M7_AE2_ABP-Ejercicio individual[Actividad Evaluada]

Pablo Varas Salamanca

24 de noviembre de 2025, 09:34:00 h

Revisión del Sistema de Capacitaciones

Instrucciones

El objetivo de esta actividad es realizar una **revisión exhaustiva** del proyecto desarrollado a lo largo del curso, identificar áreas de mejora, aplicar ajustes finales y documentar el proceso de aprendizaje.

La consigna solicita:

1. **Revisión del Producto**
   - Abrir el proyecto y examinar cada componente.
   - Responder en un documento:
     - ¿Qué funcionalidades implementaste con éxito?
     - ¿Qué partes aún podrían mejorarse?
     - ¿Hay errores o fallos identificados durante su uso?
2. **Depuración y Mejora**
   - Corregir errores en código, diseño o funcionalidad.
   - Refactorizar el código si es necesario para mejorar su legibilidad y eficiencia.
   - Agregar documentación clara en comentarios o en el `README` si aún no se ha hecho.
3. **Feedback y Retroalimentación**
   - Compartir el proyecto con un compañero o instructor y solicitar una retroalimentación honesta.
   - Registrar al menos **tres puntos de mejora** sugeridos.
   - Explicar qué cambios se implementarán basados en el feedback recibido.
4. **Ajustes Finales y Cierre**
   - Realizar ajustes finales en base al feedback.
   - Asegurarse de que toda la funcionalidad esté correctamente probada.
   - Preparar la versión final del entregable con:
     - Código limpio y estructurado.
     - Documentación del proyecto (README, instrucciones de uso, dependencias).
     - Evidencias (capturas de pantalla, enlace al repositorio, demostración en video si aplica).
5. **Entrega del Proyecto**

- Adjuntar un informe breve con:
  - Una descripción final del producto.
  - Las mejoras aplicadas tras la revisión.
  - Una reflexión personal sobre la experiencia en el desarrollo del proyecto.

Además de responder a estos puntos, en este informe integro evidencias adicionales solicitadas en la **Rúbrica Técnica Java** (consultas SQL, algoritmo de cálculo, pruebas unitarias y uso de polimorfismo).

Desarrollo

1. Revisión del producto

- 1.1 Funcionalidades implementadas con éxito

  El proyecto revisado es el **Sistema de Capacitaciones**, una aplicación web desarrollada con Spring Boot 3 que permite gestionar cursos de formación interna para una organización. Las funcionalidades que actualmente se encuentran implementadas con éxito son:

  - **Autenticación y roles de usuario** Uso de Spring Security para diferenciar los roles `ADMIN` y `EMPLEADO`. Después de iniciar sesión, cada usuario es redirigido a la sección que le corresponde.
  - **Gestión de cursos (rol ADMIN)**
    - Listado de todos los cursos disponibles.
    - Creación de nuevos cursos mediante un formulario web.
    - Edición y eliminación de cursos existentes.
    - Asignación de un instructor a cada curso.
  - **Gestión de inscripciones (rol EMPLEADO)**
    - Visualización del catálogo de cursos disponibles.
    - Inscripción del empleado en un curso, evitando duplicidad de inscripciones.
    - Descuento automático de cupos al generar una nueva inscripción.
  - **Persistencia de datos** Uso de Spring Data JPA con base de datos H2 en memoria. Las entidades `Curso`, `Empleado`, `Instructor` e `Inscripcion` representan el modelo del dominio y se relacionan mediante anotaciones JPA.
  - **API REST**
    - `GET /api/cursos`: entrega la lista de cursos en formato JSON.
    - `POST /api/inscripciones`: permite crear una inscripción desde un cliente externo utilizando un cuerpo JSON validado con `@Valid`.
  - **Interfaz web con Thymeleaf y Bootstrap** Las vistas para administrador y empleado están implementadas con Thymeleaf, siguiendo una estructura HTML5 correcta y utilizando clases de Bootstrap para lograr una presentación limpia y responsiva.

  En síntesis, el sistema cubre de forma efectiva el ciclo básico de publicación de cursos e inscripción de empleados, tanto desde la interfaz web como desde la API REST.

