¡Crea tu primer programa con MakeCode! Fundamentos de programación para jóvenes innovadores

Tecnología e Informática | Pensamiento Computacional | Design Thinking

Descripción

Este plan de clase está diseñado para introducir a estudiantes de secundaria, entre 12 y 15 años, en los fundamentos de la programación utilizando la plataforma MakeCode. Los estudiantes explorarán conceptos básicos de lógica computacional, secuencias, bucles y eventos a través de actividades prácticas y creativas. La relevancia de aprender programación radica en desarrollar habilidades para resolver problemas, pensar de manera lógica y fomentar la creatividad, competencias esenciales en un mundo cada vez más digitalizado. Además, el uso de MakeCode permite una experiencia visual e interactiva que conecta con sus intereses, haciendo tangible el impacto de la programación en la vida cotidiana, desde videojuegos hasta dispositivos inteligentes. A lo largo de seis sesiones, los alumnos aplicarán la metodología Design Thinking para empatizar con problemas, definir retos, idear soluciones, prototipar códigos y evaluar sus proyectos, promoviendo así un aprendizaje activo y centrado en su experiencia y contexto.

Objetivos de Aprendizaje

- Comprender y aplicar conceptos básicos de programación como secuencias, bucles y eventos mediante MakeCode.
- Diseñar y crear programas interactivos simples que respondan a estímulos y cumplan objetivos específicos.
- Analizar problemas cotidianos para definir retos de programación usando la metodología Design Thinking.
- Prototipar y evaluar soluciones digitales en equipo, fomentando la colaboración y el pensamiento crítico.
- Comunicar efectivamente el proceso y resultados de sus proyectos de programación.

Recursos Necesarios

- Computadoras o tablets con acceso a internet (1 por estudiante o pareja).
- Acceso a la plataforma MakeCode (https://makecode.microbit.org/ o similar).
- Proyector y pantalla para presentaciones y demostraciones.
- Material impreso con guías de actividades y fichas de trabajo.
- Cuadernos o libretas para anotaciones personales.
- Marcadores, hojas blancas y rotafolios para mapas mentales y lluvia de ideas.
- Cámara o dispositivo para grabar prototipos (opcional).

Requisitos Previos

- Habilidades básicas para manejar dispositivos digitales (computadora o tablet).

- Conocimiento previo elemental de lógica secuencial (por ejemplo, entender instrucciones paso a paso).
- Experiencia con uso básico de navegadores web e interacción con interfaces gráficas.
- Actitud colaborativa y disposición para el trabajo en equipo.

Actividades

Sesión 1: Introducción a la programación con MakeCode y Design Thinking

Fase de Inicio

Tiempo estimado: 15 minutos

Propósito de la sesión:

Conectar con conocimientos previos y motivar a los estudiantes para explorar la programación como una herramienta creativa y divertida.

Activación de conocimientos previos:

- **Docente:** "¿Alguna vez han pensado cómo se hacen los videojuegos que les gustan o cómo una alarma sabe cuándo sonar? Vamos a descubrirlo juntos."
- **Estudiantes:** Responden a la pregunta inicial y comparten experiencias con tecnología o juegos que involucren instrucciones o reglas.

Motivación y enganche:

- **Docente:** Muestra un video corto (2-3 minutos) con ejemplos de proyectos creados en MakeCode, como un juego simple o un semáforo digital.
- **Estudiantes:** Observan y comentan lo que les llamó la atención.

Contextualización:

- **Docente:** Explica cómo la programación puede ayudarles a resolver problemas y crear cosas que usan todos los días, conectando con su vida escolar y personal.
- **Estudiantes:** Reflexionan y comentan ejemplos de su entorno donde la programación podría ser útil.

Fase de Desarrollo

Tiempo estimado: 95 minutos

Presentación del contenido:

El docente introduce los conceptos básicos de programación (secuencias, bloques, eventos) a través de la plataforma MakeCode con ejemplos visuales y analogías sencillas.

Actividad 1: Explorando MakeCode

- **Objetivo:** Familiarizarse con la interfaz y bloques básicos de MakeCode.
- **Instrucciones:**
  - **Docente:** "Abramos MakeCode y exploremos juntos cómo crear un programa que muestre un corazón en la pantalla."
  - Guía paso a paso para crear el programa con la ayuda de imágenes proyectadas.
  - **Estudiantes:** Siguen los pasos en sus dispositivos y prueban el programa.
- **Organización:** Individual
- **Producto:** Programa básico funcional que muestra una imagen en MakeCode.
- **Tiempo:** 30 minutos
- **Rol del docente:** Apoyar dudas, observar el manejo del entorno, hacer preguntas sobre la lógica.

