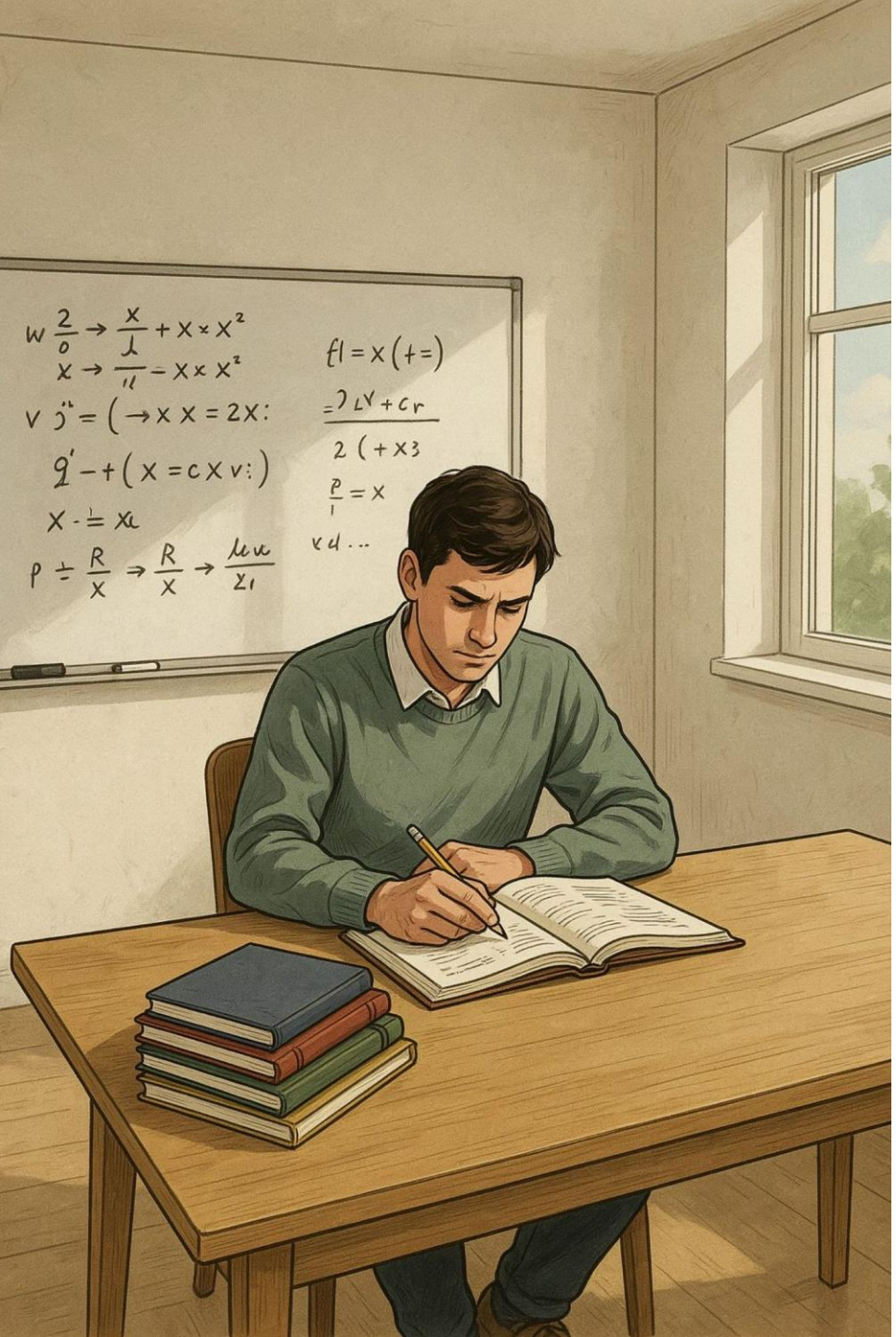


Fundamentos de Especificación Formal

DESARROLLO FORMAL DE SOFTWARE

Especificación formal y lógica: Proposiciones, Tablas de Verdad

¿Cómo expresamos las condiciones de manera precisa?

