1. Iteration:

Beskrivelse:

Spilleren bliver budt velkommen og bedt om at vælge mellem en eksisterende hero eller at lave en ny. Dette valg tages i terminalen som user_input.

Hvis man vælger en eksisterende hero, bliver alle heroes hentet fra databasen og vist i en list, hvorefter spilleren bliver bedt om at skrive heros navn som input. Der tjekkes om den skrevet hero eksister derudover tjekkes der om der er nogle heroes i databasen overhovedet. Hvis ikke bliver spilleren bedt om at lave en ny hero.

Efter valget bliver informationen fra databasen hentet og sat som de private variabler af en instans af klassen Hero.

Hvis man vælger at lave en ny hero, bedes spilleren om at skrive heros navn, hvorefter bliver den nye hero oprettes i databasen, og der laves igen en instans af denne hero.

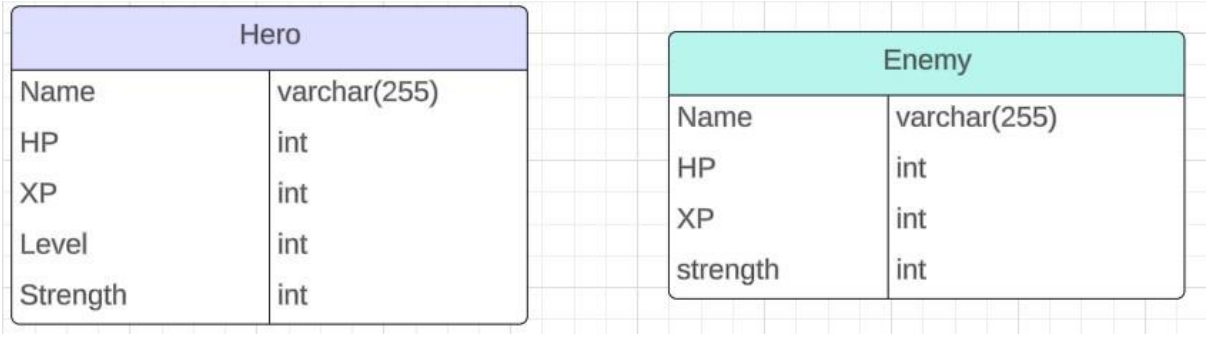
Der bliver vist en liste af alle enemies og deres tilhørende værdier. I terminalen vælges en fjende og kampen går i gang. Hver gang spileren trykker for at slå, skader hero og fjende hinanden med deres skadepoint. Når hp for enten hero eller enemy rammer 0 eller under stopper kampen.

Hvis hero vinder får de xp og dens hp bliver sat til 10 igen, hvis den er under efter kampen slutter. Taber heroen derimod, bliver hero slettet fra databasen.

Der blev spurgt hele tiden om man vil fortsætte med at spille eller ej.

