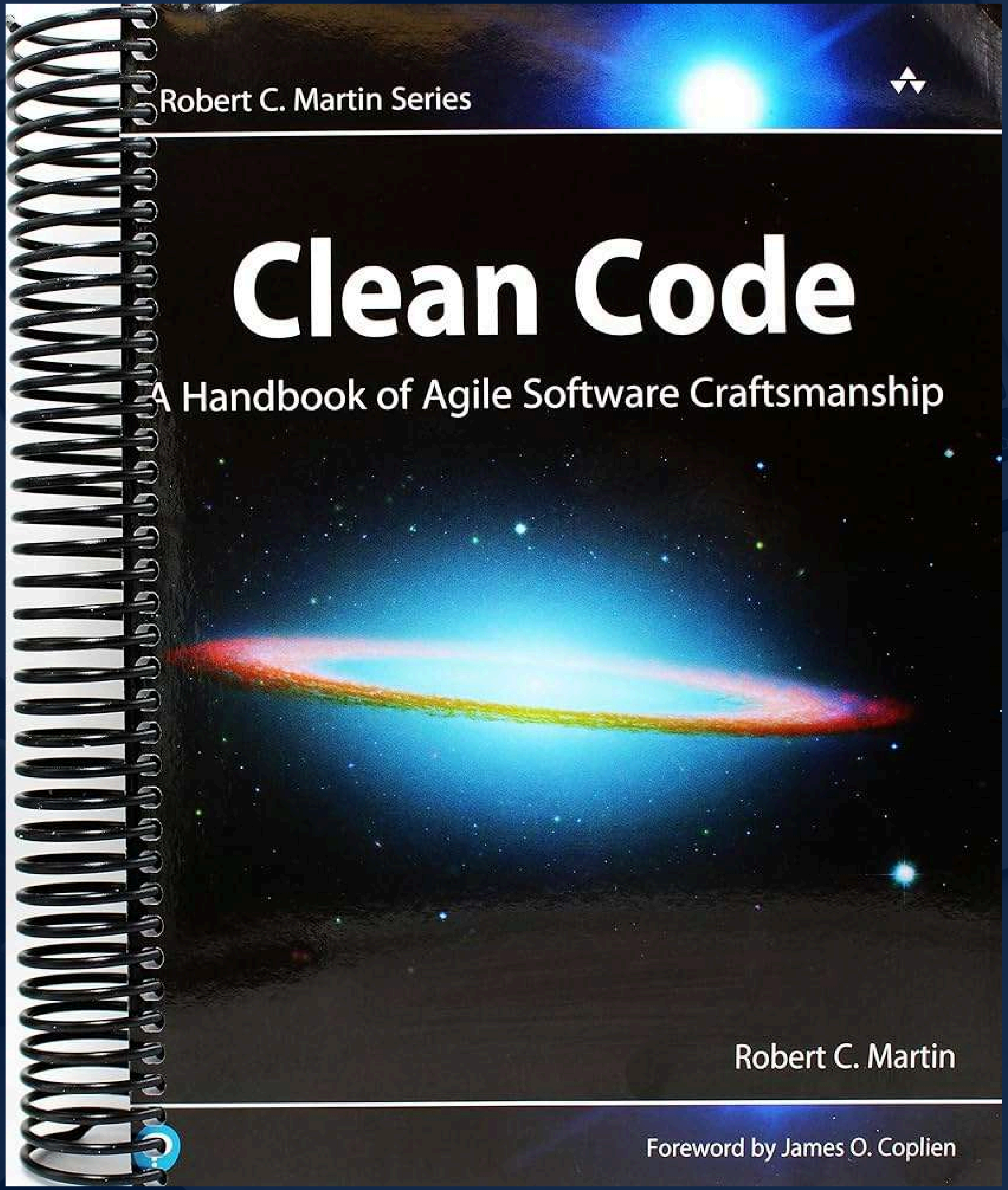
¿QUIERES MEJORAR TU CODIGO?

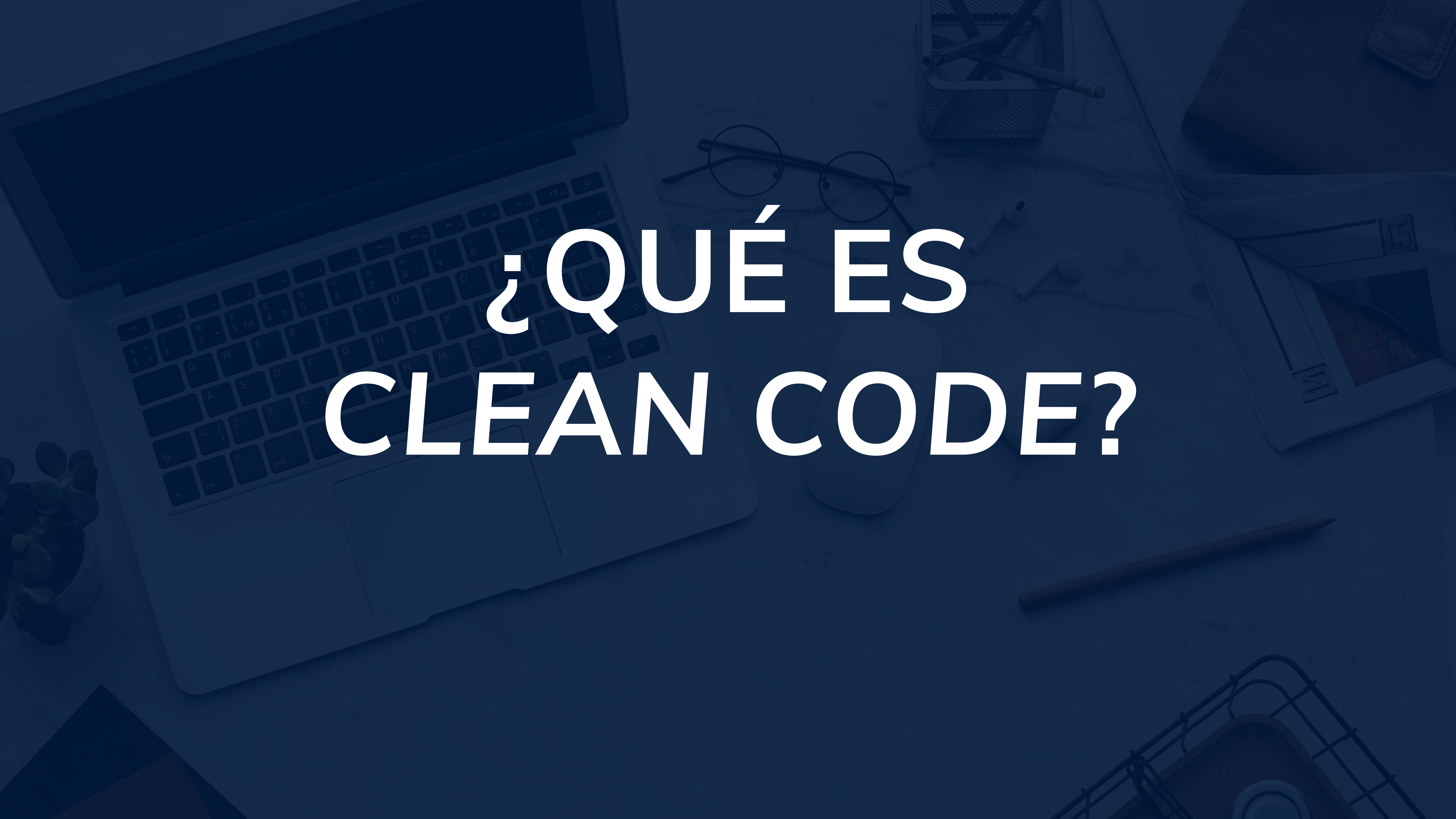
101: Clean Code y Patrones de Diseño

Peru .Net Development

Marlon E. Peña


Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Robert C. Martin
Foreword by James O. Coplien

Vamos a explorar como hacer que nuestro código sobreviva un code *review*



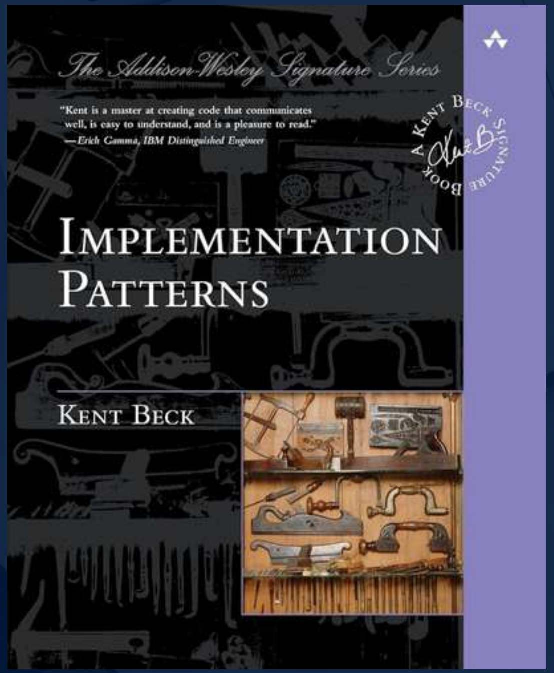
¿QUÉ ES *CLEAN CODE*?