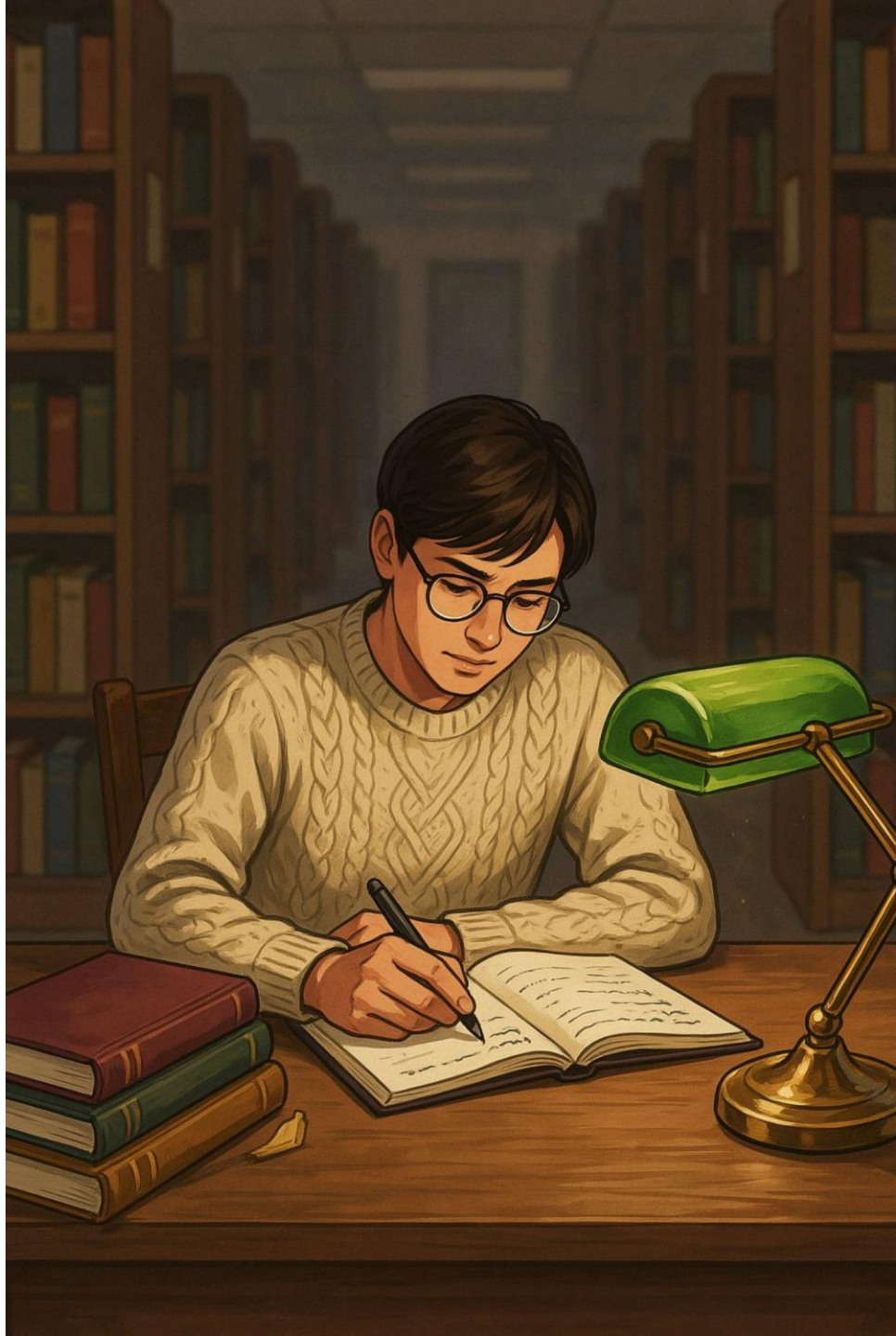
INGENIERÍA DE SISTEMAS
UNIVERSIDAD DEL PACÍFICO

Mg. Yowanna Karina Caicedo Guerrero

Objetivo de la clase

> Comprender cómo una condición expresada en lenguaje natural puede representarse de manera precisa mediante proposiciones y conectores de la lógica proposicional.

Este conocimiento será la base para avanzar hacia la **formalización de condiciones de software**: precondiciones, postcondiciones, aserciones e invariantes.



¿Dónde quedamos?

Ya vimos

- Correctitud del software
- Error, defecto y fallo
- Verificación, validación y pruebas
- Calidad y construcción

La nueva pregunta

> ¿Cómo expresamos con **precisión** aquello que el software debe cumplir?

Con **especificación formal**

¿Qué es una especificación formal?

Descripción precisa

Define propiedades, condiciones o comportamientos que debe cumplir un sistema.

Reduce ambigüedad

Elimina las imprecisiones propias del lenguaje natural.

Permite verificación

Una condición precisa puede analizarse, verificarse y comprobarse formalmente.

Hoy comenzaremos con el primer bloque: la **lógica proposicional**.