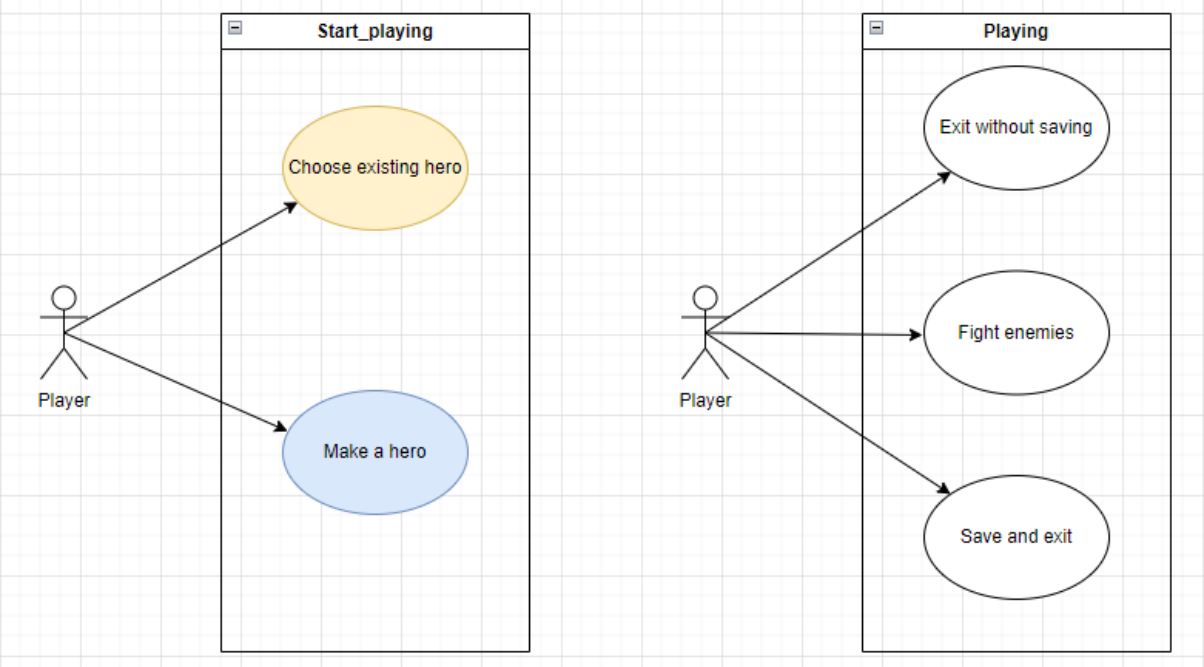
Når heroens hp rammer 1000*heroens level så stiger heroen i level.

ER-Diagram:



ER diagrammet viser tabellerne i databasen og de relationer mellem hinanden. Her kan man godt se at der er ingen relationer mellem de to tabeller.

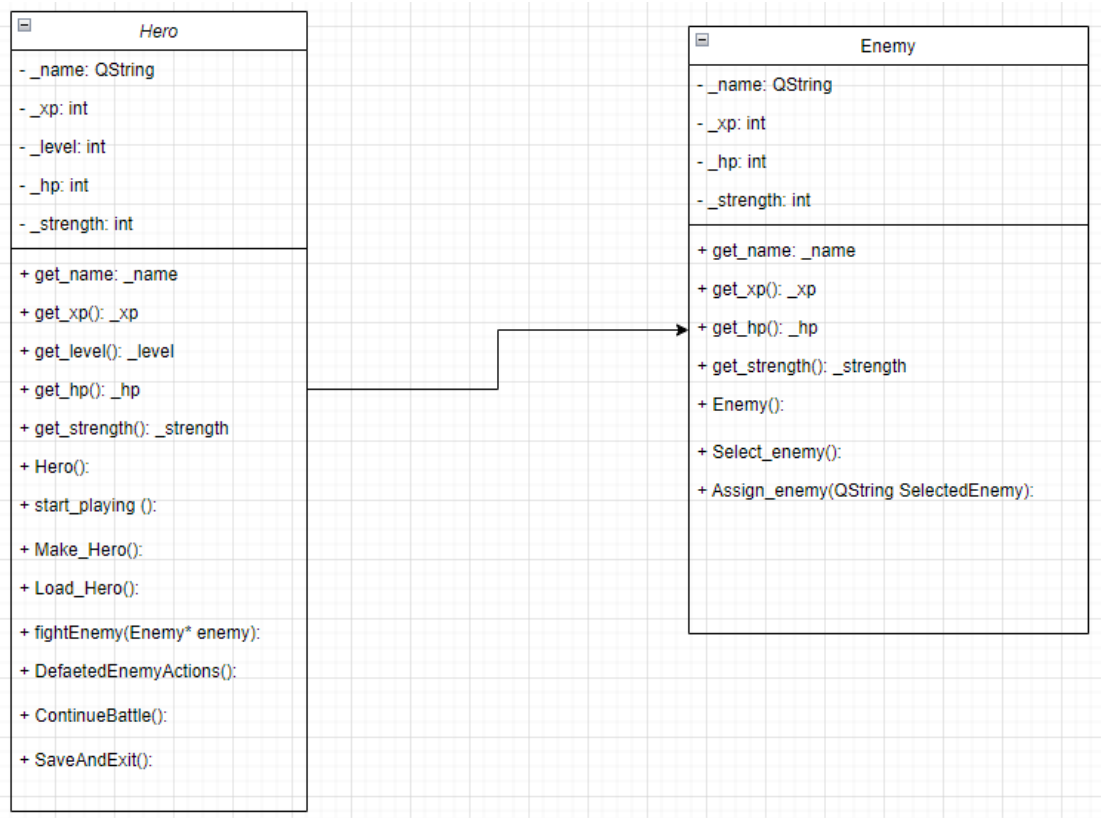
UseCase diagram:



Diagrammet viser at ved starting af spillet. Så har spilleren to muligheder, make hero eller load en.

Efter første valg, så kan spilleren vælger enten at går ud uden at gemme, spille eller gemme og gå ud af spilet.

Klasse diagram:



Uddybbning af metoder:

Hero_class:

- Getter metoder giver adgang til de private variabler
- Start_playing: Byder spilleren velkommen og giver muligheder, hvis der skaloprettes en hero eller load en.
- Make hero: Laver en ny hero med initialiseret værdier.
- Load hero: Henter hero liste fra databasen og spørger spilleren om at vælger en, hvorefter heros data bliver hentet fra databasen
- fightEnemy: Tager enemy's og heros info og simulerer spillet.
- DefaetedEnemyActions: Når enemy bliver besejret så gives der to muligheder. Enten at fight igen eller gem og går ud af spillet.
- ContinueBattle: Under spillet og hver eneste gang heroen slår, gives der muligheden for enten at fortsætte med at kæmpe i mod den valgte enemy. Eller går ud af spillet.
- SaveAndExit: Metoden opdaterer heros data i databasen og går ud af spillet.

Metoderne, som simulerer spillet, er i hero-class, fordi der tænkes at hero er master over alt. Og det er næsten kun hero data, der skal opdateres hele tiden, derfor har de været en del af hero class.

Enemy_class:

- Getter metoder giver adgang til de private variabler
- Select_enemy: Lister enemies, så spilleren kan vælger, hvad han vil kæmpe.
- Assign_enemy: Efter at spilleren har valgt enemy'en så hentes data fra database, for at spillet kunne går i gang.

2. Iteration:

Beskrivelse:

Spilleren bliver budt velkommen og bedt om at vælge mellem en eksisterende hero eller at lave en ny. Dette valg tages i terminalen som user_input.

Hvis man vælger en eksisterende hero, bliver alle heroes hentet fra databasen og vist i en list, hvorefter spilleren bliver bedt om at skrive heros navn som input. Der tjekkes om den skrevet hero eksister derudover tjekkes der om der er nogle heroes i databasen overhovedet. Hvis ikke bliver spilleren bedt om at lave en ny hero.

Efter valget bliver informationen fra databasen hentet og sat som de private variabler af en instans af klassen Hero.

Hvis man vælger at lave en ny hero, bedes spilleren om at skrive heros navn, hvorefter bliver den nye hero oprettes i databasen, og der laves igen en instans af denne hero.

Spilleren for muligheden for at vælge mellem at kæmpe imod enemy eller går ind i en grotte.

Hvis spilleren vælger at kæmpe imod en enemy, så bliver der vist en liste af alle enemies og deres tilhørende værdier. I terminalen vælges en fjende og kampen går i gang. Hver gang spileren trykker for at slå, skader hero og fjende hinanden med deres skadepoint. Når hp for enten hero eller enemy rammer 0 eller under stopper kampen.

Hvis hero vinder får de xp og dens hp bliver sat til 10 igen, hvis den er under efter kampen slutter. Taber heroen derimod, bliver hero slettet fra databasen.

Der blev spurgt hele tiden om man vil fortsætte med at spille eller ej.

Når heroens hp rammer 1000*heroens level så stiger heroen i level.

Vælger spilleren til gengæld at går ind i en grotte, så bliver der vist en liste af alle grotter og deres tilhørende værdier og hvad der skal kæmpes for at besejre grotten.