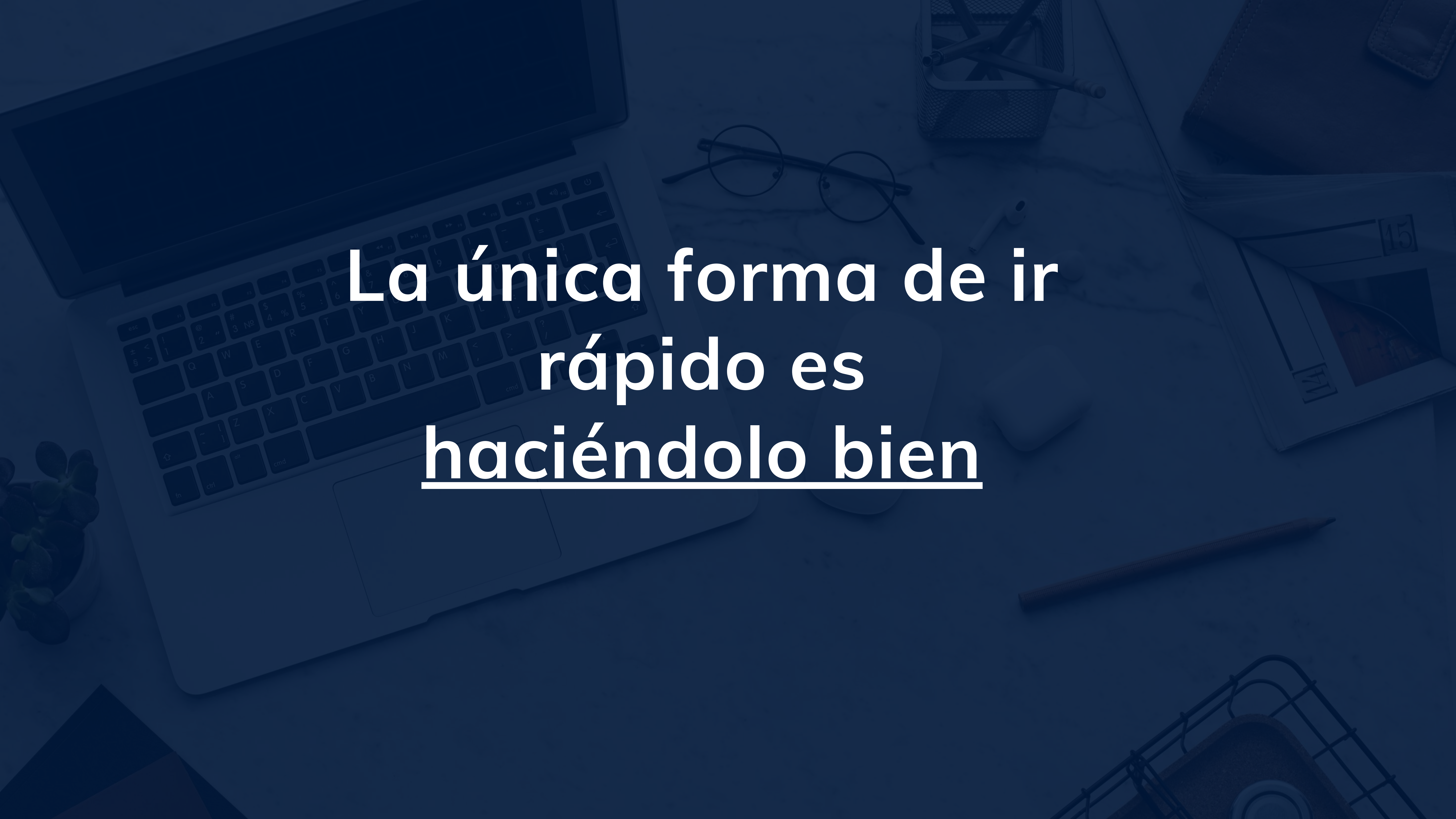
BAD CODE

¿Una premisa frágil?

"Good code matters"

-Kent Beck


The Addison-Wesley Signature Series
"Kent is a master at creating code that communicates well, is easy to understand, and is a pleasure to read."
—Erich Gamma, IBM Distinguished Engineer
A Kent Beck Signature Book
Implementation Patterns
Kent Beck

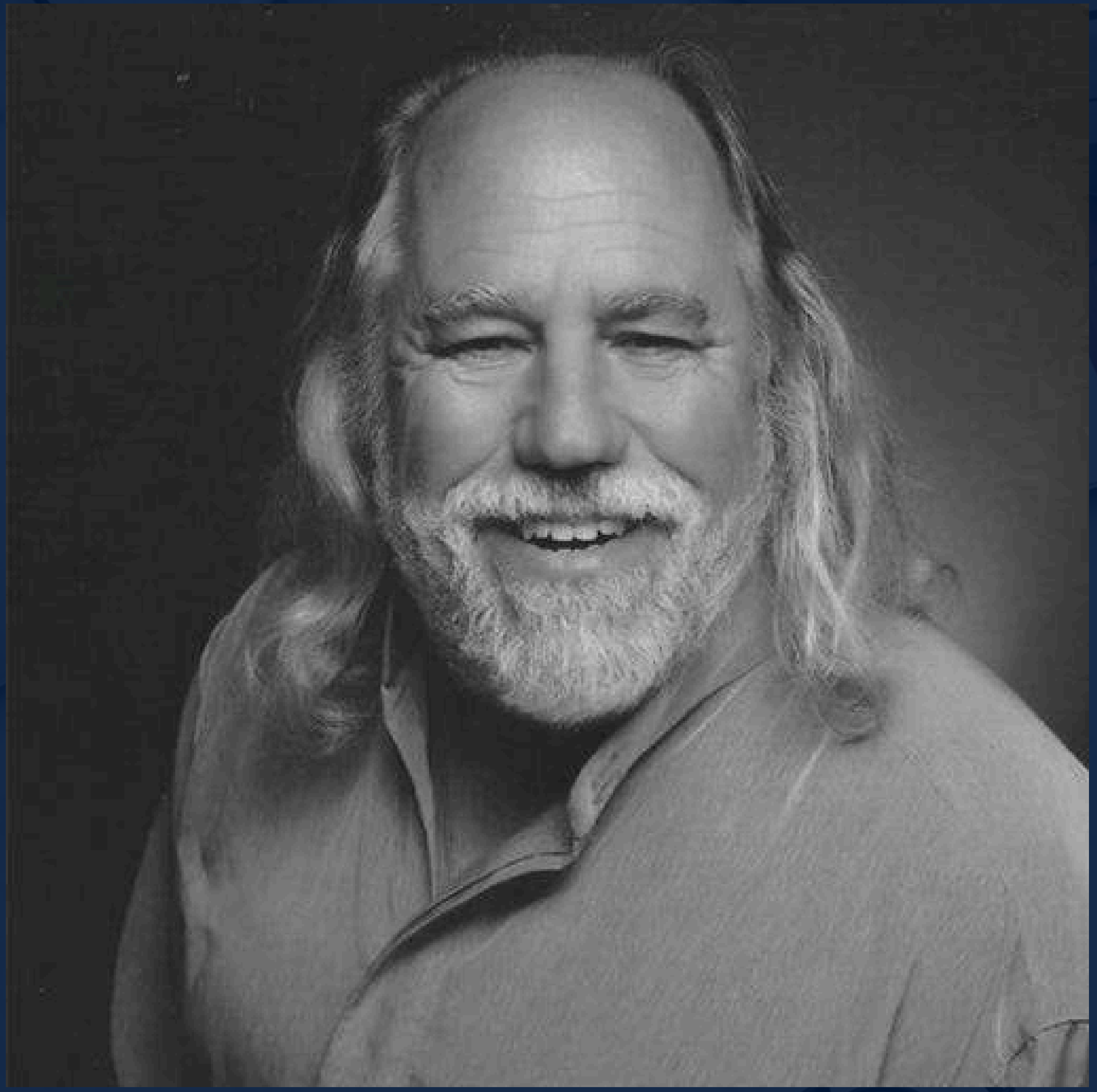


La única forma de ir rápido es haciéndolo bien

SIMPLE, DIRECTO, PROSA

“Clean code is simple and direct. Clean code reads like well-written prose...”

-Grady Booch



CON CUIDADO Y CARIÑO

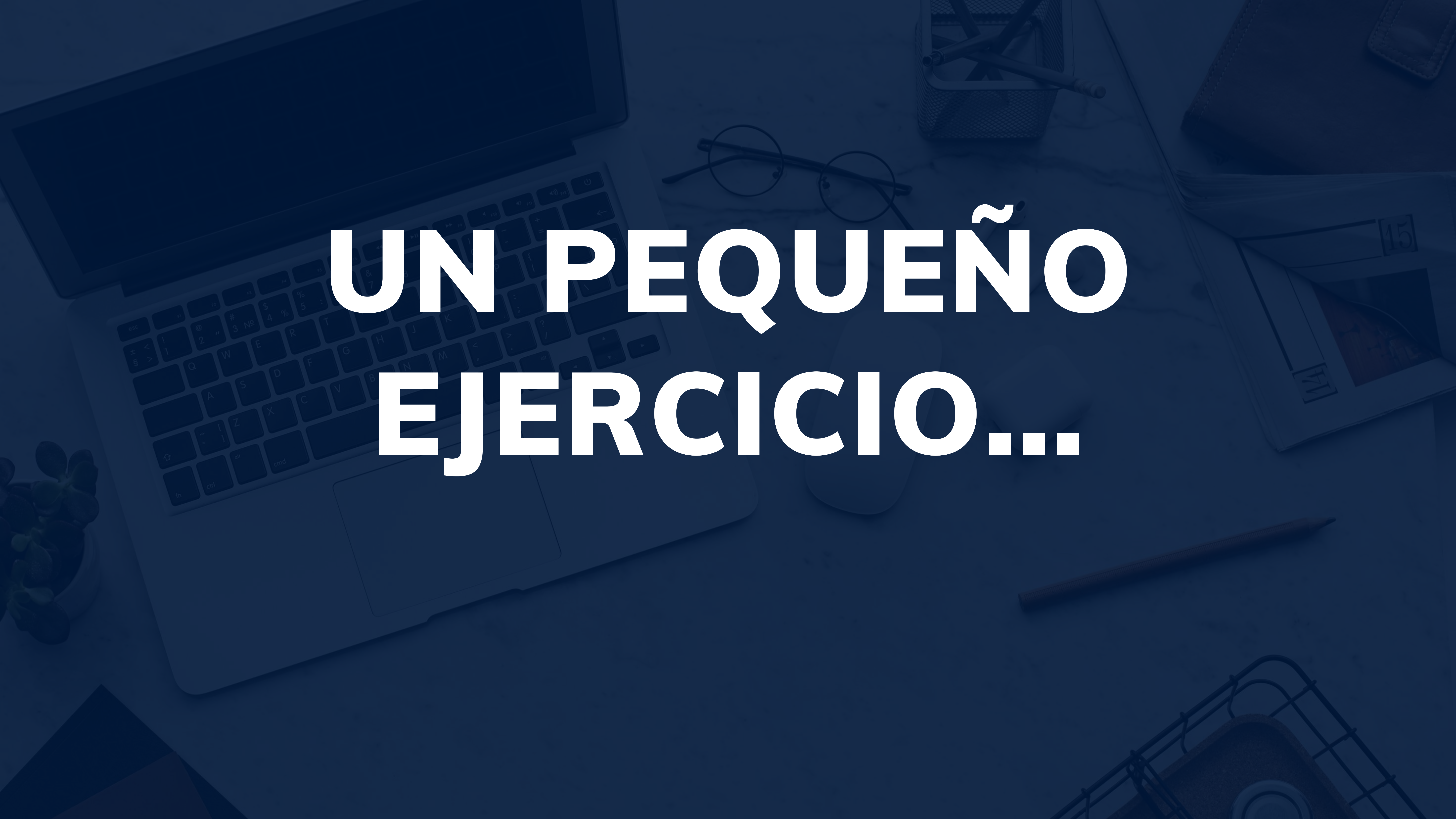
"Clean code always looks like it was written by someone who cares"

-Michael Feathers

LO QUE ESPERABAS...

