Review & Refactor

1. `messageText.trim();` antes de validar si `messageText` no es null
2. Código repetido

```
. dbParams.get("logFileFolder") + "/logFile.txt"
```

```
. if (message && logMessage) {
  if (error && logError) {
  if (warning && logWarning) {
```

3. Formación de mensaje de log de manera ineficiente, concatenación de strings
4. Variable no utilizadas

```
private boolean initialized;
String l
```

5. Mensaje con nombre en mayúscula

```
public static void LogMessage(..)
```

6. Los mensajes en consola y en Fiel se registrar en level INFO, no son marcados como error, advertencia o mensaje y no se loguean selectivamente.
7. Se entiende que el `String l` es quien arma el mensaje a loguear.
8. Según el código dado entiendo que `logMessage, logWarning, logError` son las atributos por lo que se determina qué tipo de mensajes va a loguear nuestro Logger.
9. Dado `LogMessage(String messageText, boolean message, boolean warning, boolean error)`
    a. entiendo que están manejando el level del mensaje a loguear por medio de variables booleanas.
    b. Cada message puede ser de un solo tipo a la vez. Por lo que se va a definir un TypeMessage.
    c. Asumimos que se puede cambiar la firma del metódo.

10. La validación por el valor de las variables está mal puesto en el método logMessage, ya que son valores seteados en el momento de construcción

```
if (!logToConsole && !logToFile && !logToDatabase) {
      throw new Exception("Invalid configuration");
}


if (!logError && !logMessage && !logWarning) {
      throw new Exception("Error or Warning or Message must be
      specified");
}
```

11. Filename debería ser parte de los dbParams y no harcoded `"/logFile.txt"`

12. Se refactorizó usando Factory Method en dos oportunidades, `ILogger` y `LoggerCreator`