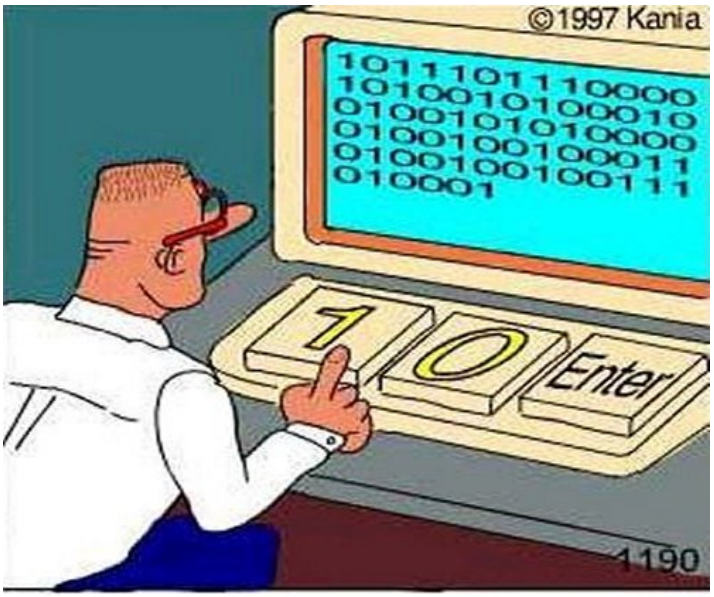
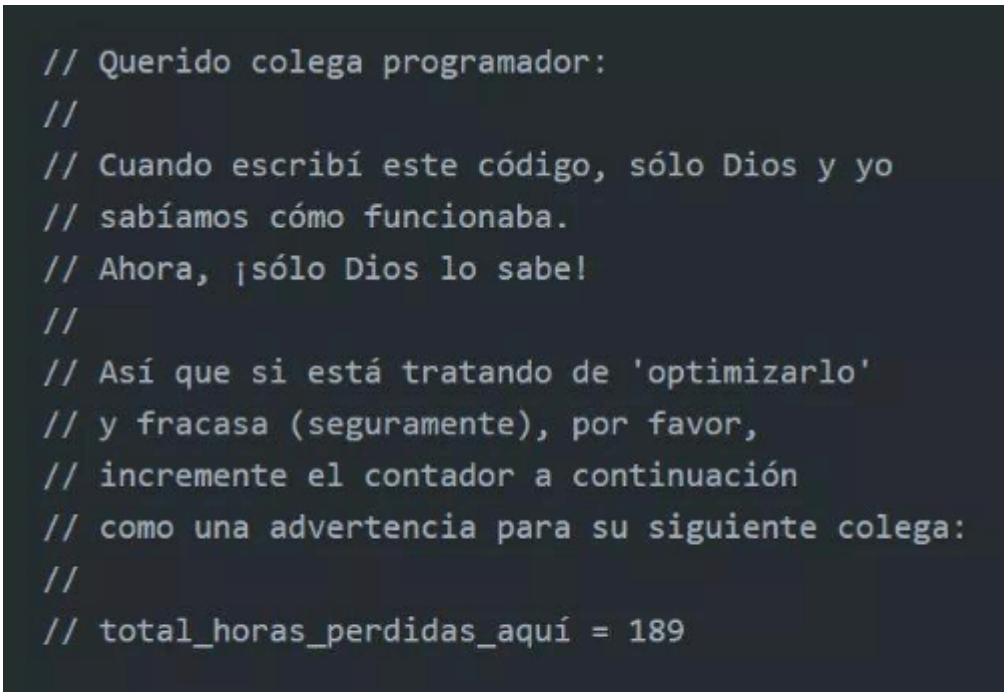
1. **Siempre define las funcionalidades de tu sistema**, por que te preguntaras, pero realizar este paso hará que puedas completar tus tareas y finalizarlas a tiempo, ya que si no lo defines, a medida que vas creando funcionalidades siempre se te va a ocurrir algo más que agregarle y al menos en mi caso, este se vuelve en un círculo vicioso