"You know you are working on clean code when each routine you read turns out to be pretty much what you expected..."

-Ward Cunningham



UN PEQUEÑO EJERCICIO…

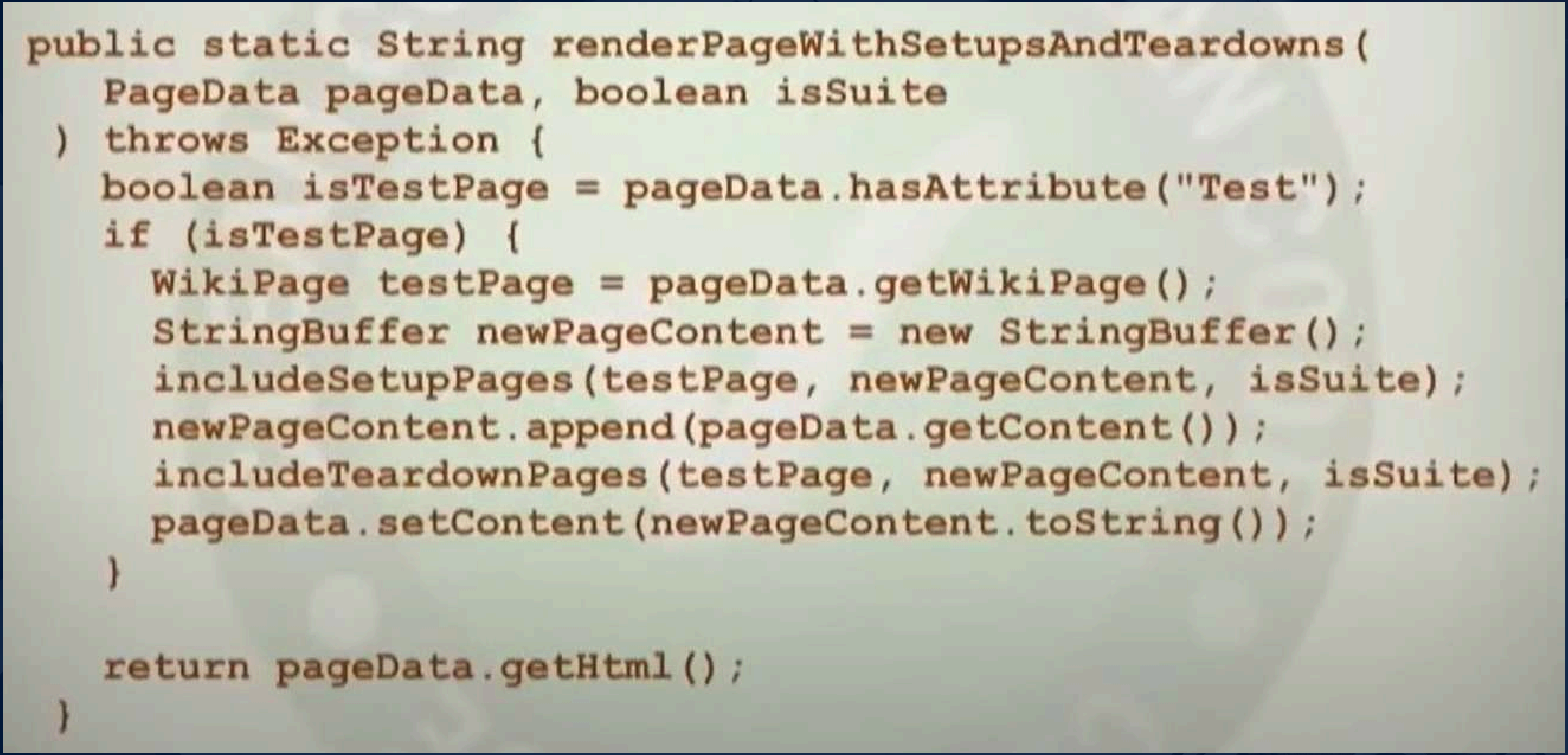
```java
public static String testableHtml(
    PageData pageData,
    boolean includeSuiteSetup
 ) throws Exception {
    WikiPage wikiPage = pageData.getWikiPage();
    StringBuffer buffer = new StringBuffer();
    if (pageData.hasAttribute("Test")) {
      if (includeSuiteSetup) {
        WikiPage suiteSetup =
          PageCrawlerImpl.getInheritedPage(
                  SuiteResponder.SUITE_SETUP_NAME, wikiPage
          );
        if (suiteSetup != null) {
          WikiPagePath pagePath =
            suiteSetup.getPageCrawler().getFullPath(suiteSetup);
          String pagePathName = PathParser.render(pagePath);
          buffer.append("!include -setup .”)
                .append(pagePathName)
                .append("\n");
        }
      }
      WikiPage setup =
        PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
```

```
  if (setup != null) {
    WikiPagePath setupPath =
      wikiPage.getPageCrawler().getFullPath(setup);
    String setupPathName = PathParser.render(setupPath);
    buffer.append("!include -setup .")
          .append(setupPathName)
          .append("\n");
  }
}
buffer.append(pageData.getContent());
if (pageData.hasAttribute("Test")) {
  WikiPage teardown =
    PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
  if (teardown != null) {
    WikiPagePath tearDownPath =
      wikiPage.getPageCrawler().getFullPath(teardown);
    String tearDownPathName = PathParser.render(tearDownPath);
    buffer.append("\n")
          .append("!include -teardown .")
          .append(tearDownPathName)
          .append("\n");
  }
  if (includeSuiteSetup) {
    WikiPage suiteTeardown =
      PageCrawlerImpl.getInheritedPage(
              SuiteResponder.SUITE_TEARDOWN_NAME,
              wikiPage
      if (suiteTeardown != null) {
        WikiPagePath pagePath =
          suiteTeardown.getPageCrawler().getFullPath(suiteTeardown);
        String pagePathName = PathParser.render(pagePath);
        buffer.append("!include -teardown .")
              .append(pagePathName)
              .append("\n");
      }
    }
  }
  pageData.setContent(buffer.toString());
  return pageData.getHtml();
}
```

