netmind | a bts company

Desarrollo de Código Potenciado con SDD

Generación de Funciones, APIs y Patrones Multi-Lenguaje

Microsoft Copilot para Desarrolladores - Spec-Driven Development

3 horas 30 minutos

Prerrequisitos y Contexto

Artefactos SDD y diferencias con el Módulo 2

Artefactos SDD Disponibles

Artefacto	Ubicación	Propósito
`spec.md`	`specs/[feature]/`	Especificación funcional con requisitos
`plan.md`	`specs/[feature]/`	Plan técnico con decisiones de arquitectura
`templates/`	`.specify/templates/`	Templates reutilizables por tipo de código

Diferencia con Módulo 2

Módulo 2 (Workflow)	Módulo 3 (Código Específico)
¿Dónde guardar prompts?	¿Qué templates usar?
¿Cuándo invocar comandos?	¿Cómo escribir código avanzado?
Ciclo general SDD	Patrones específicos multi-lenguaje

Workflow de Generación de Código con Spec Kit

1. Seleccionar template desde `.specify/templates/[tipo]`
2. Personalizar prompt con contexto del dominio
3. Invocar agente: `@backend-dev generate`
4. Ejecutar: `/speckit.analyze`
5. Validar calidad antes de guardar

Dónde Guardar Templates de Código

Tipo de Código	Ubicación del Template
Clases C#	`.specify/templates/csharp/class-template.md`
APIs REST	`.specify/templates/csharp/api-endpoint-template.md`
Tests	`.specify/templates/csharp/test-template.md`
Python functions	`.specify/templates/python/function-template.md`
TypeScript components	`.specify/templates/typescript/component-template.md`

Sección 1: Generación de Funciones y Clases

Del/spec.md al código funcional en múltiples lenguajes

1.1 Introducción a la Generación de Código SDD

1.1 Introducción a la Generación de Código SDD

- Transformación sistemática: especificación → código

1.1 Introducción a la Generación de Código SDD

- Transformación sistemática: especificación → código
- Trazabilidad directa: cada método → criterio de aceptación

1.1 Introducción a la Generación de Código SDD

- Transformación sistemática: especificación → código
- Trazabilidad directa: cada método → criterio de aceptación
- Copilot como programador par con contexto completo

1.1 Introducción a la Generación de Código SDD

- Transformación sistemática: especificación → código
- Trazabilidad directa: cada método → criterio de aceptación
- Copilot como programador par con contexto completo
- Contraste con Vibe Coding: specs estructuradas vs descripciones ambiguas

1.1 Introducción a la Generación de Código SDD

- Transformación sistemática: especificación → código
- Trazabilidad directa: cada método → criterio de aceptación
- Copilot como programador par con contexto completo
- Contraste con Vibe Coding: specs estructuradas vs descripciones ambiguas

La generación de funciones y clases desde descripciones es el núcleo de Spec-Driven Development.

1.2 Técnicas de Especificación a Código

1.2 Técnicas de Especificación a Código

- Transformación de requisitos en implementaciones SDD

1.2 Técnicas de Especificación a Código

- Transformación de requisitos en implementaciones SDD
- Algoritmos de procesamiento de datos desde specs

1.2 Técnicas de Especificación a Código

- Transformación de requisitos en implementaciones SDD
- Algoritmos de procesamiento de datos desde specs
- Lógica de negocio compleja con patrón Specification

Prompt para Módulo de Autenticación JWT

```
Implementa el módulo de autenticación JWT según specs/001-auth-api/spec.md:

## Especificación de Referencia
- Archivo: specs/001-auth-api/spec.md
- Requisito: REQ-AUTH-001 (Autenticación JWT)
- Criterios de aceptación: CA-AUTH-001 a CA-AUTH-005

## Requisitos Funcionales
1. Usuario envía credenciales (email, password)
2. Sistema valida credenciales contra base de datos
3. Sistema genera JWT con claims personalizados
4. Sistema retorna token con expiración configurada
5. Usuario puede refresh token antes de expiración

## Convenciones del Proyecto
- Clean Architecture con Dependency Injection
- FluentValidation para DTOs
- Result pattern para respuestas
```

DTOs Generados

```
public class LoginRequestDto
{
    [Required(ErrorMessage = "Email es obligatorio")]
    [EmailAddress(ErrorMessage = "Formato de email inválido")]
    public string Email { get; set; } = string.Empty;

    [Required(ErrorMessage = "Password es obligatorio")]
    public string Password { get; set; } = string.Empty;
}

public class LoginResponseDto
{
    public string AccessToken { get; set; } = string.Empty;
    public string RefreshToken { get; set; } = string.Empty;
    public DateTime ExpiresAt { get; set; }
    public Guid UsuarioId { get; set; }
    public string Email { get; set; } = string.Empty;
    public IList<string> Roles { get; set; } = new List<string>();
}</string></string>
```

AuthService Implementation

```
public class AuthService : IAuthService
{
    private readonly IUsuarioRepository _usuarioRepository;
    private readonly ITokenService _tokenService;
    private readonly ILogger<authservice> _logger;

    public async Task<result<loginresponsedto>> LoginAsync(LoginRequestDto dto, CancellationToken ct)
    {
        var usuario = await _usuarioRepository.GetByEmailAsync(dto.Email, ct);
        if (usuario == null)
            return Result<loginresponsedto>.Failure("Credenciales inválidas");

        if (!VerifyPassword(dto.Password, usuario.PasswordHash))
            return Result<loginresponsedto>.Failure("Credenciales inválidas");

        if (!usuario.EstaActivo)
            return Result<loginresponsedto>.Failure("Usuario inactivo");

        var accessToken = _tokenService.GenerateAccessToken(usuario);
```