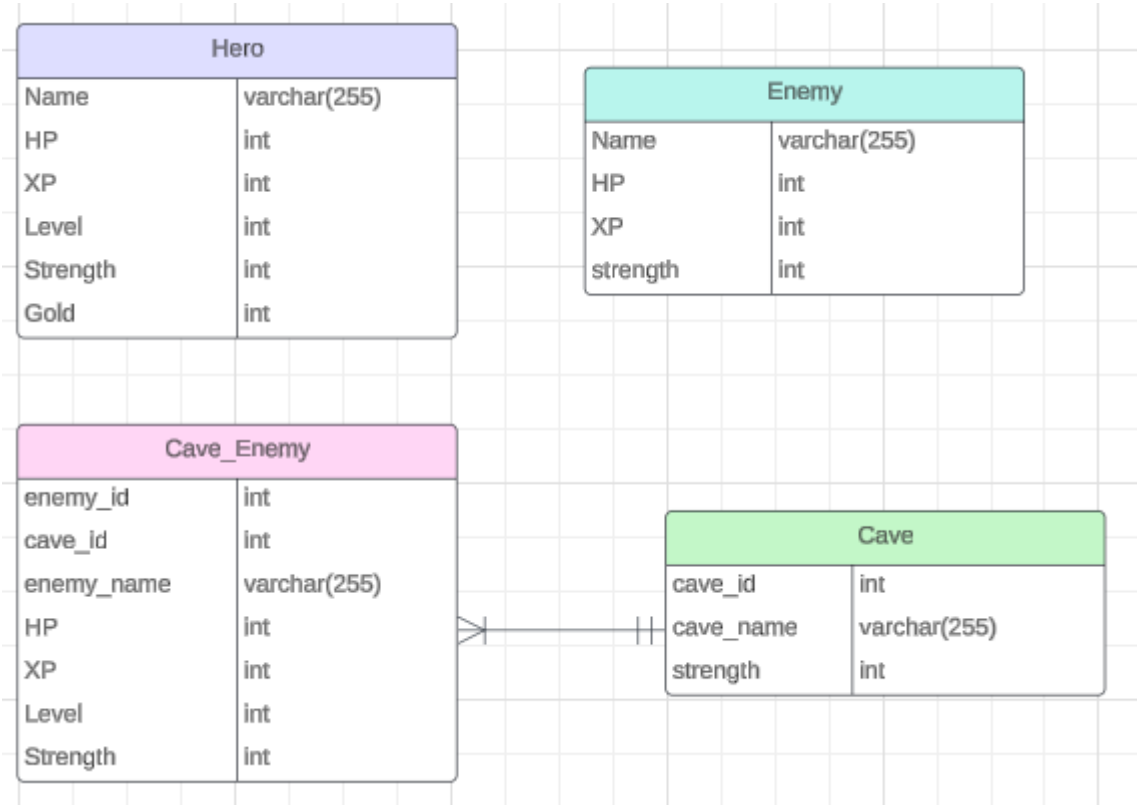
Hver gang spileren trykker for at slå, skader hero og fjende hinanden med deres skadepoint. Når hp for enten hero eller enemy rammer 0 eller under stopper kampen.

Hvis hero besejrer den første enemy så skal heroen fortsætter med at kæmper i mod det andre enemies som eksister i grotten.

Hvis hero vinder får den xp, guld og dens hp bliver sat til 10 igen, hvis den er under efter kampen slutter. Databasen bliver opdateret med de nyeste data.
Taber heroen derimod, bliver hero slettet fra databasen.