LÁPICES ABAJO

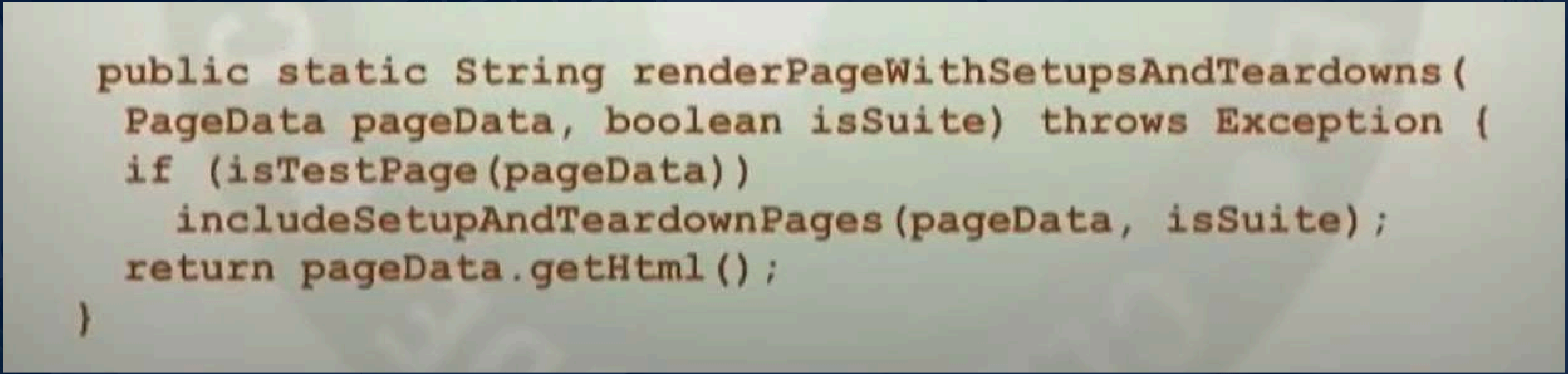
😊

REFACTORED FUNCTION

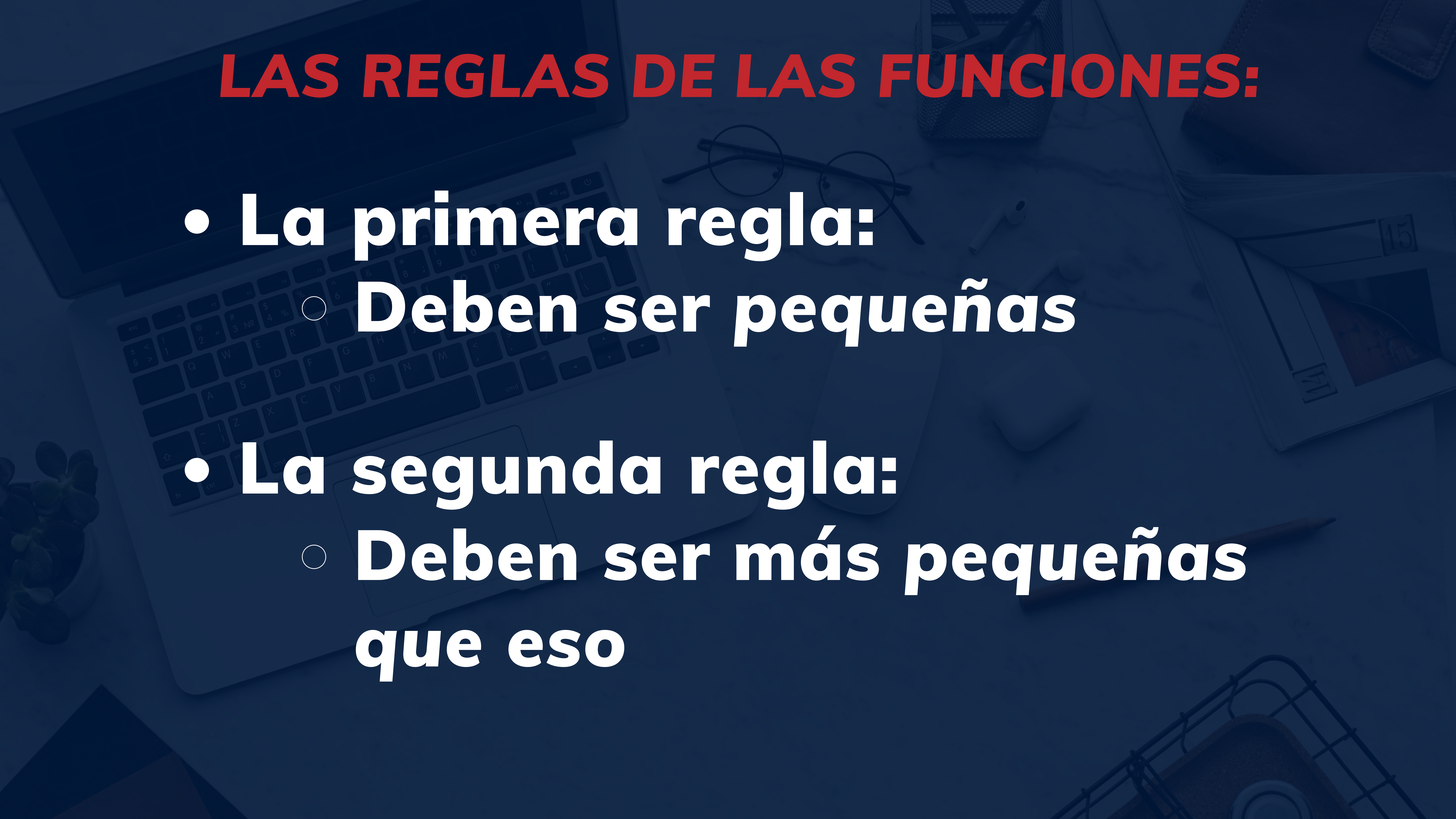


```java
public static String renderPageWithSetupsAndTeardowns(
    PageData pageData, boolean isSuite
  ) throws Exception {
    boolean isTestPage = pageData.hasAttribute("Test");
    if (isTestPage) {
      WikiPage testPage = pageData.getWikiPage();
      StringBuffer newPageContent = new StringBuffer();
      includeSetupPages(testPage, newPageContent, isSuite);
      newPageContent.append(pageData.getContent());
      includeTeardownPages(testPage, newPageContent, isSuite);
      pageData.setContent(newPageContent.toString());
    }

    return pageData.getHtml();
  }
```

IF SMALL IS GOOD, MAKE IT EVEN SMALLER



```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite) throws Exception {
  if (isTestPage(pageData))
    includeSetupAndTeardownPages(pageData, isSuite);
  return pageData.getHtml();
}
```



LAS REGLAS DE LAS FUNCIONES:

- La primera regla:
  - Deben ser *pequeñas*
- La segunda regla:
  - Deben ser más *pequeñas que eso*

