Development GuideLines Weepec

Tecnologias

- Frontend
  - Reactjs
  - ReactNative
  - Nodejs
  - Vuejs
  - MongoDB
- Server
  - Docker
  - Docker-Compose
- Backend
  - Prisma
  - Graphql
  - Loopback4
- Storage
  - Redis
  - Redux
  - MongoDB

Creación de repositorio

Se debe tener una notaciòn en la cual se denote 3 partes

- Proyecto al cual se pertenece (weepec)
- Ambito
- tecnologia principal

```
[Proyecto]-[Ambito]-[Tecnologia]
```

el cual quedaria como

```
weepec-app-reactnative
```

Manejo de ramas

Este debe ser con el flujo de trabajo git-flow, se deja una pequeña referencia aquí

Commits Messages

Motivación:

Al estar en un proyecto personal es normal usar o no usar métodos de trabajo ya que "nosotros nos entendemos" ¿pero que pasa al trabajar en equipo?

Necesitamos establecer reglar y una guia de desarrollo para que para todos sea compremsible y no encontrar cosas que sean raras y no sean entendibles para los demas, y en un futuro poder leer lo que alguien más hizo sin ninguna complicación.

Tanto commit messages, como revisión de codigo nos ayuda a hacer mas entendible, para un desarrollo y colaboración más entendible.

```
Todo en minusculas, con menos de 50 caracteres

En el body se debe de incluir una breve explicación sobre lo que se
resuelve

Los cambios que se efectuan se deben de listar de la siguiente manera
agregando Note:

Note:
- Se listan a continuación los cambios realizados
- Segundo cambio realizado

Al usar Sentry especifica el codigo que provee

Fix: #WEEPEC-K1
```

1. Especificar tipo de commit
    - `feat`: Una nueva caracteristica que se agrega al proyecto (feature)
    - `fix`: Un correccion de error (Bug Fix)
    - `style`: alguna caracteristica o correccioón respecto a estilos
    - `refactor`: Refactoring a specific section of the codebase
    - `test`: Todo lo relacionado con pruebas (Testing)
    - `docs`: Todo lo relacionado con documentación
    - `chore`: Mantenimiento de código de alguna sección o codigo base, [se puede usar emojis 😁]
    - utilizar el main topic enseguida del tipo `Tipo(Topic)` este define que sección cambio y ayuda a comprender más el tipo de commit.

El simbolo "!" debe de ser usado para denotar breaking change (Cambios fuertes) el cual corresponde a una versión superior la cual ya no funciona con una versión anterior.

```
fix!(Pets): fix plates
```

2. separar subject y body con un salto de linea

```
eg.
```

```
fix(Topic): add validation

Body
```

3. el commit message no debe de contener espacios en blanco demás
4. quitar signos de puntación innecesarios
5. no finalizar el subject con un periodo
6. todo en minusculas cada uno de los parrafos y subject
7. Use the imperative mood in the subject line
8. Usar el body del commit para explicar que cambios se realizaron
9. No asumir que quien revisa sabe todo el contexto, hay que incluir los detalles del problema
10. No pensar que el commit explica todo lo que se realizo
11. Seguir la estructura de commit definida por el equipo

Revisión de código

Al realizar pull request se debe de asiganr a un reviewer quien analizara y determinara si el pull request continua, en dado caso que el merge a master tenga inconvientes se revertira el merge

Puntos a considerar

- Evitar código comentado en cualquier punto
- uso de `console.log` debe de ser unicamente en desarrollo y cierto punto de debuging
- no usar segmentación de código y repetir código
- uso de variables fuera de scope o sin utilizar
- no se permite el uso de `var`
- para `ReactNative` y `ReactJs` el crear componentes es obligatorio
- el realizar push directamente a master esta prohibido
- la manera en realizar un release a master es con un pull request