- 1.2 Partes del sistema que podrían mejorarse

  La revisión también permitió identificar aspectos perfectibles:

  - **Experiencia de usuario** Sería conveniente añadir filtros por nombre de curso, modalidad o instructor en el listado de cursos del empleado, así como indicadores visuales claros para estados como “sin cupos” o “ya inscrito”.

- **Reportes y métricas** Actualmente el administrador puede ver los cursos e inscripciones, pero no existe un módulo de reportes. Sería útil contar con estadísticas por curso (ocupación, cantidad de inscritos, etc.) y por empleado.
- **Cobertura de pruebas** Existen pruebas unitarias iniciales, pero no se han automatizado todos los casos de uso clave (particularmente los flujos web y los controladores REST).
- **Personalización de páginas de error** El manejo de errores HTTP utiliza, en algunos casos, la página genérica de Spring. Se podrían diseñar páginas 404/400/500 más amigables y coherentes con la identidad del sistema.

- 1.3 Errores o fallos identificados durante su uso

  Durante la revisión y las pruebas manuales se observaron los siguientes potenciales problemas o riesgos:

  - Si se prueban casos límite (por ejemplo, muchos usuarios intentando inscribirse al mismo curso simultáneamente) es necesario verificar que la lógica de cupos se mantenga consistente.
  - La configuración de validación podría ampliarse para asegurar que no se puedan enviar datos inválidos desde clientes externos que llamen a la API.
  - Algunas rutas podrían mostrar la página de error por defecto cuando la sesión expira; sería preferible redirigir siempre a `/login`.

  Estos puntos alimentan las decisiones de depuración y mejora descritas en el apartado siguiente.

2. Depuración y mejora

- 2.1 Correcciones en código, diseño y funcionalidad

  A partir de la revisión se realizaron o planificaron las siguientes correcciones:

  - Revisión de la lógica de inscripción para asegurar que un empleado no pueda inscribirse en cursos sin cupos disponibles.
  - Ajuste de la actualización de cupos al crear una inscripción (`InscripcionService`), asegurando que el valor no se vuelva negativo.
  - Verificación de que las rutas protegidas por Spring Security sólo sean accesibles para usuarios autenticados con el rol adecuado.

  Estas correcciones se orientan a fortalecer la robustez del sistema, evitar errores silenciosos y mejorar la coherencia del comportamiento de la aplicación.

- 2.2 Refactorización y documentación

  - Se mantuvo y reforzó la separación en capas: controladores, servicios, repositorios y modelo.
  - Se creó el servicio `EstadisticasCapacitacionesService` para aislar el **algoritmo de cálculo** del porcentaje de ocupación de un curso (`inscritos / cupos * 100`), utilizando una sentencia repetitiva (`for-each`) para recorrer la lista de inscripciones.
  - Se introdujo la interfaz `NotificadorInscripcion` con dos implementaciones (`NotificadorConsola` y `NotificadorSilencioso`), lo que ejemplifica el uso de polimorfismo y mejora la extensibilidad de la lógica de notificación.
  - Se amplió la documentación en el `README.md` y se añadieron comentarios explicativos en clases clave (`InscripcionService`, `EstadisticasCapacitacionesService`, controladores REST).

  Estas mejoras facilitan la mantención futura del código, su comprensión por parte de otros desarrolladores y el alineamiento con la rúbrica técnica.

3. Feedback y retroalimentación

- 3.1 Sugerencias recibidas

  Tras compartir el proyecto y la rúbrica con un tercero, se identificaron al menos tres líneas de trabajo:

  1. **Visibilidad de mensajes al usuario** Incluir mensajes de éxito y error claramente visibles en la interfaz (por ejemplo, alertas de Bootstrap) para la inscripción y otros procesos críticos.
  2. **Evidencias técnicas adicionales** Incorporar ejemplos explícitos de consultas SQL, un algoritmo de cálculo aislado y pruebas unitarias, de modo que el proyecto evidencie de forma clara el uso de los contenidos del módulo.
  3. **Organización del portafolio** Documentar en un archivo específico cómo el proyecto cumple con cada punto de la Rúbrica Técnica Java, para facilitar la evaluación y su uso posterior como evidencia en el portafolio.