PATRONES DE DISEÑO

TODAS LAS PROFESIONES TIENEN SU LINGO

BRASEAR

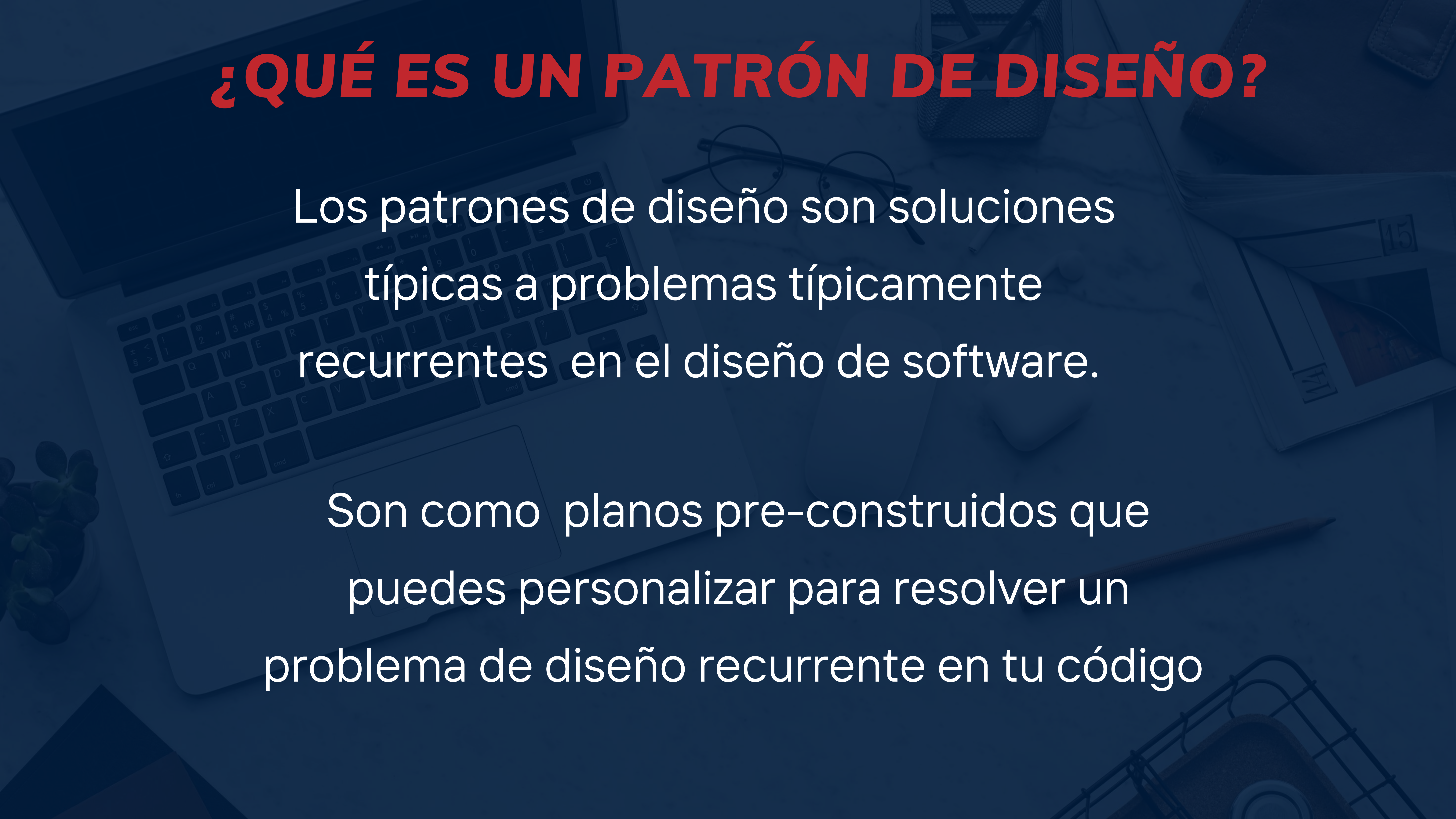
CARAMELIZAR

ESCALFAR

MARINAR

CORTE CHIFFONADE

AGARRE GARRA DE OSO

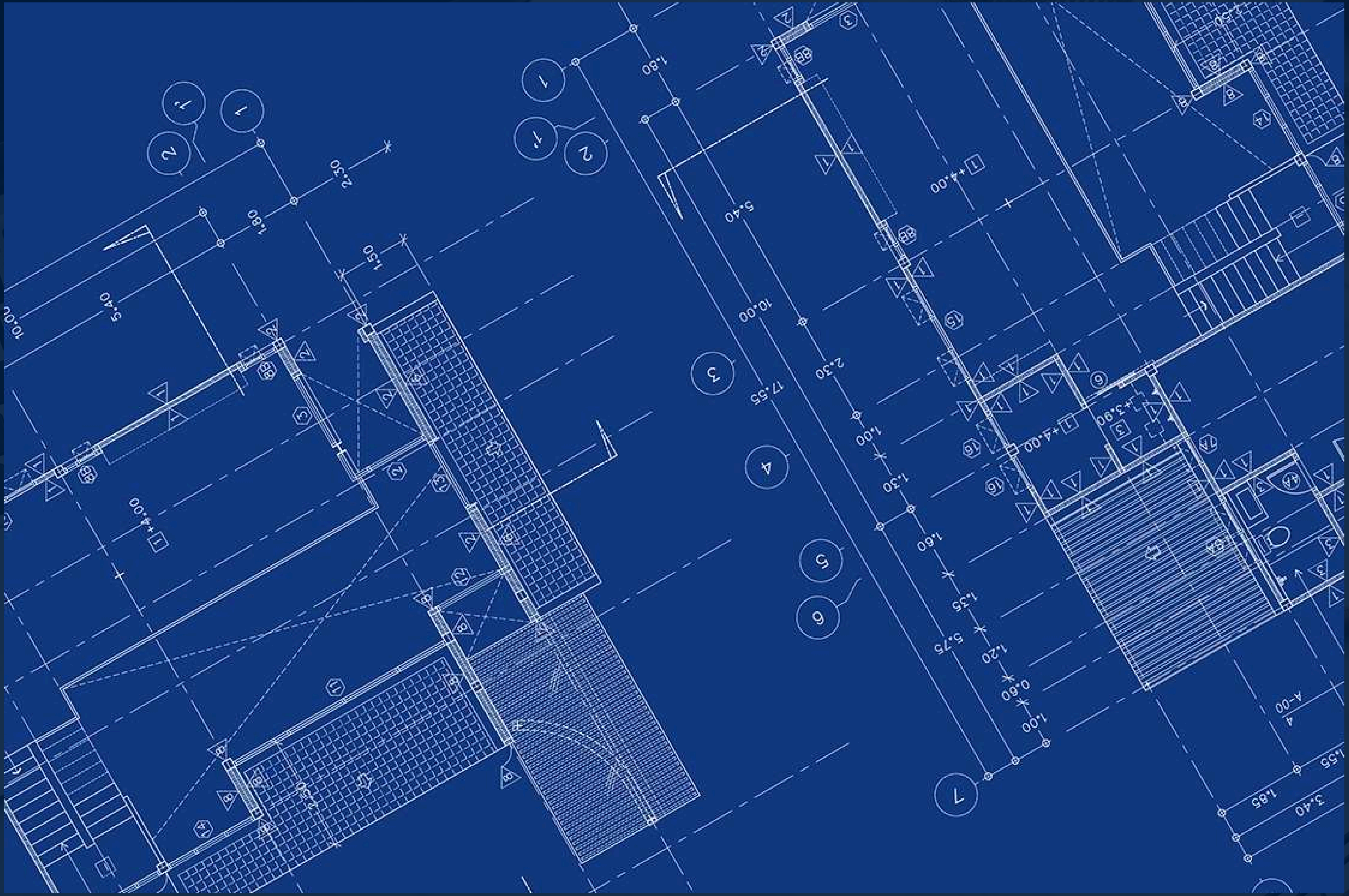


¿QUÉ ES UN PATRÓN DE DISEÑO?

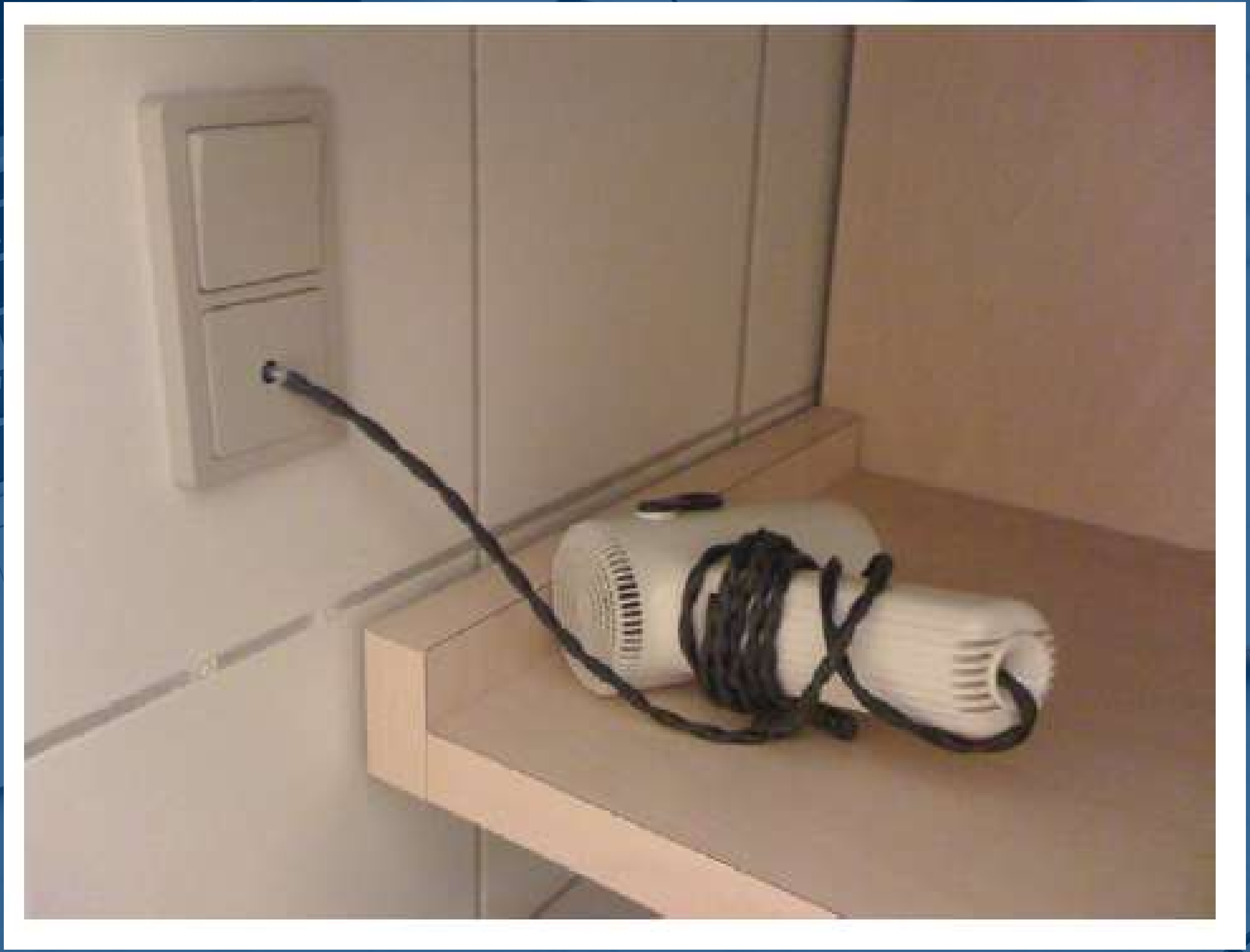
Los patrones de diseño son soluciones típicas a problemas típicamente recurrentes en el diseño de software.

Son como planos pre-construidos que puedes personalizar para resolver un problema de diseño recurrente en tu código

