESPACIO

Esta es la vieja y querida Bet & Rob compuimpresora, comúnmente llamada La 2x1. Se trató de uno de los primeros modelos de computadora con impresora integrada. El lenguaje para programarla incluía solo dos instrucciones: una para escribir un carácter en una posición de la memoria, y otra para imprimir todo el contenido de un cierto rango de direcciones de memoria.



## Lenguaje de la Bet & Rob compuimpresora

**ESCRIBIR [c, n]**: escribe el carácter *c* en la posición *n* de la memoria, donde *c* y *n* se reemplazan, respectivamente, por un carácter y un número cada vez que la instrucción es invocada.

**IMPRIMIR [n<sub>1</sub>, n<sub>2</sub>]**: imprime los caracteres que se encuentran entre las posiciones *n<sub>1</sub>* y *n<sub>2</sub>* de la memoria, donde *n<sub>1</sub>* y *n<sub>2</sub>* se reemplazan por un número cada vez que la instrucción es invocada.

A modo de ejemplo, mirá lo que sucede al ejecutar el programa que te mostramos a continuación.

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["Ñ", 16]	16	
1	ESCRIBIR ["U", 17]	17	
2	IMPRIMIR [16, 17]	18	
3		19	
4		20	



MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["Ñ", 16]	16	Ñ
1	ESCRIBIR ["U", 17]	17	
2	IMPRIMIR [16, 17]	18	
3		19	
4		20	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["Ñ", 16]	16	Ñ
1	ESCRIBIR ["U", 17]	17	U
2	IMPRIMIR [16, 17]	18	
3		19	
4		20	



NOMBRE Y APELLIDO:

CURSO:

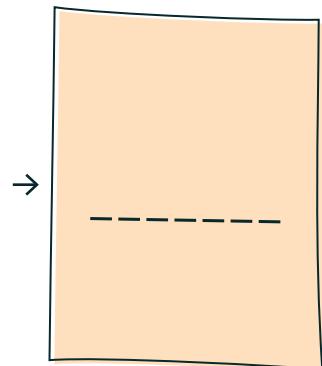
FECHA:

1. Acá hay dos programas para la compuimpresora: *Cotorra volá* y *Lara lara laringe*. Fijate qué hace cada uno y completá tanto el espacio de datos de la memoria al finalizar el programa como la palabra que imprime.

### Cotorra volá

MEMORIA	
ESPACIO DE PROGRAMA	
0	ESCRIBIR ["C",16]
1	ESCRIBIR ["O",17]
2	ESCRIBIR ["T",18]
3	ESCRIBIR ["O",19]
4	ESCRIBIR ["R",20]
5	ESCRIBIR ["R",21]
6	ESCRIBIR ["A",22]
7	IMPRIMIR [16,22]
8	

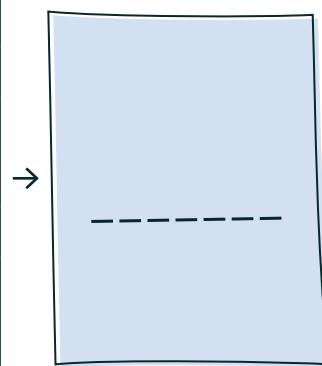
MEMORIA	
ESPACIO DE PROGRAMA	
0	ESCRIBIR ["C",16]
1	ESCRIBIR ["O",17]
2	ESCRIBIR ["T",18]
3	ESCRIBIR ["O",19]
4	ESCRIBIR ["R",20]
5	ESCRIBIR ["R",21]
6	ESCRIBIR ["A",22]
7	IMPRIMIR [16,22]
8	



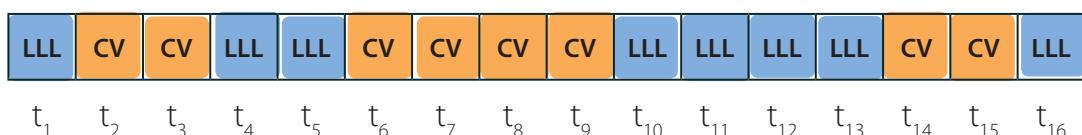
### Lará lará laringe

MEMORIA	
ESPACIO DE PROGRAMA	
0	ESCRIBIR ["L",16]
1	ESCRIBIR ["A",17]
2	ESCRIBIR ["R",18]
3	ESCRIBIR ["I",19]
4	ESCRIBIR ["N",20]
5	ESCRIBIR ["G",21]
6	ESCRIBIR ["E",22]
7	IMPRIMIR [16,22]
8	

