# Working methodology

## Definition of done

Històries d’usuari:

* Funcionalitats implementades
* Complir els criteris d’acceptació de la Historia d’usuari
* Codi se
* n se errors
* Tests unitaris creats i superats
* Funciona en tots els dispositius dels membres del grup
* Bugs resolts
* Refactor del codi
* Documentar canvis importants
* Fer entregables

Sprint:

* S’ha complert el DoD de cada Història d’usuari inclosa al sprint
* “to do’s” complets
* Totes les proves unitàries superades
* Product backlog actualitzat
* Tests es dispositius i navegadors contemplats a la documentació superats
* S’han superat els test de compatibilitat amb versions anteriors
* Proves de rendiments superades
* Tots els bugs solucionats

## Conventions

### Programar

Programarem en Java.

Tot el codi que fem haurà d’estar en ***anglès***.

Utilitzarem camelCase per escriure les variables (Exemple: var inLab)

Comentar parts del codi crítiques o complexes

### Documents

Per garantir fer els mínims errors d’ortografia utilitzarem l’extensió de Chrome ***Language Tool****.* S’haurà de descarregar i utilitzar en qualsevol text que escrivim abans d’entregar. Tots els textos que fem seran en Català.

La tipografia dels Documents serà ***PT Serif Regular*** en tots els textos. A continuació veureu els diferents títols amb el seus formats corresponents.

Título

# Título 1

## Título 2

### Título 3

#### Título 4

Texto normal

En cas de voler destacar una paraula o frase concreta del text, utilitzem negreta i cursiva per destacar la ***paraula***.

A la capçalera dels nostres documents a la dreta hem de col·locar el logo de PetWorld que és el següent.



### Altres

Actualitzar Excel cada cop que es faci alguna cosa del projecte afegint nom, titol, data, hores invertides i si cal comentari.

Canviar el Scrum Master cada Iteració

Canviar el Product Owner cada iteració

Actualitzar el Trello cada setmana per documents i altres tasques

Cada setmana Dimarts de 10 a 12 i Dijous de 12 a 13 reunions de l’equip. Es planejarà amb antelació al Trello la quedada i es comentarà els temes a discutir per reunió.

## Working environment

Gestió de Documents: Google Drive

Gestió de Tasques: Trello

Gestió de Metodología Àgil: Taiga

repositori i Issue Tracker: GitHub

Base de dades i gestió d’usuaris: Firebase

Programar app: Android Studio

Eines per comunicarse: Whatsapp, Discord i Slack

## Document

### 1. Sprint Planning Meeting

#### 1.1. Acordar objectiu del Sprint

Quin increment patirà el producte? No és un compromís estricte amb certa flexibilitat

#### 1.2. Reordenar i repuntuar històries del product backlog

Cal reordenar la llista d’històries perquè es reflecteixi quines tenen més prioritats que d’altres.

Paral·lelament és bo comprovar si les puntuacions de les històries són encertades o s’haurien de modificar, partint de l’experiència d’anteriors iteracions a l’hora d’estimar el temps necessari.

#### 1.3. Afegir històries del product BackLog al sprint.

Un cop definit objectiu i actualitzat el product BackLog caldrà incloure al nou sprint les històries que ajudin a assolir l'objectiu de la iteració. Les històries d’usuari que defineixen requisits no funcionals hauran de tenir un tractament especial, formaran part de tots els sprints, per fer-ho les puntuem amb 0 punts i no extreure'n tasques, així no afectaran el volum de feina i estran en tot moment presents al taulell del sprint.

#### 1.4. Revisar criteris d’acceptació de les històries d’usuari

Revisar els criteris d’acceptació que siguin molt clars en cada història

#### 1.5 Dividir històries en tasques

Dividir històries en tasques per fer un membre de l’equip en un temps relativament curt, que permetrà veure dependències i traçar millor forma d’implementar.

### 2 Iniciar Història d’Usuari

#### 2.1 Crear Nova Branca al git

Dedicar nova branca per la història d’usuari.

### 3. Implementar una Tasca

#### 3.1. Assignar Tasca a desenvolupador

Cada membre s’haurà d’assignar a si mateix les tasques corresponents de la columna New i arrossegar-la a la Columna in progres.