CÓMO UN PLANO

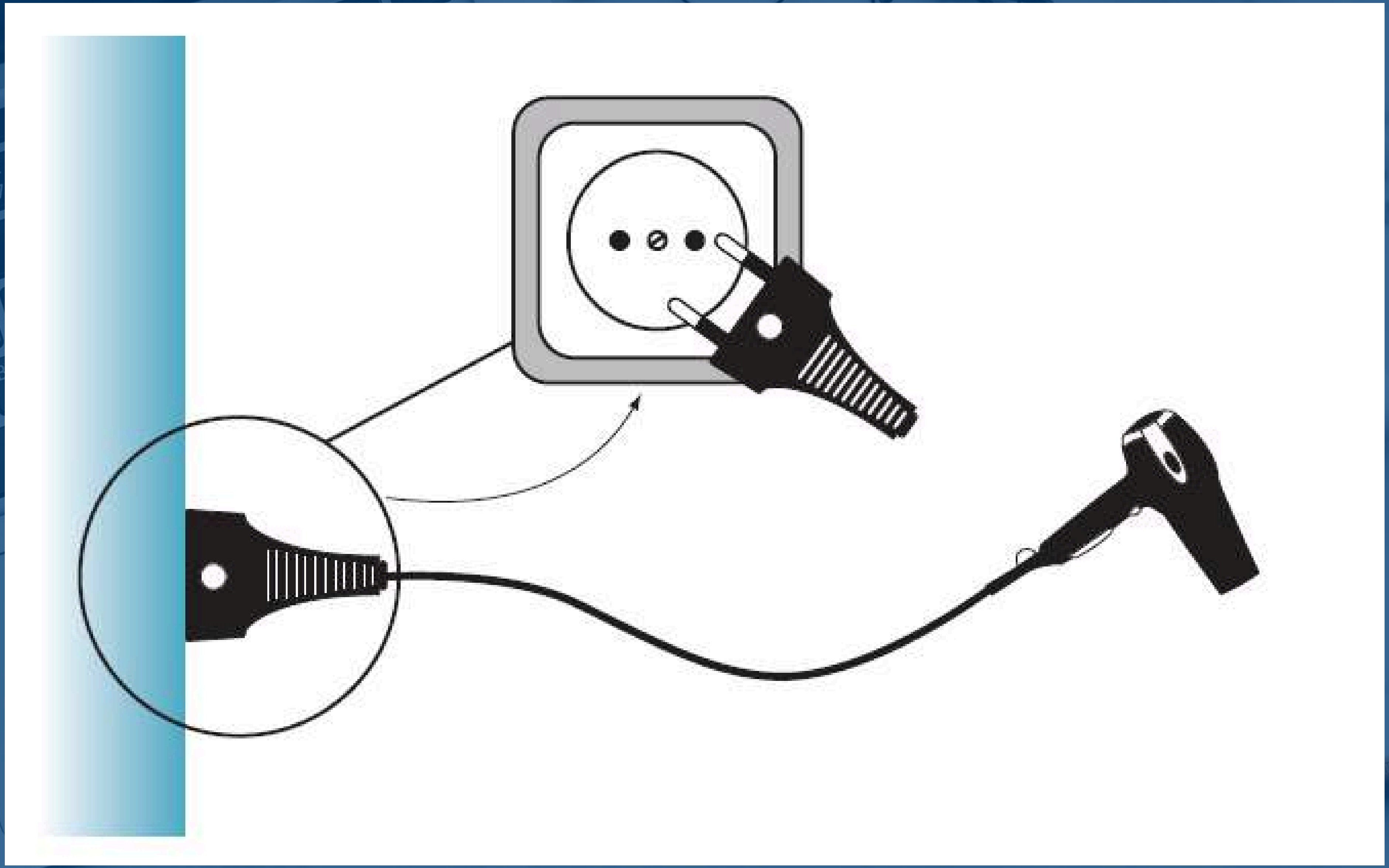


ANALICEMOS

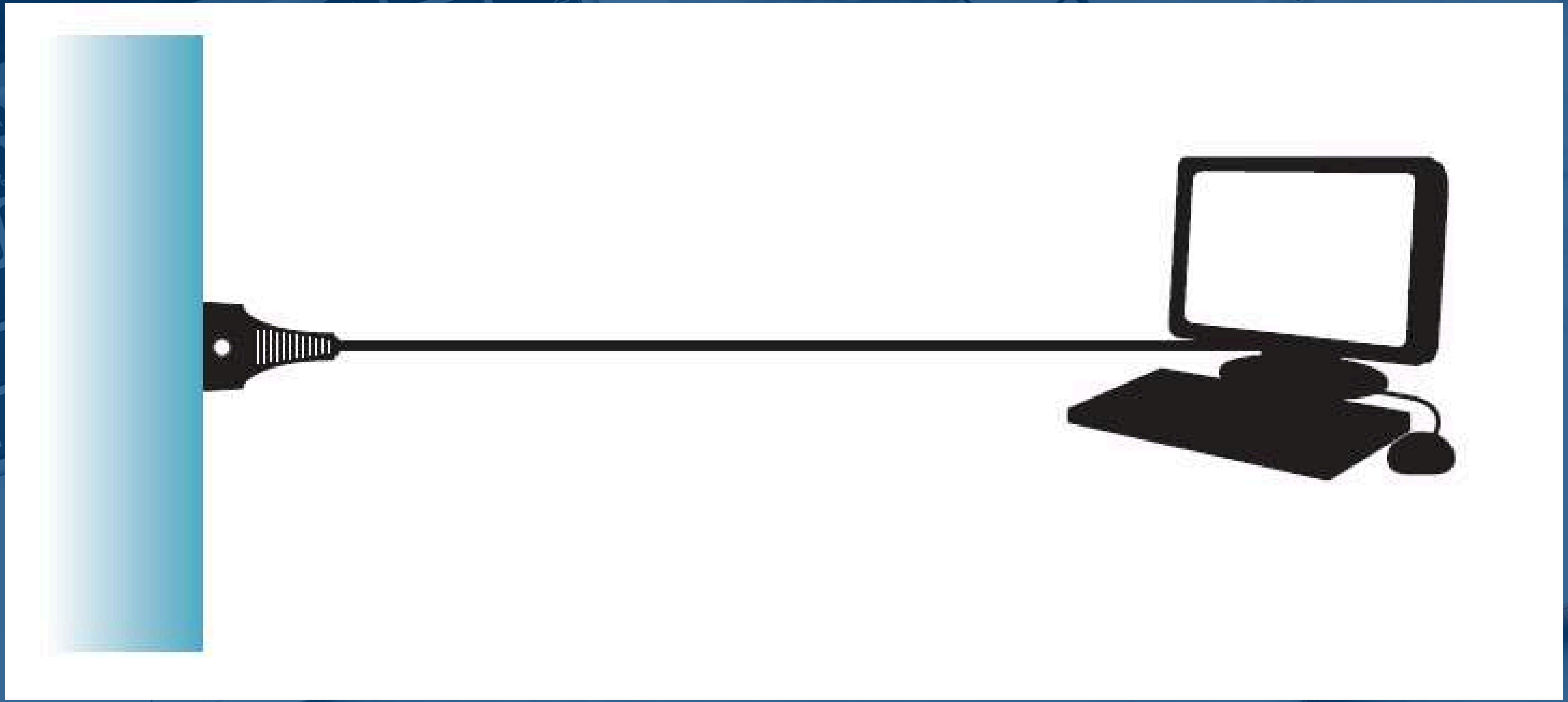


COMPARANDO CABLEADO ELECTRICO CON CÓDIGO Y DISEÑO

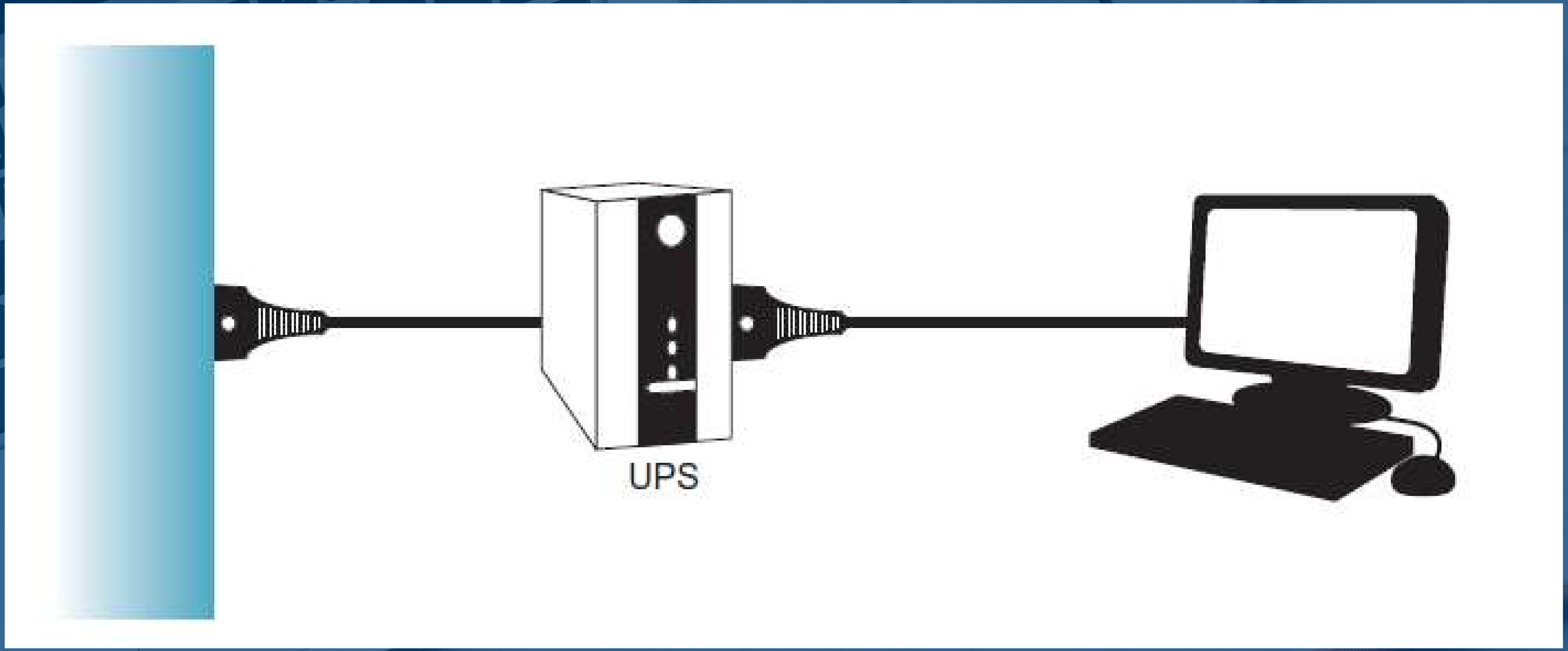
BAJO ACOPLAMIENTO



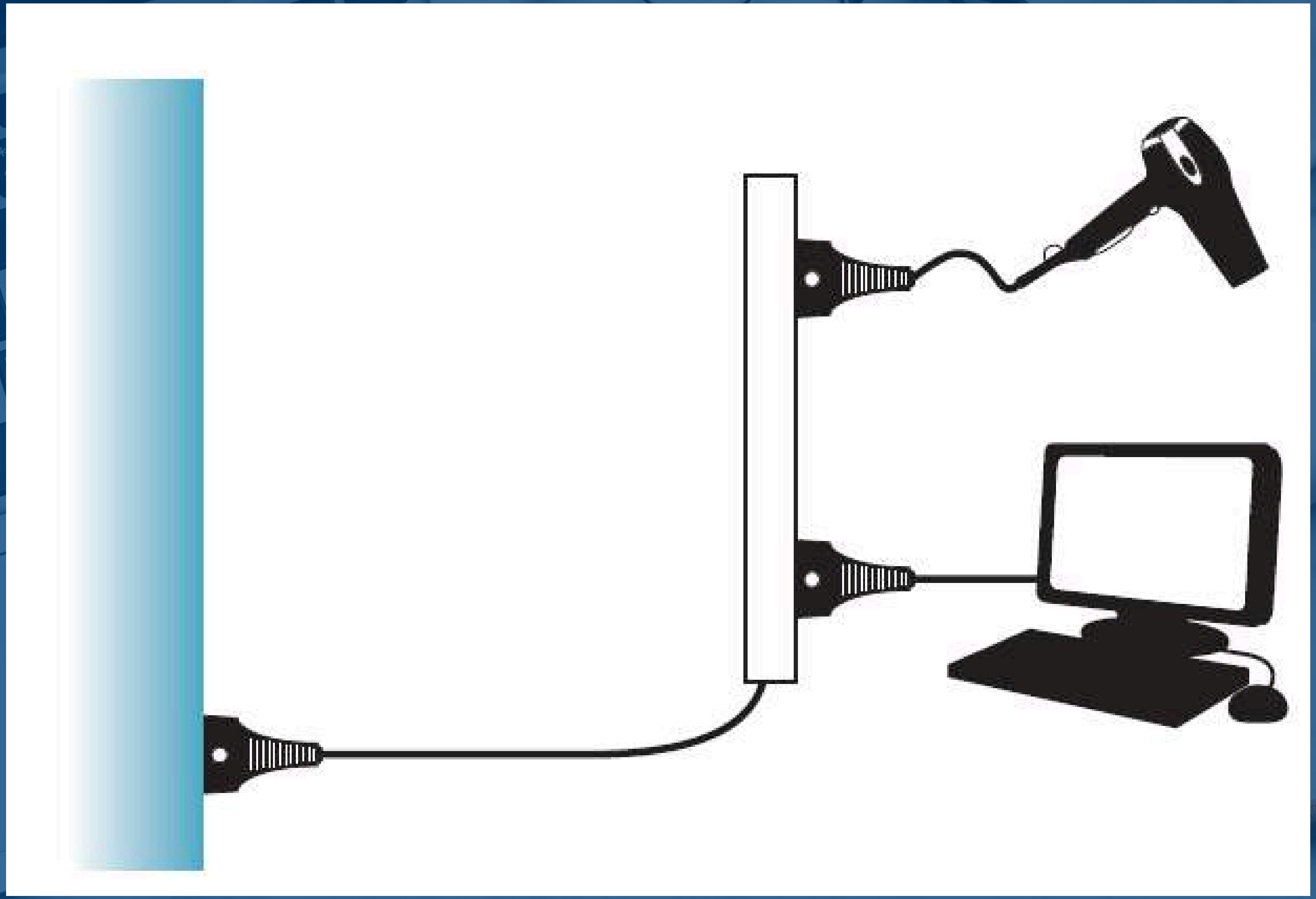
LISKOV SUBSTITUTION PRINCIPLE



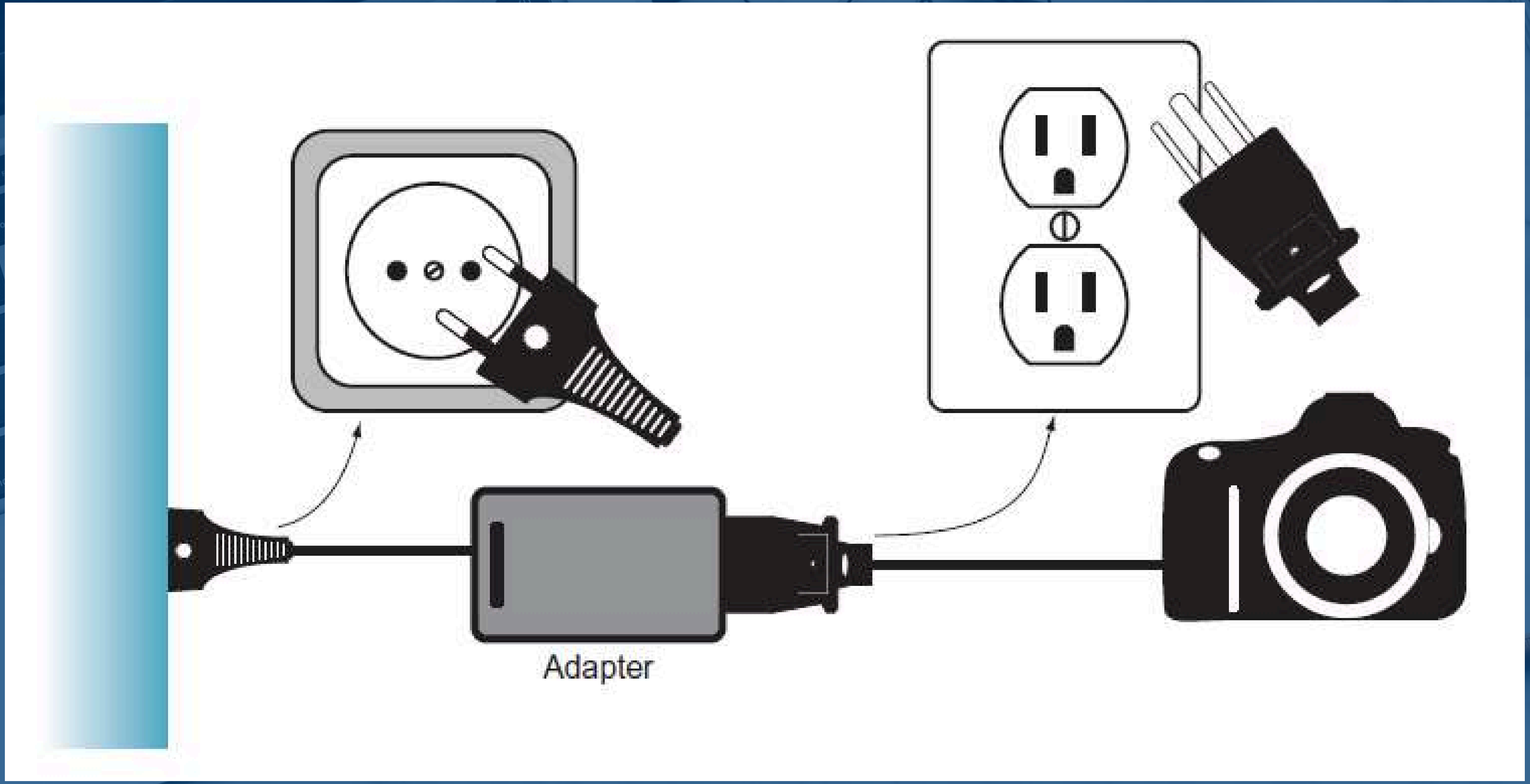
DECORATOR DESIGN PATTERN



COMPOSITE DESIGN PATTERN

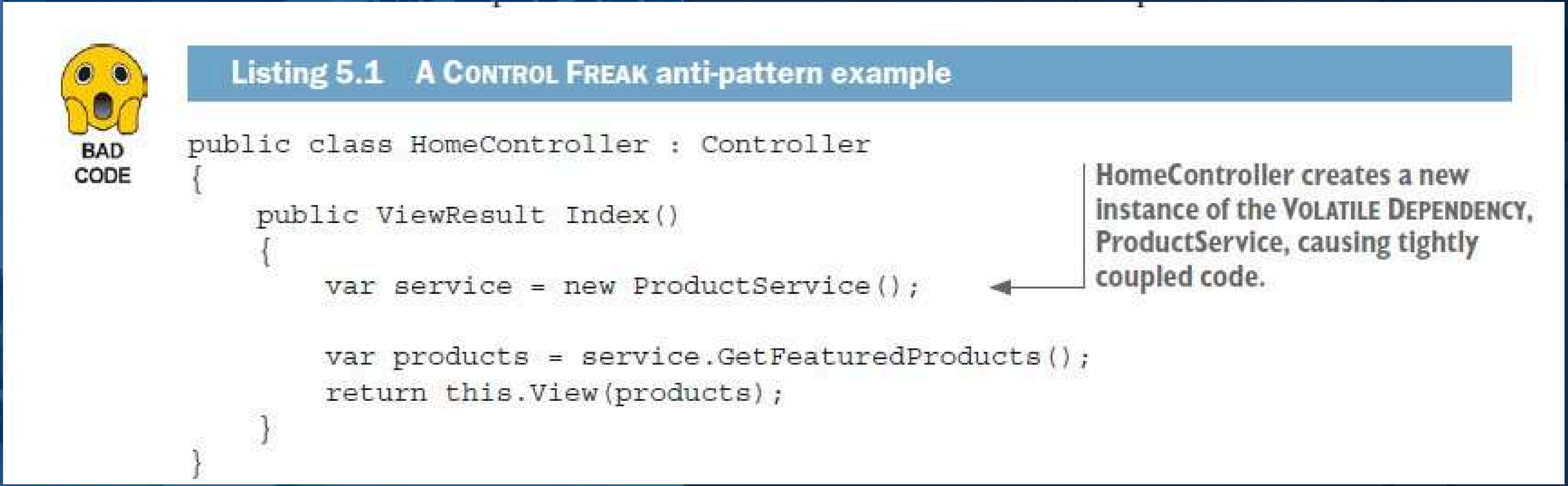


ADAPTER DESIGN PATTERN



ANTI PATRONES