Prompt para Matriz de Trazabilidad

```
@agent Genera matriz de trazabilidad para specs/001-auth-api/spec.md:

## Criterios de Aceptación
| ID | Descripción |
|----|-------------|
| CA-AUTH-001 | Login con credenciales válidas retorna JWT |
| CA-AUTH-002 | Password incorrecto retorna 401 |
| CA-AUTH-003 | Token expira en tiempo configurado |
| CA-AUTH-004 | Refresh token renueva access token |
| CA-AUTH-005 | Usuario inactivo no puede autenticarse |

## Cobertura
- Total criterios: 5
- Implementados: 5 (100%)
- Testeados: 5 (100%)
```

Prompt para ProductividadCalculator

```
Implementa el motor de procesamiento de productividad según specs/002-productivity/spec.md:

## Algoritmos Requeridos
1. Tasa de completación: tareas completadas / tareas totales * 100
2. Tiempo promedio: suma(duración) / count(tareas completadas)
3. Productividad semanal: promedio(tasas semanales)
4. Predicción: regresión lineal de últimas 4 semanas

## Requisitos de Rendimiento
- Procesamiento de 10,000 registros en < 5 segundos
- Uso de vectorización NumPy
- Cacheo de resultados por usuario/día
```

ProductividadCalculator Class

```python
class ProductividadCalculator:
    def __init__(self, cache_enabled: bool = True):
        self._cache: dict[str, MetricaProductividad] = {}
        self._cache_enabled = cache_enabled

    def calcular_tasa_completacion(
        self, df: pd.DataFrame, usuario_id: Optional[str] = None
    ) -> float:
        if usuario_id:
            df = df[df['usuario_id'] == usuario_id]

        completadas = df[df['estado'] == 'completada']
        total = len(df)

        if total == 0:
            return 0.0

        tasa = (len(completadas) / total) * 100
        return round(tasa, 2)
```

Predicción con Regresión Lineal

```python
def predecir_productividad_proxima_semana(
    self, df_historico: pd.DataFrame, semanas_historico: int = 4
) -> dict:
    df_recent = df_historico.tail(semanas_historico).copy()

    if len(df_recent) < 2:
        return {'prediccion': None, 'mensaje': 'Datos insuficientes'}

    x = np.arange(len(df_recent))
    y = df_recent['productividad_score'].values

    coefficients = np.polyfit(x, y, 1)
    slope, intercept = coefficients

    prediccion = slope * len(df_recent) + intercept
    prediccion = max(0, min(100, prediccion))

    return {
        'prediccion_proxima_semana': round(prediccion, 2)
```

Prompt para Motor de Reglas de Negocio

```
Implementa el motor de validación de reglas de negocio para PortalEmpleo:

## Reglas de Negocio
1. Usuario solo puede tener 10 tareas activas simultáneamente
2. Tarea con prioridad 'urgente' debe completarse en 24h
3. Usuario con rol 'junior' requiere aprobación para más de 5 tareas completadas/semana
4. Tarea asignada a otro usuario no puede modificarse
5. Manager puede reasignar tareas de su equipo

## Patrón Specification
public interface ISpecification<t> { bool IsSatisfiedBy(T entity); }
public interface IBusinessRule { bool IsBroken(); string Message; Severity Severity; }</t>
```

Business Rules Implementadas

```
public class MaxActiveTasksRule : BusinessRule
{
    public override bool IsBroken()
    {
        var activeTasks = _context.Tareas
            .Count(t => t.CreadoPorId == _usuarioId &&
                        t.Estado != TareaStatus.Completada &&
                        t.Estado != TareaStatus.Cancelada);
        return activeTasks >= 10;
    }

    public override string Message =>
        "Usuario ha alcanzado el límite de 10 tareas activas";

    public override Severity Severity => Severity.Warning;
}

public class UrgentTaskDeadlineRule : BusinessRule
{
```

Business Rule Engine

```
public class BusinessRuleEngine
{
    private readonly List<ibusinessrule> _rules;

    public RuleValidationResult Validate()
    {
        var brokenRules = _rules.Where(r => r.IsBroken()).ToList();

        return new RuleValidationResult
        {
            IsValid = !brokenRules.Any(),
            BrokenRules = brokenRules,
            Errors = brokenRules
                .Where(r => r.Severity == Severity.Error)
                .Select(r => r.Message)
                .ToList(),
            Warnings = brokenRules
                .Where(r => r.Severity == Severity.Warning)
                .Select(r => r.Message)
```

1.3 Ejemplos Multi-Lenguaje

1.3 Ejemplos Multi-Lenguaje

- C#: Web APIs con validación y error handling SDD

1.3 Ejemplos Multi-Lenguaje

- C#: Web APIs con validación y error handling SDD
- Python: Data Processing y ML Pipelines SDD

1.3 Ejemplos Multi-Lenguaje

- C#: Web APIs con validación y error handling SDD
- Python: Data Processing y ML Pipelines SDD
- JavaScript/TypeScript: React Components con State Management