Actividad 2: Pensando con Design Thinking – Definir un reto

- **Objetivo:** Identificar un problema sencillo que pueda resolverse con programación.
- **Instrucciones:**
  - **Docente:** "En grupos de 3, piensen en una situación cotidiana donde un programa pueda ayudar, como una alarma, un recordatorio o un juego."
  - Los grupos discuten y anotan su problema en una hoja.
  - **Estudiantes:** Comparten ideas y eligen un reto para trabajar en las siguientes sesiones.
- **Organización:** Grupos de 3 estudiantes
- **Producto:** Enunciado del reto de programación.
- **Tiempo:** 30 minutos
- **Rol del docente:** Facilitar la discusión, guiar con preguntas enfocadas y asegurarse que el reto sea claro y factible.

Actividad 3: Primer prototipo de programación

- **Objetivo:** Crear un programa simple que responda a un evento, usando bloques básicos.
- **Instrucciones:**
  - **Docente:** "Ahora, usando MakeCode, intenten crear un programa que responda cuando presionen un botón."
  - Se muestra ejemplo y se da tiempo para experimentar.
  - **Estudiantes:** Programan en parejas, prueban sus códigos y comparten experiencias.
- **Organización:** Parejas
- **Producto:** Programa con evento funcional.
- **Tiempo:** 35 minutos
- **Rol del docente:** Guiar, hacer preguntas que fomenten la solución de problemas, apoyar con errores comunes.

Diferenciación:

- **Para estudiantes avanzados:** Proponer agregar un segundo evento o usar variables simples.
- **Para estudiantes que necesitan apoyo:** Trabajo guiado con ejemplos paso a paso y apoyo individualizado.

Transición:

Docente: Resume lo aprendido y explica que en la siguiente sesión profundizaremos en más bloques y funcionalidades para enriquecer sus proyectos.

Fase de Cierre

Tiempo estimado: 10 minutos

Síntesis:

- **Docente:** Invita a los estudiantes a compartir en un “ticket de salida” por escrito tres cosas que aprendieron hoy y una duda que tienen.
- **Estudiantes:** Escriben y entregan su ticket.

Reflexión metacognitiva:

- ¿Qué fue lo que más te gustó de crear tu primer programa?
- ¿Cómo crees que la programación puede ayudarte en la escuela o en casa?
- ¿Qué reto o problema te gustaría resolver con un programa?

Retroalimentación:

Docente: Revisa los tickets, comenta aspectos destacados y aclara dudas comunes en plenaria.

Transferencia:

Docente: Explica que en la próxima sesión continuarán creando y mejorando sus programas basados en los retos definidos.

Tarea o reto:

Explorar la plataforma MakeCode en casa y probar a crear un programa nuevo con bloques diferentes.

Evaluación

Tipo de evaluación:

- **Diagnóstica:** Inicio de la sesión 1 mediante la pregunta detonadora y observación de habilidades digitales.
- **Formativa:** Durante las actividades prácticas en cada sesión, con observación directa y retroalimentación continua.
- **Sumativa:** Al final del plan, con la presentación y evaluación del proyecto final basado en el reto definido.

Criterios de evaluación:

- Aplica conceptos básicos de programación usando MakeCode (objetivo 1).
- Diseña y ejecuta programas funcionales que respondan a eventos (objetivo 2).
- Define y contextualiza un problema para resolver con programación (objetivo 3).
- Colabora eficazmente en equipo para prototipar y evaluar soluciones (objetivo 4).
- Comunica de manera clara y coherente el proceso y resultados de su proyecto (objetivo 5).

Instrumentos sugeridos:

- Lista de cotejo para seguimiento en actividades prácticas.
- Rúbrica para evaluación del proyecto final (claridad, funcionalidad, creatividad, trabajo en equipo).
- Observación directa y registros anecdóticos durante las sesiones.
- Autoevaluación y coevaluación mediante cuestionarios breves.

Evidencias de aprendizaje:

- Programas creados en MakeCode durante las sesiones.
- Enunciados de retos y propuestas de solución documentadas.
- Presentaciones o demostraciones de proyectos finales.
- Respuestas escritas en reflexiones y tickets de salida.