CONTROL FREAK ANTI-PATTERN

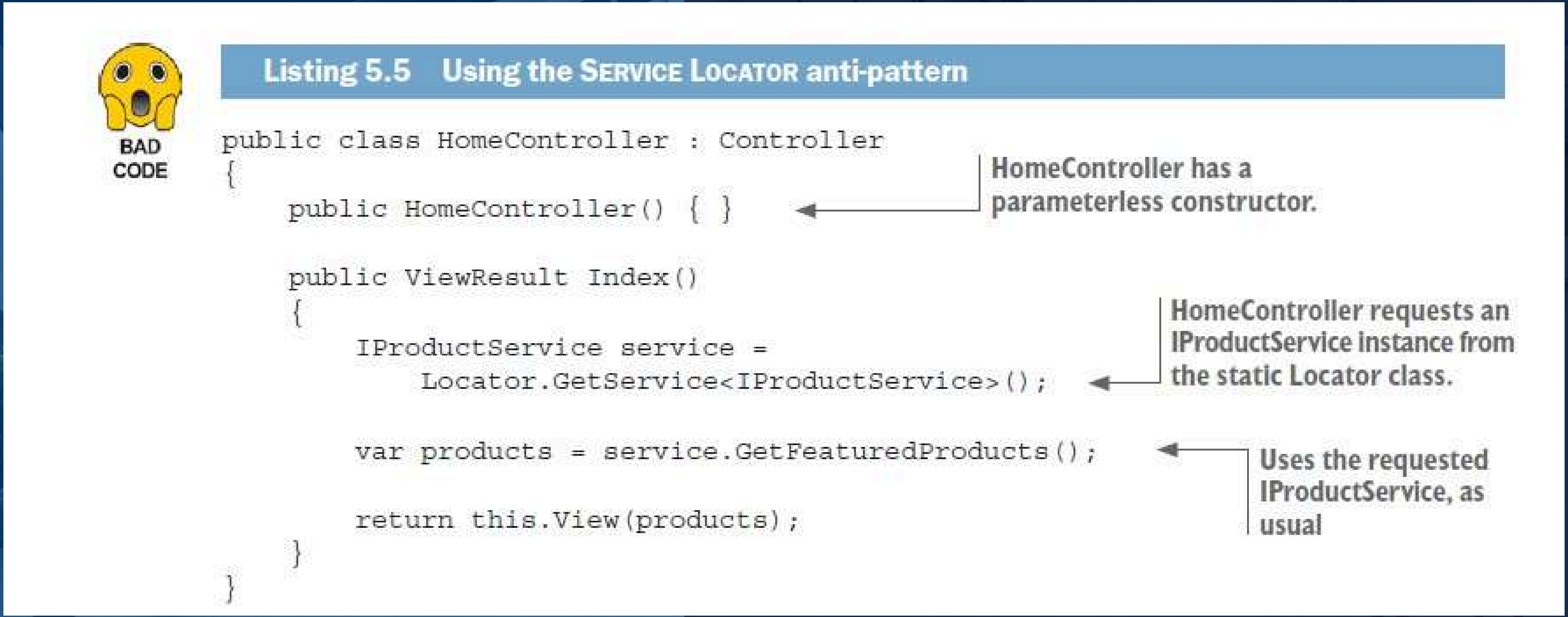


Listing 5.1 A CONTROL FREAK anti-pattern example

```
public class HomeController : Controller
{
    public ViewResult Index()
    {
        var service = new ProductService();

        var products = service.GetFeaturedProducts();
        return this.View(products);
    }
}
```

HomeController creates a new instance of the VOLATILE DEPENDENCY, ProductService, causing tightly coupled code.