1.3 Ejemplos Multi-Lenguaje

- C#: Web APIs con validación y error handling SDD
- Python: Data Processing y ML Pipelines SDD
- JavaScript/TypeScript: React Components con State Management
- Coordinación Polyglot con Spec Kit

Prompt para API C#

```
Implementa el controller de autenticación siguiendo AGENTS.md:

## Convenciones AGENTS.md
- Controller con atributos de documentación
- FluentValidation para DTOs
- ProblemDetails para errores
- CancellationToken en todos los métodos
```

AuthController con Convenciones AGENTS.md

```
[ApiController]
[Route("api/[controller]")]
[Produces("application/json")]
public class AuthController : ControllerBase
{
    private readonly IAuthService _authService;
    private readonly IValidator<loginrequestdto> _loginValidator;

    [HttpPost("login")]
    [ProducesResponseType(typeof(LoginResponseDto), StatusCodes.Status200OK)]
    [ProducesResponseType(typeof(ProblemDetails), StatusCodes.Status401Unauthorized)]
    [ProducesResponseType(typeof(ValidationProblemDetails), StatusCodes.Status400BadRequest)]
    public async Task<iactionresult> Login(
        [FromBody] LoginRequestDto request,
        CancellationToken cancellationToken)
    {
        var validationResult = await _loginValidator.ValidateAsync(request, cancellationToken);
        if (!validationResult.IsValid)
        {
```

FluentValidation

```
public class LoginRequestDtoValidator : AbstractValidator<loginrequestdto>
{
    public LoginRequestDtoValidator()
    {
        RuleFor(x => x.Email)
            .NotEmpty().WithMessage("Email es obligatorio")
            .EmailAddress().WithMessage("Formato de email inválido");

        RuleFor(x => x.Password)
            .NotEmpty().WithMessage("Password es obligatorio")
            .MinimumLength(8).WithMessage("Password mínimo 8 caracteres");
    }
}</loginrequestdto>
```

Prompt para Pipeline ML

```
Implementa pipeline de ML para predicción de productividad:

## Pipeline Stages
1. Data Ingestion: Leer datos de SQL Server
2. Data Cleaning: Missing values, outliers
3. Feature Engineering: Crear features
4. Model Training: Regresión lineal + Random Forest
5. Model Evaluation: Cross-validation
6. Model Serving: API para predicciones
```

MLPipeline Class

```python
class MLPipeline:
    def run(self, mode: str = 'train') -> Dict:
        try:
            self._log("Stage 1: Ingestión de datos")
            raw_data = self.ingestor.extract(query=self.config['data_query'], mode=mode)

            self._log("Stage 2: Limpieza de datos")
            clean_data = self.cleaner.process(raw_data)

            self._log("Stage 3: Feature engineering")
            if mode == 'train':
                featured_data, feature_names = self.engineer.create_features(clean_data)

                self._log("Stage 4: Entrenamiento")
                X, y = self._split_features_target(featured_data)
                model, metrics = self.trainer.train(X, y)

                self._log("Stage 5: Evaluación")
                eval_metrics = self.evaluator.evaluate(model, X, y)
```

ProductividadPredictor para Serving

```
class ProductividadPredictor:
    def predict(self, features: Dict) -> Dict:
        import pandas as pd

        df = pd.DataFrame([features])
        df = df[self.feature_names]

        productividad = self.model.predict(df)[0]
        confidence = self._calculate_confidence(df)

        return {
            'productividad_predicha': round(productividad, 2),
            'confidence': confidence,
            'timestamp': datetime.utcnow().isoformat()
        }
```

Prompt para React Components

```
Implementa pantalla de lista de tareas con React y Zustand:

## Stack Tecnológico
- React 18 con TypeScript
- Zustand para state management
- TanStack Query para data fetching
- React Navigation 6
```

TypeScript Types y Hooks

```typescript
export interface Tarea {
  id: string;
  titulo: string;
  descripcion: string | null;
  estado: 'pendiente' | 'en_progreso' | 'completada' | 'cancelada';
  prioridad: 'baja' | 'media' | 'alta' | 'urgente';
  fechaLimite: string | null;
  creadoEn: string;
}

export const useTasks = (filters: TareaFilters) => {
  return useQuery({
    queryKey: ['tasks', filters],
    queryFn: async () => {
      const params = new URLSearchParams({
        page: filters.page.toString(),
        pageSize: filters.pageSize.toString(),
      });
```

Zustand Store

```
interface TaskState {
  selectedTask: Tarea | null;
  filters: TareaFilters;
  sortBy: 'creadoEn' | 'fechaLimite' | 'prioridad';
  sortOrder: 'asc' | 'desc';

  setSelectedTask: (task: Tarea | null) => void;
  setFilters: (filters: Partial<tareafilters>) => void;
  setSortBy: (sortBy: 'creadoEn' | 'fechaLimite' | 'prioridad') => void;
  setSortOrder: (order: 'asc' | 'desc') => void;
  resetFilters: () => void;
}

export const useTaskStore = create<taskstate>((set) => ({
  selectedTask: null,
  filters: { page: 1, pageSize: 20 },
  sortBy: 'creadoEn',
  sortOrder: 'desc',
```