MEMORIA	
ESPACIO DE PROGRAMA	
0	ESCRIBIR ["L",16]
1	ESCRIBIR ["A",17]
2	ESCRIBIR ["R",18]
3	ESCRIBIR ["I",19]
4	ESCRIBIR ["N",20]
5	ESCRIBIR ["G",21]
6	ESCRIBIR ["E",22]
7	IMPRIMIR [16,22]
8	



2. ¡La Bet & Rob compuimpresora viene con un sistema operativo multitarea! Ahora están ambos programas en la memoria para ser ejecutados. El scheduler dividió el uso del procesador en turnos y los asignó a los procesos de la siguiente manera:<sup>1</sup>



<sup>1</sup> LLL se refiere al programa *Lará lará laringe* y CV a *Cotorra volá*.

NOMBRE Y APELLIDO:

CURSO:

FECHA:

La primera instrucción que se ejecutará es de *Lará lará laringe*, luego dos de *Cotorra volá*, y así siguiendo. Completá la evolución de la ejecución siguiendo el orden definido por el *scheduler*. Para que veas cómo hacerlo, te mostramos a continuación el efecto de la ejecución de las tres primeras instrucciones.

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	L
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

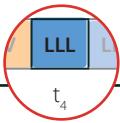
MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	C
1	ESCRIBIR ["O",17]	17	O
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

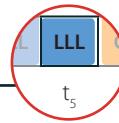
NOMBRE Y APELLIDO:

CURSO:

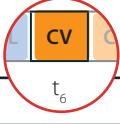
FECHA:



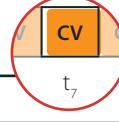
MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	



MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	



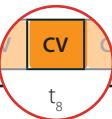
MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	



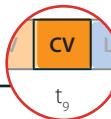
NOMBRE Y APELLIDO:

CURSO:

FECHA:

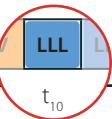


MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	
1	ESCRIBIR ["O", 17]	17	
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	

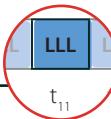


MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	
1	ESCRIBIR ["O", 17]	17	
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	

→



MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	
1	ESCRIBIR ["O", 17]	17	
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	



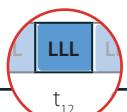
MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C", 16]	16	
1	ESCRIBIR ["O", 17]	17	
2	ESCRIBIR ["T", 18]	18	
3	ESCRIBIR ["O", 19]	19	
4	ESCRIBIR ["R", 20]	20	
5	ESCRIBIR ["R", 21]	21	
6	ESCRIBIR ["A", 22]	22	
7	IMPRIMIR [16, 22]	23	
8	ESCRIBIR ["L", 16]	24	
9	ESCRIBIR ["A", 17]	25	
10	ESCRIBIR ["R", 18]	26	
11	ESCRIBIR ["I", 19]	27	
12	ESCRIBIR ["N", 20]	28	
13	ESCRIBIR ["G", 21]	29	
14	ESCRIBIR ["E", 22]	30	
15	IMPRIMIR [16, 22]	31	

→

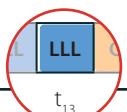
NOMBRE Y APELLIDO:

CURSO:

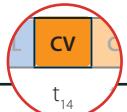
FECHA:



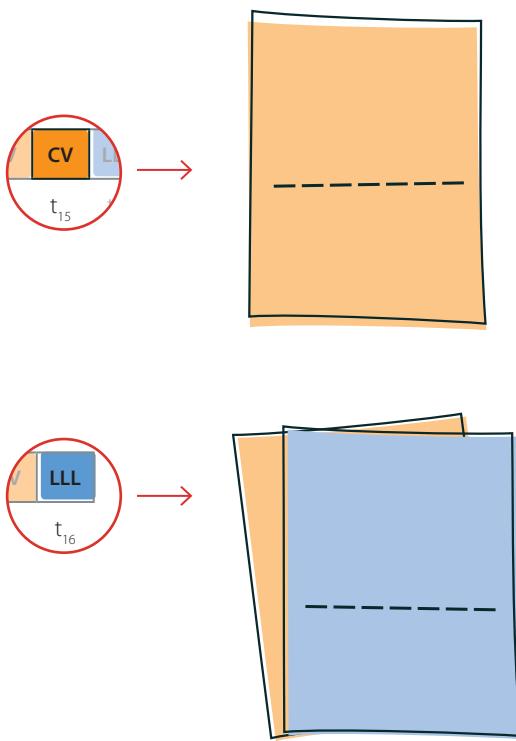
MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	



MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	

MEMORIA			
ESPACIO DE PROGRAMA		ESPACIO DE DATOS	
0	ESCRIBIR ["C",16]	16	
1	ESCRIBIR ["O",17]	17	
2	ESCRIBIR ["T",18]	18	
3	ESCRIBIR ["O",19]	19	
4	ESCRIBIR ["R",20]	20	
5	ESCRIBIR ["R",21]	21	
6	ESCRIBIR ["A",22]	22	
7	IMPRIMIR [16,22]	23	
8	ESCRIBIR ["L",16]	24	
9	ESCRIBIR ["A",17]	25	
10	ESCRIBIR ["R",18]	26	
11	ESCRIBIR ["I",19]	27	
12	ESCRIBIR ["N",20]	28	
13	ESCRIBIR ["G",21]	29	
14	ESCRIBIR ["E",22]	30	
15	IMPRIMIR [16,22]	31	



NOMBRE Y APELLIDO:

CURSO:

FECHA:

¿Qué imprimió la compuimpresora? ¿Por qué?

---

---

---

3. Describí una solución para que los distintos procesos no interfieran entre sí.

---

---

---

---

---



#### PARA NO PISARSE

Al correr varios programas a la vez, cada uno tiene que tener su propio espacio en la memoria. De este modo, no interferirá el uno con el otro.

MEMORIA			
COTORRA VOLÁ		LARÁ LARÁ LARINGE	
Programa	Datos	Programa	Datos



#### ESPACIO DE DIRECCIONES

La porción de la memoria que se le asigna a un proceso se conoce como **espacio de direcciones** y delimita el rango de direcciones que está disponible para el proceso que comienza a ejecutarse. De este modo, todas las direcciones de memoria que sean referenciadas desde el programa serán relativas a este espacio. Por ejemplo, la dirección 16 de un proceso no será la misma que la 16 de otro proceso, y así se evita la interferencia.

# GLOSARIO

**almacenamiento.** Término que se usa para referirse tanto al proceso como al dispositivo que sirve para guardar datos digitales, de forma temporal o permanente.

**alternativa condicional.** Estructura que permite ejecutar acciones sujetas a una condición. Consta de dos partes: una condición y una acción (o varias), que se ejecuta(n) si se cumple la condición.

**archivo.** Abstracción provista por el sistema operativo para que los humanos y los programas puedan agrupar lógicamente y manipular información que resulte de interés.

**arquitectura de von Neumann.** Modelo conceptual de computadora, con una unidad central de procesamiento y una memoria en la que residen datos y programas, y que se comunica con el exterior mediante dispositivos de entrada y salida. Fue ideado por el matemático de origen austrohúngaro John von Neumann en el año 1945.

**autenticación.** Verificación de la identidad de una persona o proceso.

**bit.** Acrónimo de **binary digit**, es decir, ‘dígito binario’. Es un dígito del sistema de numeración binario que tiene solo dos valores, el 0 y el 1. Toda información digitalizada se representa con un conjunto de bits y por eso se lo conoce como la unidad mínima de información.

**booleano.** Tipo de dato o expresión con solo dos valores posibles: verdadero o falso.

**caching.** Estrategia utilizada por las computadoras para disminuir el tiempo de acceso a los datos con los que trabajan los programas.

**cadena de caracteres.** También llamada *string*. Secuencia de letras, números u otros símbolos. Una cadena puede

representar, por ejemplo, un nombre, una dirección o el título de una canción. Algunas operaciones generalmente asociadas con cadenas son: longitud, concatenación y subcadena.

**celular.** Dispositivo inalámbrico electrónico que se conecta a una red celular o está preparado para tener acceso a la telefonía celular. También se lo denomina (*teléfono*) móvil.

