netmind | a bts company

Simulación Scrum - SDD

PortalEmpleo API - Planificación de Iteraciones

Spec-Driven Development con Scrum

Sprint 0 - Foundations

Configurar entorno SDD

- spec-kit init para inicializar el proyecto
- Verificar instalación de .NET 8.0 SDK
- Configurar extensiones de Copilot en VS Code
- Verificar configuración de Git

Generar constitution.md

- Ejecutar /speckit.constitution
- Definir estándares tecnológicos (.NET 8, Clean Architecture)
- Establecer requisitos de seguridad (JWT, BCrypt)
- Documentar estándares de codificación

Crear AGENTS.md

- Definir agentes especializados (@backend-dev, @data-dev, @test-dev)
- Establecer stack tecnológico y convenciones
- Configurar comandos ejecutables
- Definir límites (Always/Ask/Never)

Documentar ADRs

- ADR-001: Clean Architecture con 4 capas
- ADR-002: Entity Framework Core con In-Memory
- ADR-003: JWT HS256 para autenticación
- ADR-004: BCrypt work factor 12 para passwords

Generar diagramas de arquitectura

- Diagrama de arquitectura general (capas)
- Diagrama de flujo de autenticación
- Diagrama de ciclo de ofertas (borrador → publicada → pausada → cerrada)
- Diagrama de flujo de postulaciones

Review: Validar artefactos fundacionales

- Verificar constitution.md con todos los estándares
- Revisar AGENTS.md con agentes especializados
- Confirmar ADRs documentados
- Validar diagramas de arquitectura

Sprint 1 - Especificación Auth

Generar spec.md para Autenticación

- RF-001: Registro de usuarios (email único, password compleja, edad ≥16)
- RF-002: Login (JWT 60min/7days, BCrypt, rate limiting)
- RF-003: Refresh token
- RF-004: GET /users/me (perfil propio)
- RF-005: PUT /users/me (update de perfil)
- RF-006: DELETE /users/me (soft delete)

Clarificación y aprobación de especificación

- Ejecutar /speckit.clarify para identificar gaps
- Responder Open Questions
- Completar checklist de aprobación (funcional/técnica/SDD)
- Actualizar spec.md con respuestas

Review: Demo de spec.md aprobado

- Presentar spec.md para Autenticación
- Verificar que todos los RFs están cubiertos
- Confirmar criterios de aceptación medibles
- Aprobación formal del Product Owner

Sprint 2 - Especificación Jobs + Plan Auth

Generar spec.md para Ofertas

- RF-007: Crear oferta (BORRADOR, title max 200, contract types)
- RF-008: Publicar oferta (BORRADOR → PUBLICADA)
- RF-009: Listar ofertas (filtros q, location, contractType, salary)
- RF-010: GET /{id} (detalle de oferta)
- RF-011: PUT /{id} (editar oferta)
- RF-012: Estados de oferta (BORRADOR/PUBLICADA/PAUSADA/CERRADA)

Generar Plan.md para Autenticación

- Estructura de proyectos (.NET 8, Clean Architecture)
- Dependencias NuGet (EF Core, JWT, BCrypt, FluentValidation)
- Fases de implementación (Domain → Infrastructure → Application → API)
- Dependencias entre tareas

Generar Tasks.md para Auth

- T001: Crear entidades de dominio (User, UserRole)
- T002: Implementar interfaces de repositorio
- T003: Crear UserRepository
- T004: Implementar AuthService
- T005: Crear AuthController
- T006: Generar tests unitarios

Review: Demo de spec.md Jobs y Plan Auth

- Presentar spec.md para Ofertas de Empleo
- Mostrar Plan.md para Autenticación
- Demostrar Tasks.md con desglose de tareas
- Aprobación de planificación

Sprint 3 - Plan Jobs + Implementación Auth

Generar Plan.md para Ofertas

- Estructura de proyectos para módulo de ofertas
- Entidades: JobOffer, JobOfferStatus, ContractType
- Servicios: JobOfferService con CRUD y transiciones de estado
- Controladores: endpoints públicos y privados

Implementar entidades y repositorios (Auth)

- User.cs: Id, Email, PasswordHash, Name, BirthDate, Phone, Role, IsActive
- UserRole enum: CANDIDATE, COMPANY, ADMIN
- IUserRepository interface
- UserRepository implementación

Generar controladores y servicios (Auth)

- AuthController: POST /auth/register, /auth/login, /auth/refresh
- UsersController: GET /users/me, PUT /users/me, DELETE /users/me
- AuthService: RegisterAsync, LoginAsync, RefreshTokenAsync
- DTOs con FluentValidation

Verificación de compilación

- dotnet build --no-restore
- Corregir errores de compilación
- Ejecutar tests unitarios
- Verificar cobertura ≥80%

Review: Demo endpoints Auth

- Demostrar registro de usuario
- Mostrar login con generación de JWT
- Probar refresh token
- Verificar perfiles y soft delete

Sprint 4 - Implementación Jobs

Implementar entidades y repositorios (Jobs)

- JobOffer.cs: Id, Title, Description, Location, Salary, ContractType, Status
- JobOfferStatus enum: BORRADOR, PUBLICADA, PAUSADA, CERRADA
- ContractType enum: TIEMPO_COMPLETO, MEDIO_TIEMPO, TEMPORAL, FREELANCE
- IJobOfferRepository, JobOfferRepository

Generar controladores Jobs

- GET /job-offers (público con filtros)
- POST /job-offers (COMPANY)
- GET /{id} (público)
- PUT /{id} (COMPANY own)
- PATCH /{id}/publish, /pause, /close (COMPANY own)