TaskListScreen Component

```
export const TaskListScreen: React.FC = () => {
  const [showFilters, setShowFilters] = useState(false);
  const [refreshing, setRefreshing] = useState(false);

  const { filters, setFilters, resetFilters } = useTaskStore();

  const { data, isLoading, isFetching, refetch } = useTasks(filters);
  const completeTask = useCompleteTask();
  const deleteTask = useDeleteTask();

  const handleRefresh = async () => {
    setRefreshing(true);
    await refetch();
    setRefreshing(false);
  };

  return (
    <view style="{styles.container}">
      <flatlist data="{data?.items}" renderitem="{({" item="" })=""> (
```

Arquitectura Polyglot

```
portalempleo/
├── backend/           # C# ASP.NET Core 8
│   ├── src/
│   │   ├── API/
│   │   ├── Application/
│   │   ├── Domain/
│   │   └── Infrastructure/
│   └── tests/
├── ml-pipeline/       # Python 3.12
│   ├── src/
│   │   ├── data/
│   │   ├── models/
│   │   └── api/
│   └── notebooks/
├── frontend/          # TypeScript React 18
│   ├── src/
│   │   ├── components/
│   │   ├── screens/
│   │   ├── hooks/
```

AGENTS.md Polyglot Configuration

```
# PortalEmpleo - AGENTS.md (Polyglot)

## Multi-Language Coordination

### Backend (C#)
- Location: backend/
- Agent: @backend-dev
- Test Framework: xUnit + Moq

### ML Pipeline (Python)
- Location: ml-pipeline/
- Agent: @ml-dev
- Test Framework: pytest

### Frontend (TypeScript)
- Location: frontend/
- Agent: @frontend-dev

## Shared Contracts
```

1.4 Patrones de Codificación Limpia SDD

1.4 Patrones de Codificación Limpia SDD

- SOLID Principles en decisiones SDD

1.4 Patrones de Codificación Limpia SDD

- SOLID Principles en decisiones SDD
- Clean Architecture Patterns para estructuración

Prompt para Implementar SOLID

```
Refactoriza AuthService aplicando principios SOLID:

## Problema Actual (viola SRP, ISP, DIP)
public class AuthService : IAuthService
{
    private readonly PortalEmpleoDbContext _context;
    private readonly JwtSecurityTokenHandler _tokenHandler;
    private readonly PasswordHasher _passwordHasher;
    private readonly EmailService _emailService;
    private readonly CacheService _cache;

    public async Task<loginresult> Login(string email, string password)
    {
        // Login logic, Token generation, Password verification, Email sending, Caching
    }
}</loginresult>
```

Aplicación de SRP e ISP

```
public interface IPasswordService
{
    string HashPassword(string password);
    bool VerifyPassword(string password, string hash);
}

public interface ITokenService
{
    string GenerateAccessToken(Usuario usuario);
    Task<refreshtoken> GenerateRefreshTokenAsync(Guid usuarioId, CancellationToken ct);
}

public class AuthService : IAuthService
{
    private readonly IUsuarioRepository _usuarioRepository;
    private readonly IPasswordService _passwordService;
    private readonly ITokenService _tokenService;

    public async Task<result<loginresponsedto>> LoginAsync(LoginRequestDto dto, CancellationToken ct)
```

DIP y Testability

```
public class AuthServiceTests
{
    private readonly Mock<iusuariorepository> _mockRepo;
    private readonly Mock<ipasswordservice> _mockPassword;
    private readonly Mock<itokenservice> _mockToken;

    [Fact]
    public void Login_ValidCredentials_ReturnsSuccess()
    {
        var service = new AuthService(
            _mockRepo.Object,
            _mockPassword.Object,
            _mockToken.Object,
            Mock.Of<ilogger<authservice>>());

        result.IsSuccess.Should().BeTrue();
    }
}</ilogger<authservice></itokenservice></ipasswordservice></iusuariorepository>
```

Estructura Clean Architecture

```
src/
├── Domain/           # Enterprise business rules
├── Application/      # Application business rules
├── Infrastructure/   # Frameworks and drivers
└── API/              # Interface adapters
```

Domain Layer

```
namespace PortalEmpleo.Domain.Common;

public abstract class Entity
{
    public Guid Id { get; protected set; }
    public DateTime CreatedAt { get; protected set; }
    public DateTime? UpdatedAt { get; protected set; }
}

public abstract class ValueObject
{
    protected abstract IEnumerable<object> GetEqualityComponents();
    public override bool Equals(object? obj) => /* ... */;
}
```

Application Layer (MediatR)

```
public record CreateTareaCommand : IRequest<tarearesponsedto>
{
    public string Titulo { get; init; } = string.Empty;
    public string? Descripcion { get; init; }
    public DateTime? FechaLimite { get; init; }
    public string Prioridad { get; init; } = "media";
}

public class CreateTareaCommandHandler
    : IRequestHandler<createtareacommand, result<tarearesponsedto="">>
{
    private readonly ITareaFactory _factory;
    private readonly ITareaRepository _repository;
    private readonly IMapper _mapper;

    public async Task<result<tarearesponsedto>> Handle(CreateTareaCommand request, CancellationToken ct)
    {
        var tarea = _factory.CreateFromDto(request, GetCurrentUserId());
        await _repository.AddAsync(tarea, ct);
```

1.5 Ejemplo Práctico: Result Pattern Template

Caso: PortalEmpleo implementa el patrón Result para gestión de errores en autenticación.