**Ciencias de la Computación.** Disciplina que estudia las computadoras y los procesos algorítmicos, incluyendo sus principios, los diseños de *hardware* y *software*, su implementación y su impacto en la sociedad.

**cifrado.** Procedimiento que aumenta la seguridad de datos electrónicos mediante la codificación del contenido, de manera que solo pueda leerlo la parte autorizada que cuente con la clave para decodificarlo.

**ciudadanía digital.** Conjunto de normas de comportamiento apropiado y responsable con respecto al uso de la tecnología.

**compilación.** Proceso llevado a cabo por el compilador.

**compilador.** Pieza de *software* que, tomando como entrada un programa escrito en un lenguaje de alto nivel, genera como salida otro semánticamente equivalente con instrucciones en lenguaje ensamblador.

**computación.** Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de computadoras.

**computadora.** Máquina o dispositivo físico que realiza procesos, cálculos y sigue instrucciones provistas por programas. Recibe, procesa y genera información. Son ejemplos de computadoras las PC y los teléfonos.

**conurrencia.** Propiedad de los sistemas de computación que permite que varios procesos se ejecuten simultáneamente.

**condición.** Expresión que puede ser verdadera o falsa.

**conectividad.** Capacidad de un programa o dispositivo de conectarse con otros programas y dispositivos.

**contraseña.** Palabra solo conocida por una persona para acceder a algún sitio o servicio.

**controlador.** Pieza de *software* que se ocupa de la interacción entre las aplicaciones y algún dispositivos de entrada y salida de la computadora. Proporcionan una interfaz sencilla para comunicarnos con los componentes físicos de una computadora, sin entrar en detalles.

**CPU.** Sigla del inglés *central processing unit*, ‘unidad central de procesamiento’. Núcleo de la computadora, encargado de ejecutar las instrucciones de los programas, intercambiando información con la memoria. También se lo denomina *microporcesador*.

**datos.** Información recopilada y usada como referencia o para hacer análisis. Los datos pueden ser digitales o no digitales, y pueden presentarse de muchas maneras, que incluyen números, texto, imágenes, sonidos o video.

**datos personales.** Información que permite identificar a una persona, como, por ejemplo, el número de documento, la dirección, el teléfono, una foto, etc.

**descomponer.** Desglosar un problema o un sistema en distintas partes.

**digital.** Característica de la tecnología electrónica que usa valores discretos, generalmente 0 y 1, para generar, almacenar y procesar datos.

**disco rígido rotacional.** Dispositivo magnético de almacenamiento permanente de datos. Por lo general, posee varios platos que giran a gran velocidad y un brazo con un cabezal que lee y escribe en la superficie de los platos, todo esto dentro de un contenedor hermético.

**discos de estado sólido.** Discos cuya composición interna no tiene ningún componente mecánico, con una estructura más parecida a la de un *pendrive* que a la de un disco rígido rotacional. En ellos, se accede a los datos a muy alta velocidad.

**dispositivo de entrada.** Componente de *hardware* que permite que ingrese información a una computadora para ser procesada. Por ejemplo, un teclado, un ratón o un micrófono.

**dispositivo de salida.** Componente de *hardware* que permite que la computadora comunique al exterior el resultado obtenido tras un procesamiento. Por ejemplo, una pantalla, una impresora o unos parlantes.

**driver.** Ver **controlador**.

**ejecutar.** Llevar a cabo, de forma mecánica y ordenada, las instrucciones que forman parte de un programa.

**encriptación.** Ver **cifrado**.

**ensamblador.** Pieza de *software* que toma como entrada un programa en lenguaje ensamblador y genera como salida un programa equivalente en lenguaje de máquina.

**entorno de programación.** Herramienta para llevar a cabo el desarrollo de programas en una computadora.

**entrada.** Datos, señales u órdenes que recibe una computadora.

**evento.** Cualquier ocurrencia identificable que tenga un significado para el sistema, el *hardware* o el *software*. Los eventos generados por los usuarios incluyen pulsaciones de teclas o clics del ratón.

**gabinete.** Carcasa, por lo general metálica, que contiene los principales componentes de la computadora, como la unidad central de procesamiento, la memoria, unidades de almacenamiento –como discos rígidos, por ejemplo–, etc.

**hardware.** Conjunto de componentes físicos que conforman un sistema informático, una computadora o un dispositivo informático.