#### 3.2. Programar la funcionalitat

Fer commits de manera freqüent a la branca. Els missatges de commit han de tenir:

Separar Subject i Body amb línia en blanc.

Començar el Subject amb majúscula i en sempre en imperatiu

Explicar que s’ha canviat i per què al body, no com.

Revisar l’estat del SonarCube a mesura que anem fent commits

#### 3.3. Fer Tests

Un cop acabada la funcionalitat passar la tasca a Ready for Test i començar a implementar el test amb jUnit. Els tests són en funció dels criteris d’acceptació.

#### 3.4 Finalitzar la Tasca

Un cop acabat de testejar, comprovar que passa Definition of Done, i en cas afirmatiu passar-ho a la columna de Closed. Cal incloure les hores dedicades a la tasca, per comprovar si els punts assignats han estat correctes.

### 4. Descobrir Bug

#### 4.1. Afegir Bug a Issue Tracker (Taiga)

Quan es detecti un error s’haurà de crear una nova Issue, amb nom, tipus, descripció i prioritat. Es pot afegir captures al trobar errors, descripció molt clara.

#### 4.2. Convertir Issue en Història i Afegir-la al BackLog

Si un Issue sembla que comporta un gran volum de feina caldrà incorporar-lo com una nova Història d’Usuari. Caldrà puntuar-lo i afegir Criteris d’acceptació Convenients.

#### 4.3. Assignar Nova Història a un Sprint

Si el Product Owner i equip estan d’acord, es pot deixar la història del Issue per un nou Sprint.

#### 4.4. Corregir Bug a branca pertinent

Si l'error es detecta aviat, s’haurà de resoldre amb un commit a la mateixa branca. En cas contrari, s’haurà de fer una nova branca a partir del develop si l'error ve d’aquest sprint o del màster si ve d’un sprint anterior.

#### 4.5. Tancar Tasca i Issue

Deixar coma Closed al tauler de tasques i coma Closed a l’Issue Tracker.

#### 4.6. Fer Merge de la nova branca

En cas de fer nova branca a partir de develop, simplement fem merge amb develop. Si la branca venia de master farem merge amb develop i amb màster.

### 5. Finalitzar Història d’usuari

#### 5.1. Revisar criteris d’acceptació

Comprovar que tots el test funcionen correctament i compleixen totes les situacions dels criteris d’acceptació. Si no és el cas, crear noves tasques fins a aconseguir-ho

#### 5.2. Fer Merge amb Develop

Fer Merge amb la branca on s’ha implementat una nova funcionalitat, cal tenir cura dels conflictes per no eliminar fragments de codi necessaris.

### 6. Tancar Sprint

#### 6.1. Tancar Sprint a Taiga

Taiga tanca automàticament els sprints un cop acabat el període assignat i tancades totes les tasques. Cal assegurar-se que totes les tasques implementades ho estiguin. Les que estiguin a Close cal retornar-les al Product Backlog

#### 6.2. Fer merge amb master

Incorporar tots els canvis a la branca màster. Des de master fer merge amb develop

### 7. Sprint review meeting

#### 7.1. Analitzar feina feta

Analitzar que s’ha quedat acabat i que pendent. Disposem de gràfiques:

Iteration burndown: Progrés en l'àmbit de punts durant el període

Release burndown: Progrés en l'àmbit de producte, estat total de punts finalitzats respecte a els totals al BackLog

Velocity Chart: Comparar quantitat de punts fets al sprint en comparació amb anterior. Hauria d’anar creixent.

#### 7.2. Mostrar el producte al Product Owner

Mostrar tots els increments que hi ha hagut al producte respecte a l’anterior, provant les noves funcionalitats i recollint feedback.

#### 7.3. Refinar Product Backlog

Mirar s’hi requereix modificació. El més clàssic és situar les històries que han quedat mitges a la part superior de la següent iteració.

### 8. Sprint Retrospective meeting

#### 8.1. Reflexionar sobre el procés de desenvolupament

Reflexionar diferents situacions i conflictes durant el sprint. Ex:

Comunicació entre membres de l'equip

Planificació i repartiment

Coordinació de Frontend i Backend

Seguiment metodologia de treball

Qualitat software lliurat

#### 8.2. Identificar el que cal millorar, mantenir o evitar

Plasmar per escrit conclusions de reunions. Classificar entre evitar, millorar i conservar i el perquè.