Dónde se guarda:

- Template: `.specify/templates/csharp/result-pattern-template.md`
- Prompt: `specs/001-auth-api/code-prompts.md`
- Código: `src/Application/Common/Results/`

Result Pattern Template

```
public class Result<t>
{
    public bool IsSuccess { get; }
    public T? Value { get; }
    public string? Error { get; private set; }

    private Result(bool isSuccess, T? value, string? error)
    {
        IsSuccess = isSuccess;
        Value = value;
        Error = error;
    }

    public static Result<t> Success(T value) => new(true, value, null);
    public static Result<t> Failure(string error) => new(false, default, error);

    public TResult Match<tresult>(
        Func<t, tresult=""> onSuccess,
        Func<string, tresult=""> onFailure)
```

Integración con Spec Kit

```
# 1. Seleccionar template
specs/001-auth-api/code-prompts.md → Template: result-pattern-template.md

# 2. Personalizar para el dominio
# Cambiar T por el tipo específico (Usuario, Token, etc.)

# 3. Generar código
@backend-dev Genera Result<t> para autenticación según template result-pattern-template.md

# 4. Validar calidad
/speckit.analyze</t>
```

Sección 2: APIs y Microservicios con Copilot

Desarrollo de endpoints robustos y documentación automática

2.1 Desarrollo de Endpoints REST SDD

2.1 Desarrollo de Endpoints REST SDD

- CRUD Operations con validación robusta

2.1 Desarrollo de Endpoints REST SDD

- CRUD Operations con validación robusta
- Error handling y response formatting

2.1 Desarrollo de Endpoints REST SDD

- CRUD Operations con validación robusta
- Error handling y response formatting
- API Versioning strategies

2.1 Desarrollo de Endpoints REST SDD

- CRUD Operations con validación robusta
- Error handling y response formatting
- API Versioning strategies
- Contract-First development

Prompt para CRUD Completo

```
Implementa CRUD completo de tareas con validación SDD:

## Endpoints Requeridos
GET    /api/v1/tareas           # Lista paginada
GET    /api/v1/tareas/{id}      # Por ID
POST   /api/v1/tareas           # Crear
PUT    /api/v1/tareas/{id}      # Actualizar
DELETE /api/v1/tareas/{id}      # Eliminar (soft delete)

## Validación de Negocio
1. Título: requerido, 1-200 caracteres
2. Prioridad: enum válido
3. Fecha límite: opcional, futura o presente
4. Usuario solo modifica sus propias tareas
```

CreateTareaDto con FluentValidation

```
public class CreateTareaDto
{
    public string Titulo { get; set; } = string.Empty;
    public string? Descripcion { get; set; }
    public DateTime? FechaLimite { get; set; }
    public string Prioridad { get; set; } = "media";
    public Guid? AsignadoAId { get; set; }
}

public class CreateTareaDtoValidator : AbstractValidator<createtareadto>
{
    public CreateTareaDtoValidator()
    {
        RuleFor(x => x.Titulo)
            .NotEmpty().WithMessage("Título es obligatorio")
            .Length(1, 200).WithMessage("Título debe tener entre 1 y 200 caracteres");

        RuleFor(x => x.Prioridad)
            .Must(BeValidPriority).WithMessage("Prioridad inválida");
```

TareasController con MediatR

```
[ApiController]
[Route("api/v1/tareas")]
[Produces("application/json")]
public class TareasController : ControllerBase
{
    private readonly IMediator _mediator;

    [HttpGet]
    [ProducesResponseType(typeof(PagedResult<tarearesponsedto>), StatusCodes.Status200OK)]
    public async Task<iactionresult> GetAll([FromQuery] GetTareasQuery query)
    {
        var result = await _mediator.Send(query);
        return Ok(result);
    }

    [HttpGet("{id:guid}")]
    [ProducesResponseType(typeof(TareaResponseDto), StatusCodes.Status200OK)]
    public async Task<iactionresult> GetById(Guid id)
    {
```

Prompt para Error Handling

```
Implementa sistema de error handling consistente:

## Requisitos
- ProblemDetails para todas las APIs
- Correlation ID para trazabilidad
- No exponer información sensible
- Logging estructurado
```

GlobalExceptionHandler

```
public class GlobalExceptionHandler : IExceptionHandler
{
    private readonly ILogger<globalexceptionhandler> _logger;

    public async ValueTask HandleAsync(Exception exception, HttpContext context, CancellationToken ct)
    {
        var correlationId = context.TraceIdentifier;
        _logger.LogError(exception, "Error processing request {CorrelationId}", correlationId);

        var (statusCode, problem) = exception switch
        {
            ValidationException ve => CreateValidationProblem(ve, correlationId),
            NotFoundException ne => CreateNotFoundProblem(ne, correlationId),
            BusinessRuleException bre => CreateBusinessRuleProblem(bre, correlationId),
            _ => CreateInternalServerProblem(exception, correlationId)
        };

        context.Response.StatusCode = statusCode;
        context.Response.ContentType = "application/problem+json";
```

Custom Exceptions

```
public abstract class AppException : Exception
{
    public string ErrorCode { get; }
    public HttpStatusCode StatusCode { get; }

    protected AppException(string message, string errorCode, HttpStatusCode statusCode)
        : base(message)
    {
        ErrorCode = errorCode;
        StatusCode = statusCode;
    }
}

public class NotFoundException : AppException
{
    public NotFoundException(string message)
        : base(message, "NOT_FOUND", HttpStatusCode.NotFound) { }
}
```