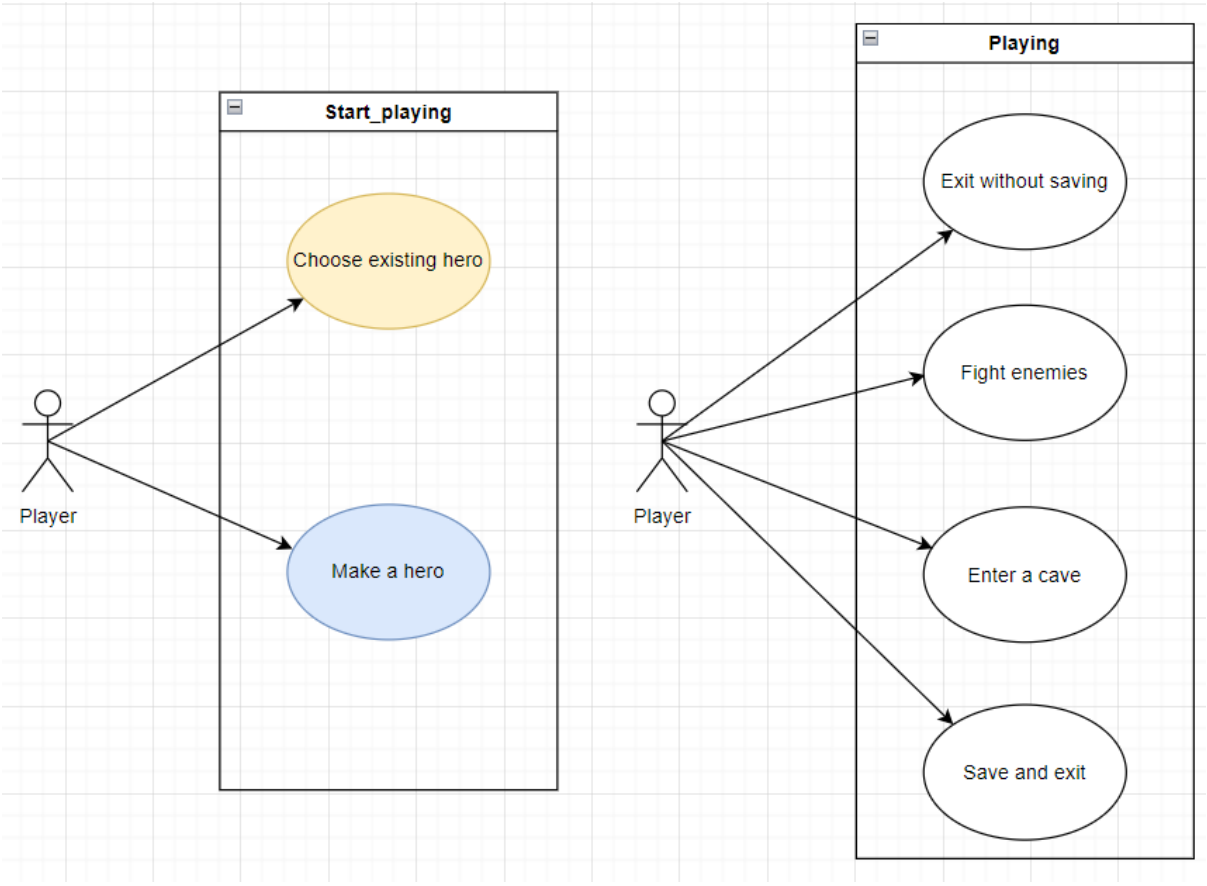
Der blev spurgt hele tiden om man vil fortsætte med at spille eller ej.

ER-Diagram:



ER diagrammet viser relationen mellem de forskellige tabeller i databasen, hvor der kan ses at ”Cave_Enemy” bruger ”Cave” ved at have (cave_id) i sig. Herudover kan man se at der er ingen relationer mellem hero og enemy tabeller.