**implementación.** Proceso que consiste en expresar el diseño de una solución en un lenguaje de programación que puede ejecutarse en un dispositivo de cómputo.

**información.** La materia prima con la que trabajan las computadoras; se trata de conocimiento expresado de alguna forma particular. Algunas de las formas que puede tomar la información son: números, texto, sonidos, imágenes o videos.

**instrucción.** Cada una de las indicaciones que forman parte de un programa. En el caso de las instrucciones que componen un programa, deben poder ser realizadas mecánicamente por la máquina que las interpreta.

**lenguaje de máquina.** Conjunto de instrucciones que interpreta de manera directa el microprocesador o la CPU.

**lenguaje de programación.** Lenguaje que se utiliza para escribir programas. Brinda una manera de describir, sin ambigüedades, una secuencia de instrucciones elegidas de un conjunto predefinido.

**lenguaje ensamblador.** Lenguaje de programación de bajo nivel cuyas instrucciones representan las acciones básicas

que las computadoras, microprocesadores, microcontroladores y otros circuitos integrados son capaces de realizar.

**memoria.** Componente de *hardware* que almacena información.

**memoria caché.** Módulo de memoria dedicada a los datos usados o solicitados con más frecuencia para su recuperación a gran velocidad.

**memoria RAM.** Sigla del inglés *random access memory*, ‘memoria de acceso aleatorio’. Memoria que permite a la CPU almacenar, leer y modificar valores. Es una memoria no permanente o volátil que, si se interrumpe el suministro de energía, pierde su contenido.

**memoria volátil.** Almacenamiento temporal utilizado por dispositivos informáticos.

**modularidad.** Característica de un *software* que ha sido dividido en partes más pequeñas.

**núcleos (de un procesador).** Unidades de ejecución independientes dentro de una CPU.

**offline.** En español, ‘fuera de línea’. Indica que un dispositivo, *software* o usuario está desconectado de Internet.

**online.** En español, ‘en línea’. Indica que un dispositivo, *software* o usuario está conectado a Internet.

**parámetro.** Variable de un tipo especial que se usa en un procedimiento para referirse a la parte de los datos que se reciben como entradas.

**periférico.** Unidad de *hardware* que provee una o más funciones de computación dentro de un sistema computacional. Puede proveer la entrada de datos a la computadora, aceptar la salida de datos o ambas.

**phishing.** Utilización de sitios falsos para robar la información de inicio de sesión de usuarios que la ingresan pensando que se encuentran en un sitio real.

**píxel.** Superficie homogénea más pequeña que compone una imagen digital. Se define por su brillo y color. En las pantallas de las computadoras, una imagen se muestra casi siempre mediante una cuadrícula de píxeles, cada uno ajustado al color requerido.

**placa madre.** Componente de **hardware** al que se conectan los demás, de manera que permite su intercomunicación. Cuenta con espacios dedicados especialmente al procesador y a la memoria, y con conectores a los que se pueden encajar múltiples dispositivos de entrada y de salida.

**proceso.** Programa en ejecución.

**procedimiento.** Módulo independiente de un programa que lleva a cabo una tarea específica. Puede ser referenciado desde otros puntos del programa.

**procesador.** Ver **CPU**.

**programa.** Conjunto de instrucciones escritas en un lenguaje de programación que la computadora ejecuta para lograr un objetivo particular, como el tratamiento de textos, el diseño de gráficos, la resolución de problemas matemáticos, el manejo de bancos de datos, entre otros.

**red social (digital).** Plataformas de Internet de las que forman parte un conjunto de usuarios que están relacionados de acuerdo a algún criterio (relación profesional, amistad, etc.).

**registros de CPU.** Memoria pequeña y rápida dentro de la CPU que permite almacenar instrucciones y datos temporalmente mientras un programa se está ejecutando.

Cada registro tiene un nombre particular para poder ser leído o escrito; ese nombre está formado típicamente por unas pocas letras en mayúscula, por ejemplo, registro RAX.

**repetición.** Representación, dentro de un programa, del hecho de que una instrucción o una serie de instrucciones debe ejecutarse repetidamente una cierta cantidad de veces.

**scheduler.** Programa del sistema operativo que se encarga de dividir el tiempo de uso del procesador y determinar qué proceso se ejecuta en cada turno.