Prompt para API Versioning

```
Implementa API versioning para PortalEmpleo:

## Estrategia
- URL versioning: /api/v1/tareas
- Default version: v1
- Migration path: v1 → v2

## Versiones
- v1: Implementación inicial
- v2: Breaking changes (nuevos campos requeridos)
```

API Versioning Configuration

```
builder.Services.AddApiVersioning(options =>
{
    options.DefaultApiVersion = new ApiVersion(1, 0);
    options.AssumeDefaultVersionWhenUnspecified = true;
    options.ReportApiVersions = true;
    options.ApiVersionReader = ApiVersionReader.Combine(
        new UrlSegmentApiVersionReader(),
        new HeaderApiVersionReader("X-API-Version"));
});

[ApiController]
[Route("api/v{version:apiVersion}/[controller]")]
[ApiVersion("1.0")]
[ApiVersion("2.0")]
public class TareasController : ControllerBase
{
    [HttpGet]
    [MapToApiVersion("1.0")]
    public IActionResult GetV1() => Ok(_service.GetAll());
```

Prompt para Contract-First

```
Implementa contract-first development con OpenAPI:

## Workflow
1. Definir OpenAPI spec
2. Generar DTOs desde spec
3. Implementar controller
4. Validar implementación contra spec
```

OpenAPI Spec

```yaml
openapi: 3.0.3
info:
  title: PortalEmpleo API
  version: 1.0.0
paths:
  /api/v1/tareas:
    get:
      summary: Lista de tareas
      parameters:
        - name: page
          in: query
          schema:
            type: integer
            default: 1
      responses:
        '200':
          description: Lista de tareas
          content:
            application/json:
```

2.2 Middlewares y Validaciones SDD

2.2 Middlewares y Validaciones SDD

- Authentication/Authorization Middleware JWT

2.2 Middlewares y Validaciones SDD

- Authentication/Authorization Middleware JWT
- Rate Limiting Middleware

2.2 Middlewares y Validaciones SDD

- Authentication/Authorization Middleware JWT
- Rate Limiting Middleware
- Request/Response Logging estructurado

Prompt para Middleware de Autenticación

```
Implementa JWT authentication middleware:

## Requisitos
- Validar JWT token en cada request
- Extraer claims del token
- Verificar roles y permisos
- Rate limiting por usuario
```

JWT Authentication Handler

```
public class JwtAuthenticationHandler : AuthenticationHandler<authenticationschemeoptions>
{
    private readonly IJwtService _jwtService;

    protected override Task<authenticateresult> HandleAuthenticateAsync()
    {
        var token = Request.Headers.Authorization
            .FirstOrDefault(h => h.StartsWith("Bearer "))?
            .Substring("Bearer ".Length);

        if (string.IsNullOrEmpty(token))
            return Task.FromResult(AuthenticateResult.NoResult());

        try
        {
            var principal = _jwtService.ValidateToken(token);
            var ticket = new AuthenticationTicket(principal, Scheme.Name);
            return Task.FromResult(AuthenticateResult.Success(ticket));
        }
```

Rate Limiting Middleware

```
public class RateLimitingMiddleware
{
    private readonly RequestDelegate _next;
    private readonly IDistributedCache _cache;

    public async Task InvokeAsync(HttpContext context)
    {
        var userId = context.User.GetUserId() ?? context.TraceIdentifier;
        var cacheKey = $"ratelimit:{userId}:{DateTime.UtcNow:yyyyMMddHHmm}";

        var requestCount = await _cache.GetOrCreateAsync(cacheKey, async entry =>
        {
            entry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(1);
            return "1";
        });

        if (int.Parse(requestCount) > 100)
        {
            context.Response.StatusCode = StatusCodes.Status429TooManyRequests;
```

Prompt para Logging Estructurado

```
Implementa logging estructurado con Serilog:

## Formato
{
  "timestamp": "2025-01-23T10:30:00Z",
  "level": "INFO",
  "message": "Tarea created",
  "service": "PortalEmpleo.API",
  "correlationId": "abc-123",
  "userId": "user-456",
  "action": "CreateTarea",
  "durationMs": 45.2
}
```

Logging Middleware

```
Log.Logger = new LoggerConfiguration()
    .WriteTo.Console()
    .WriteTo.File("logs/log-.txt", rollingInterval: RollingInterval.Day)
    .Enrich.FromLogContext()
    .Enrich.WithProperty("Service", "PortalEmpleo.API")
    .CreateLogger();

public class LoggingMiddleware
{
    private readonly RequestDelegate _next;
    private readonly ILogger<loggingmiddleware> _logger;

    public async Task InvokeAsync(HttpContext context)
    {
        var correlationId = context.TraceIdentifier;
        var startTime = DateTime.UtcNow;

        context.Response.Headers["X-Correlation-ID"] = correlationId;
```

2.3 Documentación Automática SDD

2.3 Documentación Automática SDD

- OpenAPI/Swagger Generation desde specs

2.3 Documentación Automática SDD

- OpenAPI/Swagger Generation desde specs
- Documentación de endpoints con ejemplos

Prompt para Documentación OpenAPI

```
@docs-dev Genera documentación OpenAPI desde specs:

## Proceso
1. Leer specs/001-task-api/spec.md
2. Extraer endpoints y DTOs
3. Generar spec OpenAPI
4. Documentar con ejemplos
```

Documentación de Endpoint

````
# Task Management API

## Create Tarea

**Endpoint:** `POST /api/v1/tareas`

**Authentication:** Bearer Token (JWT)

**Request Body:**
```json
{
  "titulo": "string (required, 1-200 chars)",
  "prioridad": "baja | media | alta | urgente",
  "fechaLimite": "ISO8601 datetime (optional)"
}
```

**Responses:**
| Code | Description |
````

2.4 Ejemplo Práctico: API Endpoint Template

Caso: PortalEmpleo genera endpoint REST para gestión de tareas con documentación automática.