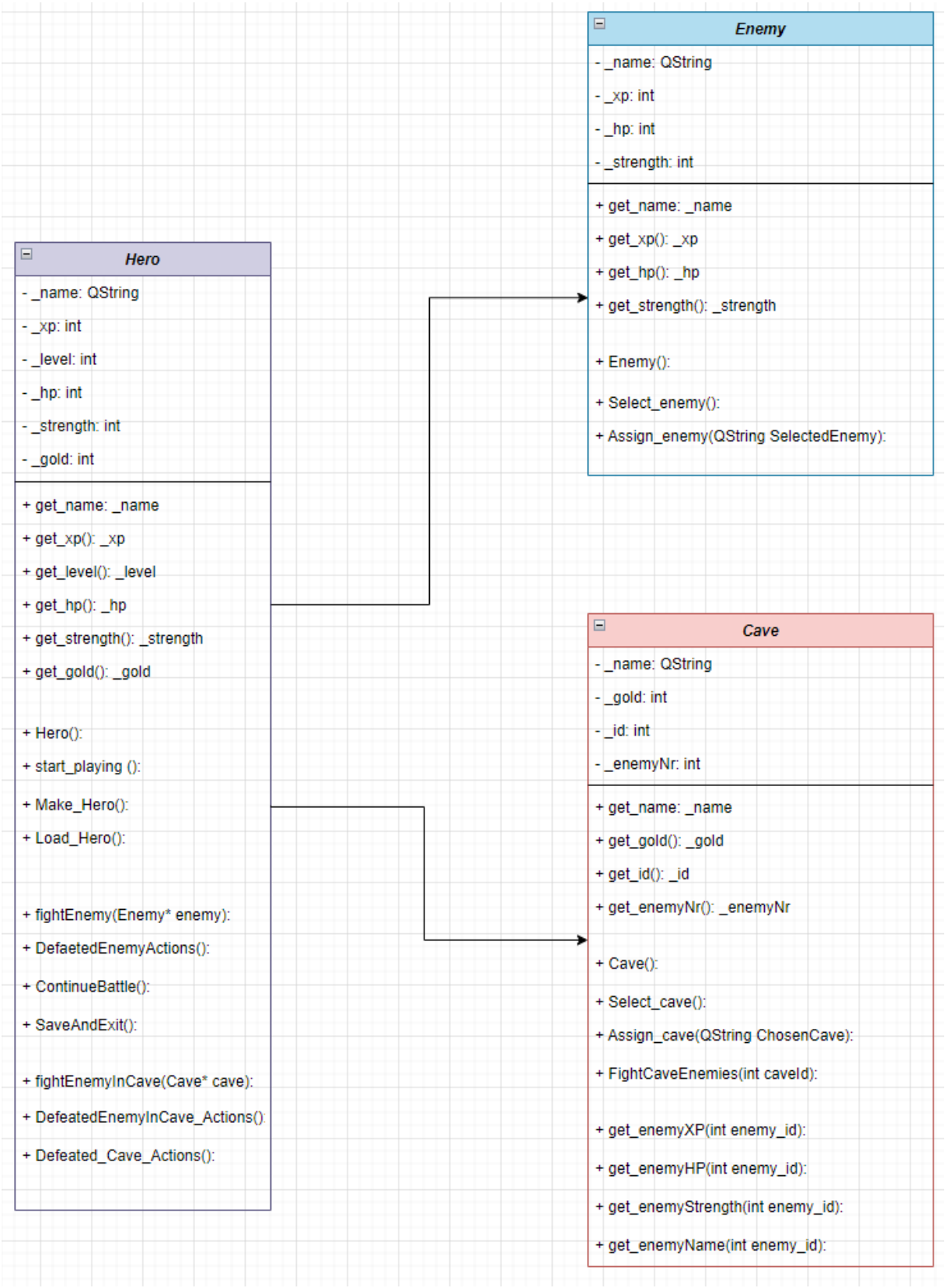
UseCase diagram:



Diagrammet viser at ved starting af spillet. Så har spilleren to muligheder, make hero eller load en.

Efter første valg, så kan spilleren vælger enten at går ud uden at gemme, kæmpe med en enemy, går ind i en grotte eller gemme og gå ud af spilet.

Klasse diagram:



Diagrammet viser relationer mellem de forskellige klasser i koden. Her kan der ses, at der er hero-klasse har noget med enemy- og cave-klasse men ikke afhængig af disse.