Del lenguaje natural a la lógica

> "La reserva puede realizarse si hay cupos disponibles y la solicitud es válida."

Esta frase cotidiana contiene una condición lógica que podemos representar de forma exacta.

Representación formal

P

"Hay cupos disponibles"

Q

"La solicitud es válida"

Condición

P ∧ Q — compacta y sin ambigüedad

¿Qué es una proposición?

Es un **enunciado declarativo** al que podemos asignar un valor de verdad: **Verdadero (V)** o **Falso (F)**.

☑ Son proposiciones

"Hay cupos disponibles."

"La solicitud es válida."

“El estudiante está matriculado.”

”El sistema tiene conexión”

❌ No son proposiciones

¿Hay cupos?

(pregunta)

Reserva ahora.

(orden)

Proposiciones simples y compuestas

Esta distinción es fundamental al formalizar condiciones en programas: las condiciones reales suelen ser **compuestas**.

Conectores lógicos básicos

¬P — Negación

"No P" · Invierte el valor de verdad

P ∧ Q — Conjunción

"P y Q" · Verdadera si ambas son V

P ∨ Q — Disyunción

"P o Q" · Verdadera si al menos una es V

P → Q — Implicación

"Si P, entonces Q"

P ↔ Q — Bicondicional

"P si y solo si Q"

Hoy nos concentramos especialmente en **¬, ∧ y ∨**.

Negación: ¬P

Se lee: **“No P”**

La negación **invierte el valor de verdad** de una proposición.

P = "Hay cupos disponibles"
¬P = "No hay cupos disponibles"

Si **P es V**, entonces **¬P es F**.
Si **P es F**, entonces **¬P es V**.

> La negación es el conector más simple, pero esencial para construir condiciones de guarda y exclusión en software.

Tabla de verdad

P	¬ P
V	F
F	V

Conjunción: P ∧ Q

> Se lee: **“P y Q”**
> Es verdadera **únicamente** cuando P y Q son ambas verdaderas.

P = "Hay cupos disponibles"
Q = "La solicitud es válida"

Para permitir la reserva, **ambas condiciones deben cumplirse simultáneamente**. Si falla cualquiera de las dos, la reserva no procede.

Tabla de verdad

P	Q	P ∧ Q
V	V	V
V	F	F
F	V	F
F	F	F

Disyunción: P ∨ Q

> Se lee: **“P o Q”**
> Es verdadera cuando **al menos una** de las proposiciones es verdadera.

P = "El usuario es estudiante"
Q = "El usuario es docente"

En lógica, el **"o" es inclusivo**: puede cumplirse una condición, la otra, o ambas a la vez.

Tabla de verdad

P	Q	P ∨ Q
V	V	**V**
V	F	**V**
F	V	**V**
F	F	F

Implicación: P → Q

Se lee: **"Si P entonces Q"**

Permite relacionar una condición con una consecuencia

P = "condición". Hay cupos disponibles.

Q = "consecuencia". Se puede realizar la reserva.

P → Q: "Si hay cupos disponibles, entonces la reserva puede continuar." Se establece en **una dirección.** (P → Q)