Dónde se guarda:

- Template: `.specify/templates/csharp/api-endpoint-template.md`
- Prompt: `specs/001-task-api/api-prompts.md`

API Endpoint Template

```
[ApiController]
[Route("api/v1/[controller]")]
public class [EntityName]Controller : ControllerBase
{
    private readonly I[EntityName]Service _service;

    [HttpGet]
    [ProducesResponseType(typeof([EntityName]ResponseDto), StatusCodes.Status200OK)]
    public async Task<iactionresult> GetAll([FromQuery] PaginationRequest request)
    {
        var result = await _service.GetAllAsync(request);
        return result.Match(Ok, Problem);
    }
}</iactionresult>
```

Integración con Spec Kit

```
# 1. Verificar tasks.md
# TASK-003: Implementar endpoint GET /api/tareas

# 2. Seleccionar y personalizar template
specs/001-task-api/api-prompts.md → Template: api-endpoint-template.md

# 3. Generar código
@backend-dev Aplica template api-endpoint-template.md
             para endpoint GET /api/tareas

# 4. Validar con /speckit.analyze
/specKit.analyze --template api-endpoint-template.md --coverage 90
```

Resultado Generado

```
/// <summary>
/// Obtiene todas las tareas paginadas
/// </summary>
/// <param name="request">Parámetros de paginación
/// <param name="ct">Cancellation token
/// <returns>Lista paginada de tareas</returns>
[HttpGet]
[ProducesResponseType(typeof(PagedResult<tarearesponsedto>), StatusCodes.Status200OK)]
[ProducesResponseType(typeof(ProblemDetails), StatusCodes.Status400BadRequest)]
public async Task<iactionresult> GetAll([FromQuery] PaginationRequest request, CancellationToken ct)
{
    var result = await _tareaService.GetAllAsync(request, ct);
    return result.Match(Ok, Problem);
}</iactionresult></tarearesponsedto>
```

Sección 3: Soporte Multilenguaje y Buenas Prácticas

Coordinación entre tecnologías y estándares transversales

3.1 Coordinación entre Tecnologías

3.1 Coordinación entre Tecnologías

- C# Backend + TypeScript Frontend

3.1 Coordinación entre Tecnologías

- C# Backend + TypeScript Frontend
- Python Data Processing + React Visualization

Prompt para Coordinación Backend-Frontend

```
@agent Genera guía de integración C# + TypeScript:

## Contratos Compartidos
export interface Tarea {
  id: string;
  titulo: string;
  descripcion: string | null;
  estado: 'pendiente' | 'en_progreso' | 'completada' | 'cancelada';
  prioridad: 'baja' | 'media' | 'alta' | 'urgente';
  fechaLimite: string | null;
  creadoEn: string;
}

export interface CreateTareaDto {
  titulo: string;
  descripcion?: string;
  prioridad?: 'baja' | 'media' | 'alta' | 'urgente';
}
```

API Client TypeScript

```
export const tareaApi = {
  getAll: async (params?: { page?: number; pageSize?: number; estado?: string }): Promise<pagedresult<tarea
    const response = await apiClient.get<pagedresult<tarea>>('/api/v1/tareas', { params });
    return response.data;
  },

  getById: async (id: string): Promise<tarea> => {
    const response = await apiClient.get<tarea>(`/api/v1/tareas/${id}`);
    return response.data;
  },

  create: async (dto: CreateTareaDto): Promise<tarea> => {
    const response = await apiClient.post<tarea>('/api/v1/tareas', dto);
    return response.data;
  },

  delete: async (id: string): Promise<void> => {
    await apiClient.delete(`/api/v1/tareas/${id}`);
  }
```

Error Handling Unificado

```
export const handleApiError = (error: unknown): ApiError => {
  if (error instanceof AxiosError && error.response?.data) {
    return error.response.data as ApiError;
  }

  return {
    type: 'unknown',
    title: 'Unexpected error',
    status: 500,
    detail: 'An unexpected error occurred',
    correlationId: 'unknown',
  };
};
```

Prompt para Pipeline Data-Visualization

```
@agent Genera integración Python ML + React visualization:

## Arquitectura
ml-pipeline/          # Python FastAPI
├── src/
│   ├── api/
│   │   └── main.py   # FastAPI endpoint
│   └── models/
│       └── predictor.py
frontend/             # React
├── src/
│   ├── services/
│   │   └── analyticsApi.ts
│   └── screens/
│       └── AnalyticsScreen.tsx
```

