Apartados	Excelente	Bien	Suficiente	Insuficiente	Muy deficiente	Puntuación
Programa componentes de acceso a datos identificando las características que debe poseer un componente y utilizando herramientas de desarrollo.	Los componentes de acceso a datos están correctamente diseñados siguiendo principios de abstracción, reutilización y bajo acoplamiento. Se aplican patrones como Repositorio o DAO, y se organizan en capas claras. Se emplean interfaces, genéricos y buenas prácticas de codificación, facilitando la mantenibilidad y extensibilidad del sistema.	Los componentes están bien estructurados, con una separación clara entre la lógica de negocio y el acceso a datos. Se aplica algún patrón básico y el diseño permite su reutilización en otros contextos. La implementación es coherente y funcional.	Se programan componentes funcionales pero con estructuras rígidas o acopladas. La lógica de acceso a datos está mezclada parcialmente con la lógica de negocio o de presentación, y no se aplican patrones conocidos.	El código de acceso a datos carece de organización, se repite o se incluye directamente en otros componentes. No hay una separación clara de responsabilidades ni uso intencional de técnicas de diseño.	No se han creado componentes diferenciados. El acceso a datos se hace de forma desorganizada, con lógica dispersa y sin uso de técnicas de programación estructurada o herramientas adecuadas.	30%
Se han probado y documentado los componentes desarrollados.	Todos los componentes han sido probados con pruebas unitarias y de integración automatizadas usando JUnit. Documentación completa y clara con Swagger/OpenAPI y comentarios en el código.	Se incluyen pruebas unitarias o de integración para los componentes principales. La documentación es adecuada, aunque podría mejorarse la cobertura o el uso de Swagger.	Se realizan algunas pruebas manuales o automatizadas, pero no se cubren todos los casos. La documentación es escasa o poco estructurada.	Hay pocas pruebas o están mal implementadas. La documentación es mínima, y no se utiliza Swagger ni herramientas similares.	No hay pruebas ni documentación. El funcionamiento debe deducirse revisando directamente el código.	30%
Desarrolla aplicaciones de acceso a almacenes de datos, aplicando medidas para mantener la seguridad y la integridad de la información.	La API implementa seguridad robusta con Spring Security, JWT, cifrado de contraseñas, validación de datos y control de roles/autorizaciones. Se maneja la integridad de datos con validaciones.	La aplicación incorpora autenticación básica y roles, y protege adecuadamente los endpoints más críticos. Se manejan validaciones y errores de forma correcta.	Se protege el acceso mediante autenticación simple. Hay validaciones, pero limitadas, y sin gestión completa de errores o roles.	Medidas de seguridad superficiales o incompletas. Hay vulnerabilidades evidentes y falta de control de acceso o validaciones.	No se han aplicado medidas de seguridad ni validaciones. Los datos y endpoints están completamente expuestos.	40%