Bicondicional: P ↔ Q

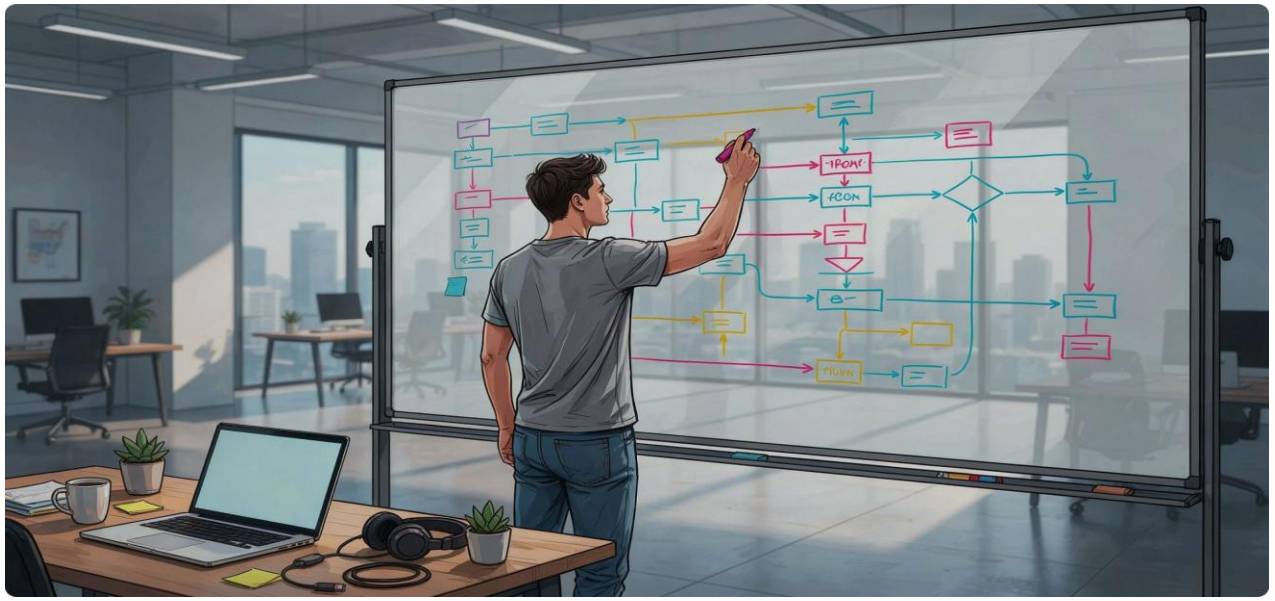
Se lee: **"P si y solo si Q"**

P y Q deben tener el mismo valor de verdad.

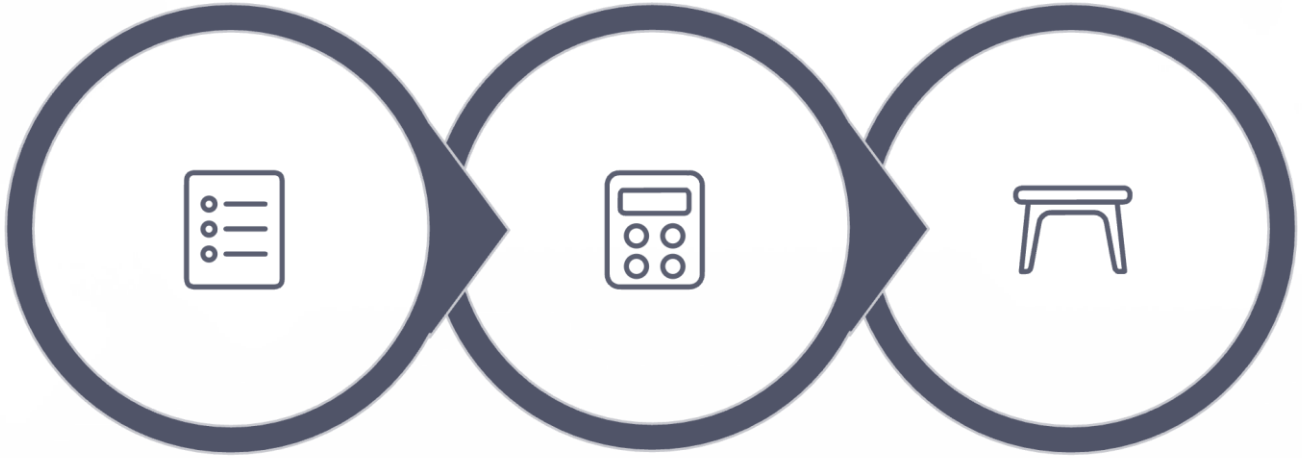
si **P** es verdadera, **Q** también debe ser verdadera

si **Q** es verdadera, **P** también debe ser verdadera.

Por eso el **bicondicional** representa una relación **en ambos sentidos.** P ↔ Q = (P → Q) ∧ (Q → P)



Tablas de verdad



Listar
VV,VF,FV,FF

Evaluar la
expresión

Leer la
columna
resultado

Con **dos proposiciones** P y Q existen **cuatro combinaciones posibles.** La tabla nos permite analizar el comportamiento de cualquier expresión compuesta de forma sistemática y exhaustiva.

Ejemplo: Tablas de Verdad

P	Q	P ∧ Q	P∨ Q	¬P
V	V			
V	F			
F	V			
F	F			

Interpretación

P: Hay cupos

Q: Solicitud válida.

> ¿Qué ocurre si hay cupos, pero la solicitud no es válida?
> → P = V, Q = F → P ∧ Q = **F**. La reserva no procede.

Relación con el software

Una condición de un programa puede expresarse como una **proposición o combinación de proposiciones**.