- 3.2 Cambios implementados en base al feedback

  En respuesta a estas sugerencias:

  - Se crearon consultas SQL de ejemplo en un archivo `docs/consultas_bd.sql` que cubren SELECT, JOIN, WHERE, ORDER BY y GROUP BY sobre las tablas del sistema.
  - Se desarrolló el servicio `EstadisticasCapacitacionesService` y su clase de prueba `EstadisticasCapacitacionesServiceTest` para dejar evidencia del uso de algoritmos y pruebas unitarias.
  - Se elaboró el documento `docs/RUBRICA_CUMPLIMIENTO_M6_M7.md`, donde se indica explícitamente qué parte del código o de la documentación responde a cada criterio de la rúbrica técnica.

4. Ajustes finales y pruebas

- 4.1 Pruebas funcionales del sistema

  Para cerrar el proceso se realizaron pruebas manuales de los principales flujos:

  - **Flujo ADMIN**
    - Iniciar sesión como administrador.
    - Crear, editar y eliminar cursos de prueba.
    - Verificar que las modificaciones se reflejen en la vista del empleado.
  - **Flujo EMPLEADO**
    - Iniciar sesión como empleado.
    - Revisar el catálogo de cursos y realizar una inscripción.
    - Confirmar que los cupos se actualizan correctamente y que no se permiten inscripciones duplicadas.
  - **Flujo API REST**
    - Ejecutar `GET /api/cursos` para comprobar que devuelve el JSON esperado.
    - Ejecutar `POST /api/inscripciones` con un cuerpo válido y verificar que se crea la inscripción y se descuenta un cupo.

- 4.2 Pruebas unitarias y revisión general

- Se corrieron las pruebas JUnit de `EstadisticasCapacitacionesServiceTest` para verificar el cálculo correcto del porcentaje de ocupación en casos con y sin cupos.
- Se realizó una revisión rápida del árbol de clases y paquetes para asegurar consistencia en nombres y responsabilidades.

Con estas acciones se considera que la versión actual del Sistema de Capacitaciones está en condiciones de ser incluida en el portafolio del módulo.

5. Informe final y reflexión personal

- 5.1 Descripción final del producto

  El **Sistema de Capacitaciones** es una aplicación web pensada para apoyar la gestión de cursos internos en una organización. Permite a un administrador dar de alta cursos, asignar instructores y controlar cupos, mientras que los empleados pueden revisar el catálogo e inscribirse en las alternativas disponibles.

  El proyecto utiliza una arquitectura típica de aplicaciones Spring Boot: controladores web y REST, servicios de negocio, repositorios basados en Spring Data JPA y entidades de dominio. Las vistas se construyen con Thymeleaf y Bootstrap, y la seguridad se implementa con Spring Security.

- 5.2 Mejoras aplicadas tras la revisión

  Las principales mejoras logradas en esta actividad son:

  - Clarificación de la lógica de inscripción y actualización de cupos.
  - Creación de un servicio dedicado a estadísticas con un algoritmo de cálculo explícito y probado.
  - Incorporación de polimorfismo a través de la interfaz `NotificadorInscripcion`.
  - Añadido de consultas SQL de ejemplo y de un documento de trazabilidad con la rúbrica técnica.
  - Ampliación de la documentación general del proyecto y de las evidencias incluidas en el portafolio.

- 5.3 Reflexión personal sobre el proceso

  Esta actividad me permitió mirar el Sistema de Capacitaciones no sólo como un conjunto de funcionalidades que "funcionan", sino como un **producto técnico evaluable** según criterios claros (rúbrica). Al revisar el código, corregir detalles, agregar pruebas y documentar la relación con cada punto de la rúbrica, comprendí mejor la importancia de la calidad interna del software: legibilidad, estructura en capas, pruebas y documentación.

  Además, integrar este proyecto al portafolio de productos me ayuda a mostrar de forma concreta cómo aplico Java, Spring Boot y buenas prácticas de desarrollo en un contexto real vinculado a la educación de adultos. La revisión final y la redacción de este informe también me dan insumos para seguir mejorando el sistema en futuras iteraciones.

Bibliografía

- Material `M7_AE2_ABP-Ejercicio individual [Actividad Evaluada]`, Plataforma Skillnest.
- `Rúbrica Técnica Java`, documento de evaluación del módulo.
- Documentación oficial de Spring Boot y Spring Data JPA (`https://spring.io`).
- Documentación de Thymeleaf (`https://www.thymeleaf.org`) y Bootstrap (`https://getbootstrap.com`).