**seguridad informática.** Protección contra el acceso no autorizado a los recursos informáticos o su alteración, mediante el uso de tecnología, procesos y capacitación.

**sistema de archivos.** Subsistema del sistema operativo que permite crear, leer y modificar archivos, además de organizarlos en carpetas.

**sistema operativo.** Conjunto de programas que se ubica entre los programas que usamos habitualmente y el **hardware** de la computadora. Se ocupa, entre otras cosas, de la interacción de los programas con todos los periféricos de entrada y salida, de la administración de la unidad central de procesamiento y la memoria, de la organización de los archivos, etc.

**software.** Conjunto de programas que se ejecutan en una computadora u otro dispositivo informático.

**string.** Ver **cadena de caracteres**.

**unidad central de procesamiento.** Ver **CPU**.

**variable.** Nombre que denota una porción de memoria en la que se almacena información. Usando variables, los programas pueden leer, escribir y modificar datos.

## FOTOS E ILUSTRACIONES TÉCNICAS

USB-C: [https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/LeTV\\_X600\\_USB\\_Type\\_C\\_port.jpg/800px-LeTV\\_X600\\_USB\\_Type\\_C\\_port.jpg](https://upload.wikimedia.org/wikipedia/commons/thumb/6/63/LeTV_X600_USB_Type_C_port.jpg/800px-LeTV_X600_USB_Type_C_port.jpg) / Disco externo: Freepik.com /  
Disco SSD: [https://upload.wikimedia.org/wikipedia/commons/thumb/5/5e/Vertex\\_2\\_Solid\\_State\\_Drive\\_by\\_OCZ-top\\_oblique\\_PNr%C2%B00307.jpg/1920px-Vertex\\_2\\_Solid\\_State\\_Drive\\_by\\_OCZ-top\\_oblique\\_PNr%C2%B00307.jpg](https://upload.wikimedia.org/wikipedia/commons/thumb/5/5e/Vertex_2_Solid_State_Drive_by_OCZ-top_oblique_PNr%C2%B00307.jpg/1920px-Vertex_2_Solid_State_Drive_by_OCZ-top_oblique_PNr%C2%B00307.jpg) / Placa de video: <https://upload.wikimedia.org/wikipedia/commons/thumb/6/62/AGP-Video-Card.jpg/1024px-AGP-Video-Card.jpg> / Freepik.com

Program.AR, Fundación Sadosky  
Av. Córdoba 832, 5º piso.  
Ciudad Autónoma de Buenos Aires, Argentina.

Ciencias de la computación para el aula : 2do. ciclo de secundaria / Claudia Banchoff Tzancoff ... [et al.] ; contribuciones de Vanessa Aybar Rosales ... [et al.] ; compilado por Silvina Justianovich ; coordinación general de Vanina Klinkovich ; Hernán Czemerinski; editado por Ignacio David Miller ; Alejandro Palermo ; fotografías de Facundo Manini ; ilustrado por Jaqueline Schaab ; Juan Martín Serrovalle ; Klinko ; prólogo de María Belén Bonello ; Fernando Pablo Schapachnik. - 1a edición para el profesor - Ciudad Autónoma de Buenos Aires : Fundación Sadosky, 2019. Libro digital, PDF - (Ciencias de la Computación para el aula / Klinkovich, Vanina; Czemerinski, Hernán; 4)

Archivo Digital: descarga  
ISBN 978-987-27416-8-6

1. Informática. 2. Programación. 3. Educación Secundaria. I. Banchoff Tzancoff, Claudia. II. Aybar Rosales, Vanessa, colab. III. Justianovich, Silvina, comp. IV. Klinkovich, Vanina, coord. V. Czemerinski, Hernán, coord. VI. Miller, Ignacio David, ed. VII. Palermo, Alejandro, ed. VIII. Manini, Facundo, fot. IX. Schaab, Jaqueline, ilus. X. Serrovalle, Juan Martín, ilus. XI. Klinko, ilus. XII. Bonello, María Belén, prolog. XIII. Schapachnik, Fernando Pablo, prolog. CDD 004.0712

Queda hecho el depósito que dispone la Ley 11.723  
Ediciones Colihue.

 Primera edición: Diciembre de 2019.

El contenido del manual se distribuye bajo la licencia Creative Commons Compartir Igual.