1

Condición natural

"Se puede reservar cuando hay cupos y la solicitud es válida"

2

Formalización

$P \wedge Q$

3

Próximos pasos

Precondiciones, postcondiciones, aserciones e invariantes

Actividad de clase

TRABAJO EN GRUPOS

1 Elige una funcionalidad

Selecciona una funcionalidad que ya esté definida en tu proyecto y escríbela.

2 Escribe cinco condiciones

Identifica cinco condiciones que debe cumplir la funcionalidad. Luego, convierte cada condición en una proposición y asígnale una letra: **P, Q, R, S, T**

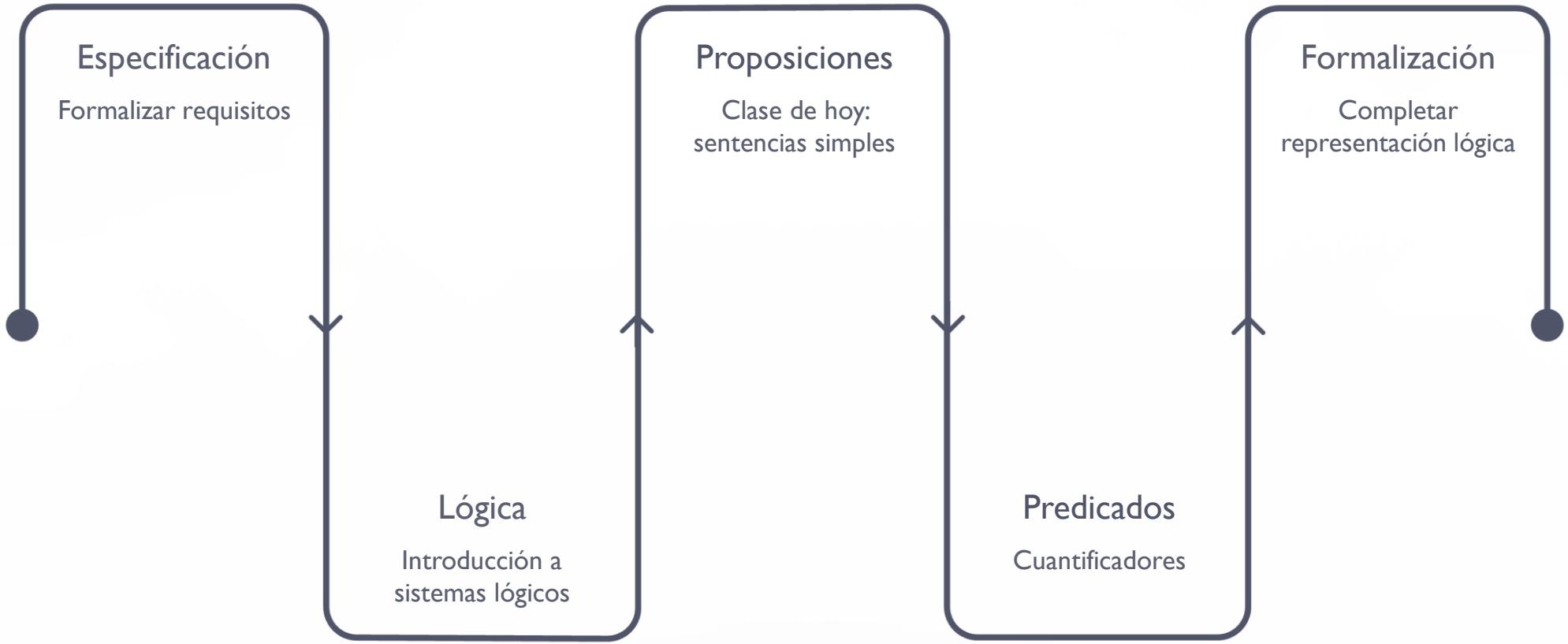
3 Construye ocho expresiones

Usando ¬, ∧ o ∨ con las proposiciones definidas. Construye tres expresiones y explica qué significa cada una en el contexto de tu funcionalidad.

4 Tabla de verdad + explicación

Con las expresiones construida, realiza su tabla de verdad y explica con palabras qué significa cada resultado en el contexto de la funcionalidad.

Cierre


Especificación
Formalizar requisitos
Lógica
Introducción a sistemas lógicos
Proposiciones
Clase de hoy: sentencias simples
Predicados
Cuantificadores
Formalización
Completar representación lógica

Pregunta de salida

> ¿Por qué la lógica puede ser útil para **desarrollar software**?

> ¿Qué diferencia hay entre decir "*el programa debe permitir reservar*" y expresar una **condición precisa** sobre cuándo puede realizarse la reserva?

Reflexiona sobre estas preguntas antes de la próxima sesión.