2. **Estandarizar las reglas del desarrollo**: Si trabajas en equipo **crea un documento que estandarice** la manera de codificar: esto suele ser de gran ayuda al momento de modificar o encontrar algún error específico en el código, ya que definirlas desde un principio como se declaran las variables, funciones, clases o constantes y sera mas facil entender el codigo que vos no escribiste.(Véase el anexo 1)



LOS VERDADEROS PROGRAMADORES PROGRAMAN EN BINARIO

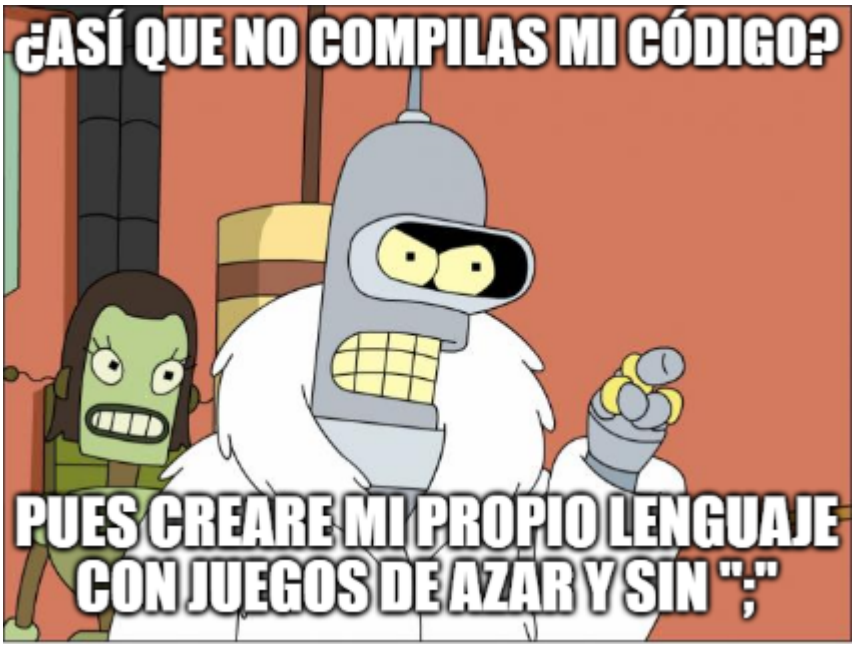
3. **Crear código limpio y coméntalo**: esto es muy importante no solo para poder crear la documentación de tus sistema sino que te servirá para que otras personas puedan entender lo que programaste, o incluso a ti mismo cuando lo tengas que modificar.



```
// Querido colega programador:
//
// Cuando escribí este código, sólo Dios y yo
// sabíamos cómo funcionaba.
// Ahora, ¡sólo Dios lo sabe!
//
// Así que si está tratando de 'optimizarlo'
// y fracasa (seguramente), por favor,
// incremente el contador a continuación
// como una advertencia para su siguiente colega:
//
// total_horas_perdidas_aquí = 189
```

4. **Aprende a testear tu código:** aprender a crear test para probar tu código es muy buena práctica ya que siempre estarás al tanto de qué cosas pueden llegar a salir mal o que pueda causar un error y así poder arreglarlo antes de que pase a producción.

5. **Sigue capacitandote:** Aprender nuevos Frameworks, nuevos lenguajes, técnicas de testeo, leer código de otras personas, interpretar tus errores y los de los demás, utilizar apps de diseño y de modelado. Toda herramienta que agregues a tu catálogo te ayudará ver un abanico más amplio de soluciones y a elegir la que mejor se adapte a tu proyecto



6. **Reutiliza el código:** Puede ser que pases mucho tiempo diseñando una funcion o una clase, perfeccionandola y haciéndola más eficiente, por lo que no tiene sentido y seria una perdida de tiempo usarlo solo una vez.