Python FastAPI

```python
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import Optional

from src.models.predictor import ProductividadPredictor

app = FastAPI(
    title="Productividad Analytics API",
    description="ML-powered analytics for PortalEmpleo",
    version="1.0.0"
)

predictor = ProductividadPredictor.load("models/production_model.pkl")

class ProductivityRequest(BaseModel):
    usuario_id: str
    periodo_dias: int = 30

class ProductivityResponse(BaseModel):
```

3.2 Estándares Transversales SDD

3.2 Estándares Transversales SDD

- Logging estructurado

3.2 Estándares Transversales SDD

- Logging estructurado
- Configuración centralizada

3.3 Quality Gates SDD

3.3 Quality Gates SDD

- Code Review Checklists

3.3 Quality Gates SDD

- Code Review Checklists
- Métricas automatizadas

Prompt para Quality Gates

```
@agent Configura quality gates automatizados:

## Métricas Objetivo
- Coverage: >= 80%
- Complejidad ciclomática: < 10
- Duplicación: < 3%

## GitHub Actions
- name: Quality Gate
  run: |
    dotnet test --collect:"XPlat Code Coverage"
    if coverage < 80:
      echo "Coverage below 80%"
      exit 1
```

3.4 Ejemplo Práctico: Template Multi-Lenguaje

Caso: PortalEmpleo implementa Task CRUD en C# Backend y TypeScript Frontend.

Dónde se guarda:

- Template C#: `.specify/templates/csharp/crud-template.md`
- Template TS: `.specify/templates/typescript/crud-template.md`
- Contratos: `src/Shared/Contracts/`

Sección 4: Caso Práctico - PortalEmpleo

Implementación de 16 endpoints desde specs

Arquitectura del Proyecto PortalEmpleo

```
portalempleo/
├── backend/                    # C# ASP.NET Core 8
│   ├── src/
│   │   ├── PortalEmpleo.API/
│   │   ├── PortalEmpleo.Application/
│   │   ├── PortalEmpleo.Domain/
│   │   └── PortalEmpleo.Infrastructure/
│   └── tests/
├── ml-pipeline/                # Python 3.12
│   ├── src/
│   │   ├── data/
│   │   ├── models/
│   │   └── api/
│   └── tests/
├── frontend/                   # TypeScript React 18
│   ├── src/
│   │   ├── components/
│   │   ├── screens/
│   │   ├── hooks/
```

Workflow SDD para PortalEmpleo

Workflow SDD para PortalEmpleo

1. **Specify:** Crear `specs/001-task-api/spec.md`

Workflow SDD para PortalEmpleo

1. **Specify:** Crear `specs/001-task-api/spec.md`
2. **Plan:** Generar `plan/architecture_plan.md`

Workflow SDD para PortalEmpleo

1. **Specify:** Crear `specs/001-task-api/spec.md`
2. **Plan:** Generar `plan/architecture_plan.md`
3. **Tasks:** Descomponer en `tasks/implementation_tasks.md`

Workflow SDD para PortalEmpleo

1. **Specify:** Crear `specs/001-task-api/spec.md`
2. **Plan:** Generar `plan/architecture_plan.md`
3. **Tasks:** Descomponer en `tasks/implementation_tasks.md`
4. **Implement:** Usar templates y prompts por lenguaje

Prompt Inicial del Proyecto

```
@agent Inicializa PortalEmpleo con SDD:

## Proyecto
Sistema de gestión de tareas con análisis de productividad

## Stack Tecnológico
- Backend: C# ASP.NET Core 8
- ML Pipeline: Python 3.12
- Frontend: TypeScript React 18

## Generar
1. Estructura de directorios SDD
2. AGENTS.md multi-lenguaje
3. Templates base para cada lenguaje
4. Spec inicial para módulo de tareas
```

Endpoints a Implementar

Módulo	Endpoint	Método	Descripción
Autenticación	/api/v1/auth/register	POST	Registro de usuarios
	/api/v1/auth/login	POST	Inicio de sesión
	/api/v1/auth/refresh	POST	Renovación de token
Ofertas	/api/v1/job-offers	GET	Listar ofertas
	/api/v1/job-offers	POST	Crear oferta
	/api/v1/job-offers/{id}	GET	Detalle de oferta
	/api/v1/job-offers/{id}/status	PATCH	Cambiar estado

Integración Final

Integración Final

- Backend C# expone REST API

Integración Final

- Backend C# expone REST API
- ML Pipeline Python proporciona predicciones

Integración Final

- Backend C# expone REST API
- ML Pipeline Python proporciona predicciones
- Frontend TypeScript consume ambas APIs

Integración Final

- Backend C# expone REST API
- ML Pipeline Python proporciona predicciones
- Frontend TypeScript consume ambas APIs
- Contratos compartidos vía shared-types

Resumen del Módulo 3

- Generación de código desde specs en múltiples lenguajes
- Patrones de Clean Architecture y SOLID
- Desarrollo de APIs REST robustas con SDD
- Middlewares de autenticación y logging
- Coordinación polyglot con Spec Kit
- Quality gates para código generado

3 horas 30 minutos completados

Recursos Adicionales

- GitHub Spec Kit Repo
- Microsoft Learn - Spec-Driven Development
- GitHub Blog - AGENTS.md Best Practices
- Uncle Bob - Clean Architecture
- SOLID Principles

Gracias

Módulo 3 Completado

Siguiente: Modulo 4 - Testing, Refactorización, Documentación y DevOps