Implementar servicios de ofertas

- CreateAsync: validación y creación en BORRADOR
- PublishAsync: BORRADOR → PUBLICADA (irreversible)
- ListAsync: filtros, paginación (default 10, max 100)
- UpdateAsync: solo BORRADOR o PAUSADA
- Pause/Close: transiciones de estado

Generar eventos y handlers

- JobOfferPublishedEvent
- ApplicationSubmittedEvent
- Event handlers para notificaciones
- Observer Pattern implementado

Review: Demo endpoints Jobs

- Crear oferta en borrador
- Publicar oferta
- Listar con filtros
- Probar transiciones de estado

Sprint 5 - Implementación Apps + Testing

Implementar entidades, repositorios y servicios (Apps)

- Application.cs: Id, JobOfferId, CandidateId, Status, AppliedAt, Notes
- ApplicationStatus enum: PENDIENTE, EN_REVISION, ENTREVISTADO, ACEPTADO, RECHAZADO
- IApplicationRepository, ApplicationRepository
- ApplicationService: ApplyAsync, GetMyApplicationsAsync, UpdateStatusAsync

Generar Tests Unitarios

- AuthServiceTests: Register validación email único, Login credenciales
- UserServiceTests: Update perfil, Soft delete
- JobOfferServiceTests: Publicar oferta, Listar con filtros
- ApplicationServiceTests: Postulación única, Cambio de estado

Generar Tests de Integración

- AuthControllerTests: Register, Login, Refresh token
- JobOffersControllerTests: CRUD ofertas, filtros, paginación
- ApplicationsControllerTests: Postulación, estados
- Framework: xUnit + Moq + FluentAssertions

Verificar cobertura ≥80%

- dotnet test --collect:"XPlat Code Coverage"
- Analizarレポート de cobertura
- Añadir tests para coverage gaps
- Validar coverage mínimo alcanzado

Review: Demo con tests passing

- Ejecutar suite completa de tests
- Mostrar coverage report
- Demostrar funcionalidades con tests passing
- Aprobación de Quality Gate

Sprint 6 - Refactorización

Análisis de code smells

- Long Method (métodos >30 líneas)
- Large Class (clases >500 líneas)
- Feature Envy
- Data Clumps
- Primitive Obsession

Ejecutar refactorización

- Extract Method para métodos largos
- Replace Primitive with Object
- Move Method a clases apropiadas
- Introduce Parameter Object

Validación final de tests

- Todos los tests pasan después de refactorizar
- Coverage se mantiene ≥80%
- Verificar funcionalidad no afectada
- Documentar refactorizaciones realizadas

Demo Final completa

- Demostrar PortalEmpleo API completo
- Flujo completo de usuario (registro → login → buscar ofertas → postularse)
- Flujo de empresa (crear oferta → publicar → gestionar postulaciones)
- Verificación vs todos los spec.md

Sprint 7 - Documentación

Checklist de Preparación

- Verificar que toda funcionalidad está implementada
- Confirmar que todos los tests pasan
- Revisar cobertura de código ≥80%
- Validar compilación sin warnings

Generar Spec.md para Documentación

- Definir estructura de documentación del proyecto
- Documentar decisiones de arquitectura
- Especificar formato de API docs (Swagger/OpenAPI)
- Definir estándares de XML Documentation

Generar Plan.md para Documentación

- Planificar generación de documentación técnica
- Asignar tareas de documentación
- Definir timeline de revisión
- Establecer criterios de calidad

Generar Documentación API (Swagger/OpenAPI)

- Configurar Swashbuckle para ASP.NET Core
- Documentar todos los endpoints con OpenAPI 3.0
- Incluir ejemplos de request/response
- Documentar códigos de error

Generar XML Documentation

- Documentar todas las clases públicas
- Documentar métodos con parámetros y return values
- Incluir remarks y examples
- Habilitar generación de docs desde build

Actualizar README.md

- Descripción del proyecto
- Instrucciones de instalación y configuración
- Guía de uso con ejemplos
- Documentar variables de entorno

Sprint 8 - DevOps

Configurar Quality Gates

- Análisis estático con SonarQube o similar
- Cobertura mínima de tests ≥80%
- Validación de convenciones de código
- Check de vulnerabilidades de seguridad

Generar Spec.md para DevOps

- Definir requisitos de infraestructura
- Especificar pipeline de CI/CD
- Documentar estrategia de despliegue
- Definir environment configuration

Generar Plan.md para DevOps

- Planificar arquitectura de pipeline
- Definir stages del CI/CD
- Planificar estrategias de testing automatizado
- Documentar secretos y variables sensibles

Generar Pipeline CI/CD (GitHub Actions)

- Workflow de build y test
- Workflow de despliegue a staging
- Workflow de despliegue a producción
- Integración con quality gates

Generar Pipeline de Tests Automatizado

- Ejecución de unit tests en cada push
- Ejecución de integration tests
- Generación de reportes de cobertura
- Notificación de resultados

Configurar Análisis de Coherencia SDD

- Verificar implementación vs spec.md
- Validar consistencia entre módulos
- Revisar trazabilidad de requirements
- Generar report de coherencia

Retrospective del curso

- ¿Qué funcionó bien con SDD + Scrum?
- ¿Qué mejoraríamos en próximos proyectos?
- Lecciones aprendidas sobre agentes IA
- Propuestas de mejora a constitution.md

Gracias

Simulación Completada