Sobre la estandarización:

En el acuerdo de convivencia se incluyo algunas pautas a la hora de codificar pero esas son solo las que en ocasiones anteriores han llevado a que queramos arrancarnos los ojos los unos a los otros, Por lo que se entiende que cumplirlas es esencial para la convivencia.

A continuación haremos mención completa a las pautas que se usarán en este equipo de trabajo. Las mismas tratarán de ser cumplidas, acto en el cual fallaremos estrepitosamente por lo que se designará a un encargado de verificar y correjir:

1. General:
    a. Los nombres no contienen “ñ” ni tildes
    b. Los bloques irán tabulados para que todo su contenido esté un nivel más allá de la línea introductoria y de la línea de cierre. Ej:

```
    <Col md="8">
        <Row md="2">
            <FormItem name="Nombre" />
            <FormItem name="Apellido" />
        </Row>
        <Row>
            <FormItem name="Direccion" />
        </Row>
    </Col>
```

    Esto se aplica a bloques html, React,{},[],(), etc

    c. Es preferible que los bloques no ocupen más de una pantalla, es decir 49 líneas de código. incluidas las funciones
    d. Las sentencias largas usarán múltiples líneas como si fueran un bloque. Pero su cierre “;” ira en la línea final y no debajo
2. Las carpetas:

    Se distribuyó los archivos en carpetas para que se pueda encontrar rápidamente lo que se está buscando.

    a. [Root]: la carpeta principal no incluirá ningún archivo de código, en ella solo se incluye los archivos de proyecto: “gitignore”, los “package” de npm y el Readme
    b. doc: la carpeta doc incluye la documentación de diseños y modelos
    c. public: los archivos web estáticos compartido públicamente por el cliente React
    d. src: incluye todo el código propiamente dicho. Suelto en ella estarán los archivos del cliente React. y en subcarpetas:
        d.a. Views: incluye todos las vistas de la interfaz del cliente
        d.b. Components: partes prearmadas de Views
        d.c. style: los css del cliente
        d.d. Server: todo el código del servidor express que funcionara como api

        d.d.a. routes: las interfaces propias de la api. Entiéndase que estas son interfaces entre el cliente y el servidor.
        d.d.b. views: En caso de incluir alguna interfaz para el usuario, la misma iría acá.
3. Los archivos:
    a. Cada archivo representa una clase y sus funciones asociadas. Dicha clase es la que se exportará. En caso de que se desee exportar alguna función o subclase estas solo estarán en el mismo archivo si estás sumamente relacionadas con la clase principal, caso contrario tendrán su propio archivo de pequeño submódulo con algunas funciones relacionadas
    b. Cuando el archivo es de una clase su nombre será el de la clase mientras que la misma se exportará por defecto.
    c. Los archivos de vistas empiezan con la palabra View
    d. Los archivos de clases y vistas empiezan cada palabra con mayúscula. los submódulos con minúscula.
4. Las Clases:
    a. Los nombres de clases empiezan cada palabra con mayúscula.
    b. Las vistas y componentes de las mismas no se harán con clases, sino por medio de funciones. Aun así sus nombres serán los de una clase. La razón de esto es que estamos usando Hooks. De ser necesario crear alguna como clase, se debe verificar que no afecte al funcionamiento, (no se podrá usar ningún componente que requiera Hooks)
5. Las Funciones:
    a. Los nombres de las funciones empiezan en minuscula la primera palabra y empieza con mayúscula las demás. ej: estoEsUnaFuncion();
    b. Toda función propiamente dicha se define como una constante lambda, es decir: const nombreFuncion = () =>{ }
    c. Las llaves se usarán sólo en caso de incluir sentencias fuera del return
6. Las constantes
    a. Los valores constantes se escriben en mayúscula: const PORT = 3000
7. Las variables