ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA EN SOFTWARE

MANUAL DE ESTÁNDARES DE DESARROLLO DE SOFTWARE PARA LA PROGRAMACIÓN DE HUMAN SCHEDULE

GRUPO 3

DOCENTE: ING PATRICIO PACCHA

QUITO, AGOSTO 2024

Contenido

1. Propósito

El propósito fundamental de este proyecto es implementar un sistema de registro de asistencias para empleados mediante un lector de código de barras, que fue proporcionado por la empresa, asegurando una gestión precisa y segura de las horas laborales. Este sistema no solo registra con exactitud las horas de entrada y salida, sino que también garantiza un control de asistencia confiable. Además, su integración facilita la gestión de recursos humanos al optimizar la administración del tiempo de trabajo tanto diariamente como mensualmente.

2. Destinatario

Este manual está diseñado para los integrantes del proyecto y para aquellos interesados en la Programación Orientada a Objetos (POO), mediante JAVA y NetBeans IDE 22; sin mencionar del uso de base de datos de SQLite. Su finalidad es indicar pautas claras, acerca de la nomenclatura de clases, variables, interfaces, nombres de las clases, entre otros elementos. Estas directrices aseguran la consistencia y precisión en el formato a lo largo del desarrollo del proyecto, incluyendo actualizaciones, correcciones de errores y futuras iteraciones. El propósito fundamental es facilitar la comprensión del código y fomentar la uniformidad, lo que conduce a un desarrollo más eficiente y mantenible durante su época de funcionamiento. Esto no solo mejora la colaboración entre los desarrolladores, sino que también permite una mejor gestión del código a largo plazo, reduciendo errores y facilitando futuras actualizaciones y modificaciones.

3. Lineamientos a seguir

3.1 Estilo de codificación

Emplear un estilo de codificación adecuado, incluyendo la convención de nomenclatura, sangrías, uso de espacios en blanco y comentarios. Es por ello necesario aclarar que, durante la codificación del código, principalmente se priorizó el uso del estilo LowerCamelCase, mientras que en los DDL y DML se utilizó principalmente el estilo UpperCamelCase.

3.2 Documentación

Implementar comentarios claros y precisos que expliquen el propósito y la funcionalidad del código, principalmente en partes complejas para facilitar su comprensión y mantenimiento. Junto con ello, los comentarios estarán colocados de manera que se pueda genera un javaDoc.

3.3 Uso de herramientas de gestión de versiones

Ocupar una herramienta de control de versiones, como Git, para facilitar la colaboración en equipo y el seguimiento de cambios en el código.

3.4 Pruebas y control de calidad

Verificar que el programa cumpla con las funciones esperadas mediante pruebas exhaustivas:

- **Pruebas funcionales:** Confirmar que cada función opere según lo esperado.
- **Pruebas de rendimiento:** Evaluar la velocidad y eficiencia del programa.
- **Pruebas de carga y estrés:** Asegurar que el software maneje condiciones extremas sin fallos.
- **Pruebas de usabilidad**: Garantizar que el producto sea intuitivo y accesible.
- **Pruebas de seguridad:** Detectar y corregir vulnerabilidades.
- Utilizar pruebas automatizadas y manuales para mantener un ciclo de desarrollo de alta calidad.

3.5 Mantenibilidad y escalabilidad

Realizar actualizaciones periódicas (anualmente) del software para incluir nuevas características, mejoras de seguridad y correcciones de errores, manteniendo el sistema actualizado y funcional de acuerdo con la actualidad de Ecuador.

4. Estilo de codificación y convenciones de nomenclatura

4.1 Nomenclatura de variables y métodos

Se utilizará el formato lowerCamelCase, en el cual las palabras se escriben juntas, es decir sin espacios, la palabra empezara con una letra minúscula y cada palabra después de la primera comenzarán con una letra en mayúscula. Además, es recomendable el uso de nombres descriptivos para mejorar la comprensión del código. Por otra parte, los métodos deben ser verbos o un conjunto de palabras que comiencen con un verbo y los nombres de las variables no deben comenzar con los caracteres guion bajo o el signo de dólar.

4.2 Nomenclatura de clases e interfaces

Se manejará el formato UpperCamelCase, que guarda similitud con el CamelCase solo que la primera letra de cada palabra será en mayúscula. Para las interfaces se coloca una I en mayúscula al principio del nombre. Es importante recordar que todas las clases e interfaces se escriben en singular.

Clases:

- Nombres descriptivos y específicos.
- Utilizar sustantivos o frases sustantivas.

Ejemplos:

Customer, Order, InvoiceProcessor.

Interfaces:

- Nombres que representen capacidades o roles.
- Prefijo 'I' para indicar que es una interfaz.

Ejemplos:

ICalculable, ISerializable, IUserService.

4.3. Nomenclatura de paquetes y archivos

Se nombrarán con el estilo UpperCamelCase, los archivos de acuerdo con los tipos de datos guardados y los paquetes en relación con su funcionalidad.

Los archivos de código deben nombrarse según la clase o interfaz principal que contienen. Los nombres deben ser descriptivos y reflejar la funcionalidad principal del archivo.

Ejemplos:

ClienteServicio.java, ProductoControlador.java

Subpaquetes: Utilizar subpaquetes para organizar el código de manera lógica y modular.

5. Bibliotecas externas

- **java.sql. *;**

Proporciona un grupo de clases que permitirá generar una conexión con SQLite que será de gran utilidad, durante la ejecución del CRUD (create, read, udate, delete).

- **java.time.LocalDateTime;**

Permite la lectura de datos de tiempo actual de la localidad.

- **java.util.*List*;**

Ofrece una interfaz en Java que representa una colección ordenada y indexada de elementos.

- **java.time.DayOfWeek;**

Simplifica la escritura ya que es una enumeración en Java que representa los días de la semana, desde Lunes (MONDAY) hasta Domingo (SUNDAY).

- **javafx.fxml.FXML;**

Permite marcar campos en un controlador de vista (controller) que están asociados con elementos de la interfaz de usuario definidos en un archivo FXML.

- **javafx.fxml.*Initializable*;**

Permite implementar el método Initializable es para realizar configuraciones adicionales o inicializar datos que dependan de la vista y sus elementos.

- **javafx.scene.control.*;**

Ofrece un paquete en JavaFX que contiene clases y componentes para construir interfaces de usuario gráficas (GUI).

6 . Referencias

[1] L. Amparo, «UNAM,» 12 5 2019. [En línea]. Available: http://hp.fciencias.unam.mx/~alg/normas/estilo.html. [Último acceso: 8 1 2024].

[2] D. T. Agency, «Digital Talent Agency,» 17 4 2023. [En línea]. Available: https://dtagency.tech/10- buenas-practicas-para-programadores/. [Último acceso: 8 1 2024].

[3] A. U. Cartagena99, «Academia Cartagena99,» 5 3 2021. [En línea]. Available: https://www.cartagena99.com/recursos/alumnos/apuntes/210302115131-Tema4.pdf. [Último acceso: 8 1 2024].