SERVICE LOCATOR



BAD CODE

Listing 5.5 Using the SERVICE LOCATOR anti-pattern

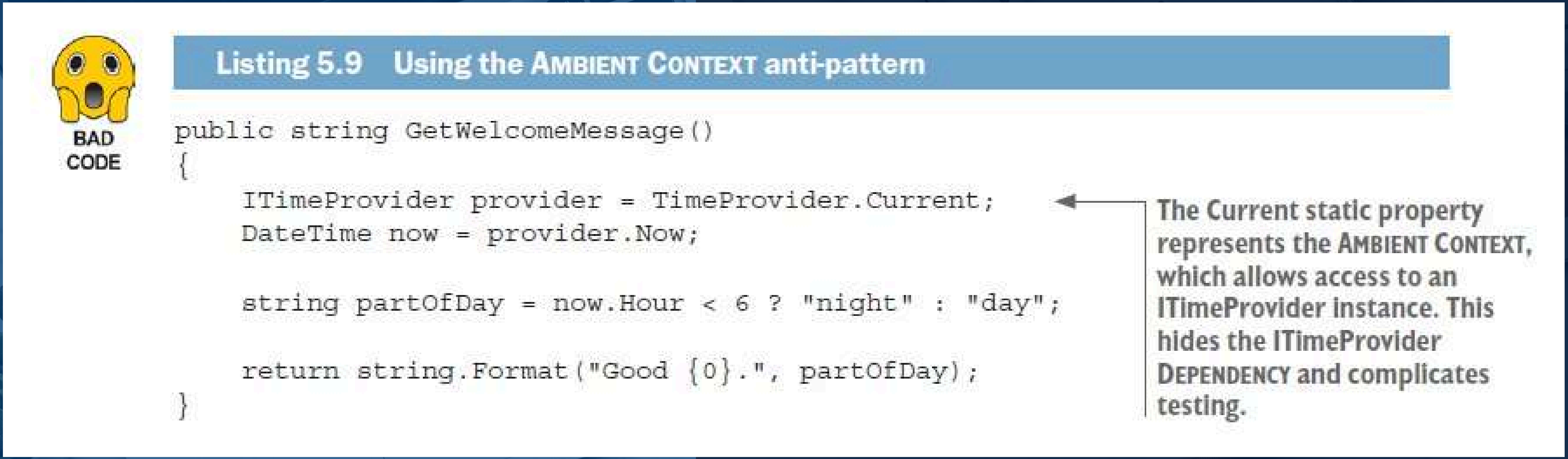
```
public class HomeController : Controller
{
    public HomeController() { }

    public ViewResult Index()
    {
        IProductService service =
            Locator.GetService<IProductService>();

        var products = service.GetFeaturedProducts();

        return this.View(products);
    }
}
```

HomeController has a parameterless constructor.

HomeController requests an IProductService instance from the static Locator class.

Uses the requested IProductService, as usual

AMBIENT CONTEXT



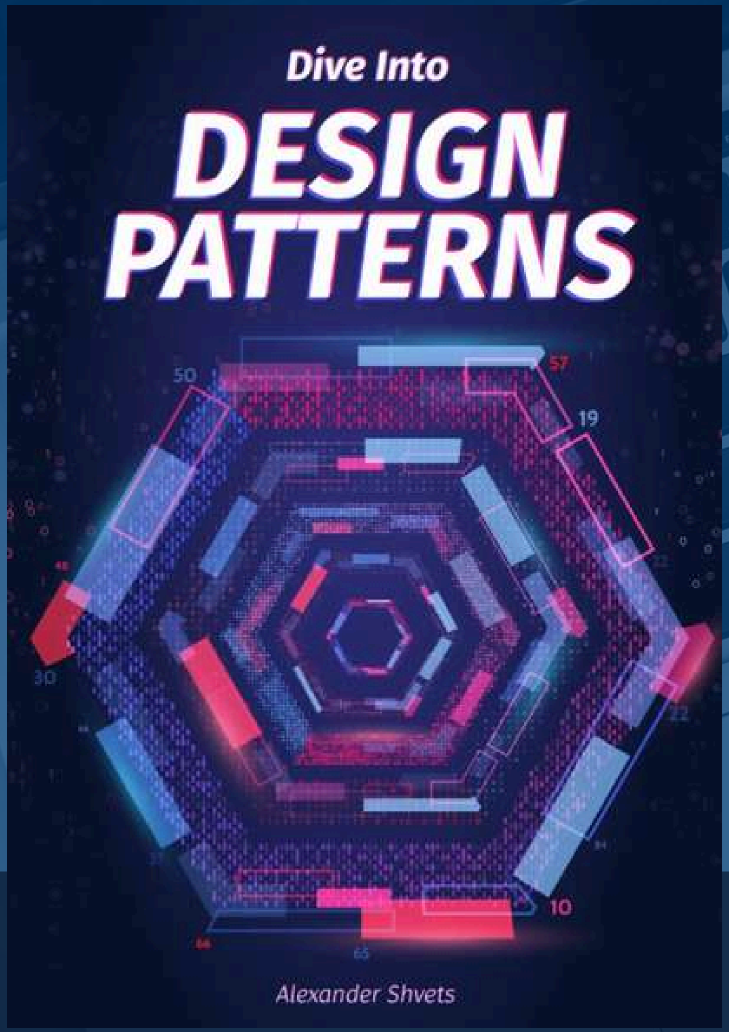
BAD CODE

Listing 5.9 Using the AMBIENT CONTEXT anti-pattern

```
public string GetWelcomeMessage()
{
    ITimeProvider provider = TimeProvider.Current;
    DateTime now = provider.Now;

    string partOfDay = now.Hour < 6 ? "night" : "day";

    return string.Format("Good {0}.", partOfDay);
}
```

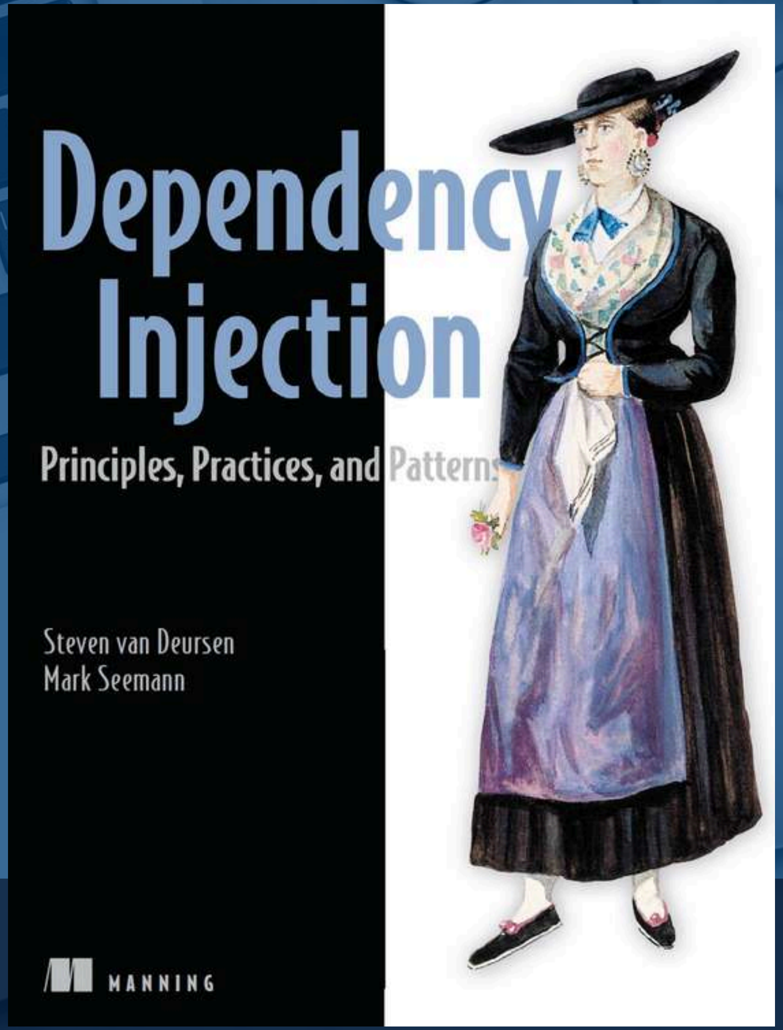
The Current static property represents the AMBIENT CONTEXT, which allows access to an ITimeProvider instance. This hides the ITimeProvider DEPENDENCY and complicates testing.

MATERIALES EXTRAS



DIVE INTO DESIGN PATTERNS

LIBRO



DEPENDENCY INJECTION: PRINCIPLES, PRACTICES AND PATTERNS

LIBRO

HTTPS://REFACTORING.GURU/DESIGN-PATTERNS

WEB

CONCLUSIONES

- Aprendamos sobre patrones de diseño, nos permitirá comunicarnos efectivamente con otros programadores.

- Clean Code es un tema excelente que aprender para ser mejores en escribir código

Marlon Peña

Software Engineer

¿PREGUNTAS?

Marlon E. Peña

Ingeniero de Software