Uddybbning af metoder:

- De samme metoder som 1. iteration
- Følgen metoder blev tilføjet:
    - fightEnemyInCave: Tager enemy's i grotten og heros info og simulerer spillet.
    - DefeatedEnemyInCave_Actions: Efter enemy i grotten bliver besejret, får spilleren mulighed for at enten fortsætte med at kæmpe i mod den næste enemy i grotten eller går ud af spillet.
    - Defeated_Cave_Actions: Efter spilleren har besejret alle enemy i grotten, så får spilleren mulighed for enten at fortsætte med spillet eller gemme og gå ud af spillet.

Metoderne, som simulerer spillet, er i hero-class, fordi der tænkes at hero er master over alt. Og det er næsten kun hero data, der skal opdateres hele tiden, derfor har de været en del af hero class.

I dette punkt kunne man godt se, at det vil være smartere, hvis man har en klasse for spillet. Hvor hele spillet og simulationer sker, og hero-klassen er for sig selv.

Enemy_class:

- De samme metoder som 1. iteration

Cave_class:

- Getter metoder giver adgang til de private variabler
- FightCaveEnemies: Byder spilleren velkommen og giver muligheder, hvis der skaloprettes en hero eller load en.
- Select_cave: Lister alle grotter, så spilleren kan vælger, hvilken skulle kæmpes.
- Assign_cave: Efter spilleren har valgt en grotte så hentes data fra database, for at spillet kunne går i gang.
- Getter metoder for enemy i cave: Henter de nødvendige data om enemies fra databasen.

3. Iteration:

Programmet starter med at bede brugen om at læse de nødvendige dokumentationer for at køre spillet. Derudover beder programmet brugen om at skrive ”username” og ”password” for den lokal database, som brugeren anvender.

Skriver spilleren de nødvendige informationer korrekt, bliver spilleren budt velkommen og bedt om at vælge mellem en eksisterende hero eller at lave en ny. Dette valg tages i terminalen som user_input.

Hvis man vælger en eksisterende hero, bliver alle heroes hentet fra databasen og vist i en list, hvorefter spilleren bliver bedt om at skrive heros navn som input. Der tjekkes om den skrevet hero eksister derudover tjekkes der om der er nogle heroes i databasen overhovedet. Hvis ikke bliver spilleren bedt om at lave en ny hero.

Efter valget bliver informationen fra databasen hentet og sat som de private variabler af en instans af klassen Hero.

Hvis man vælger at lave en ny hero, bedes spilleren om at skrive heros navn, hvorefter bliver den nye hero oprettes i databasen, og der laves igen en instans af denne hero.

Spilleren for muligheden for at vælge mellem at kæmpe imod enemy eller går ind i en grotte.

Før heroen går i gang med dette sendes heroen til en magic_shop, hvor den har mulighed for at købe nogen magi. Magi bliver listet i en liste med deres nødvendige informationer hentet fra databasen. Heroen vælger i terminalen, hvilken magi, der skal købes, hvor der tjekkes om hero har nok guld eller ej. Efter købet gennemføres sendes spilleren til næste valg.

Hvis spilleren har valgt at kæmpe imod en enemy, så bliver der vist en liste af alle enemies og deres tilhørende værdier. I terminalen vælges en fjende og kampen går i gang. Hver gang spileren trykker for at slå, gives der en mulighed for enten at slå ved brug af magi eller slå uden magi. Alt efter valget skader hero og fjende hinanden med enten deres skadepoint samt magi_styrke og magi_selvstyrke eller bare deres skadepoint. Når hp for enten hero eller enemy rammer 0 eller under stopper kampen.

Hvis hero vinder får de xp og dens hp bliver sat til 10 igen, hvis den er under efter kampen slutter. Taber heroen derimod, bliver hero slettet fra databasen.

Der blev spurgt hele tiden om man vil fortsætte med at spille eller ej.

Når heroens hp rammer 1000*heroens level så stiger heroen i level samt i magic_level.

Vælger spilleren til gengæld at går ind i en grotte, så bliver der vist en liste af alle grotter og deres tilhørende værdier og hvad der skal kæmpes for at besejre grotten.

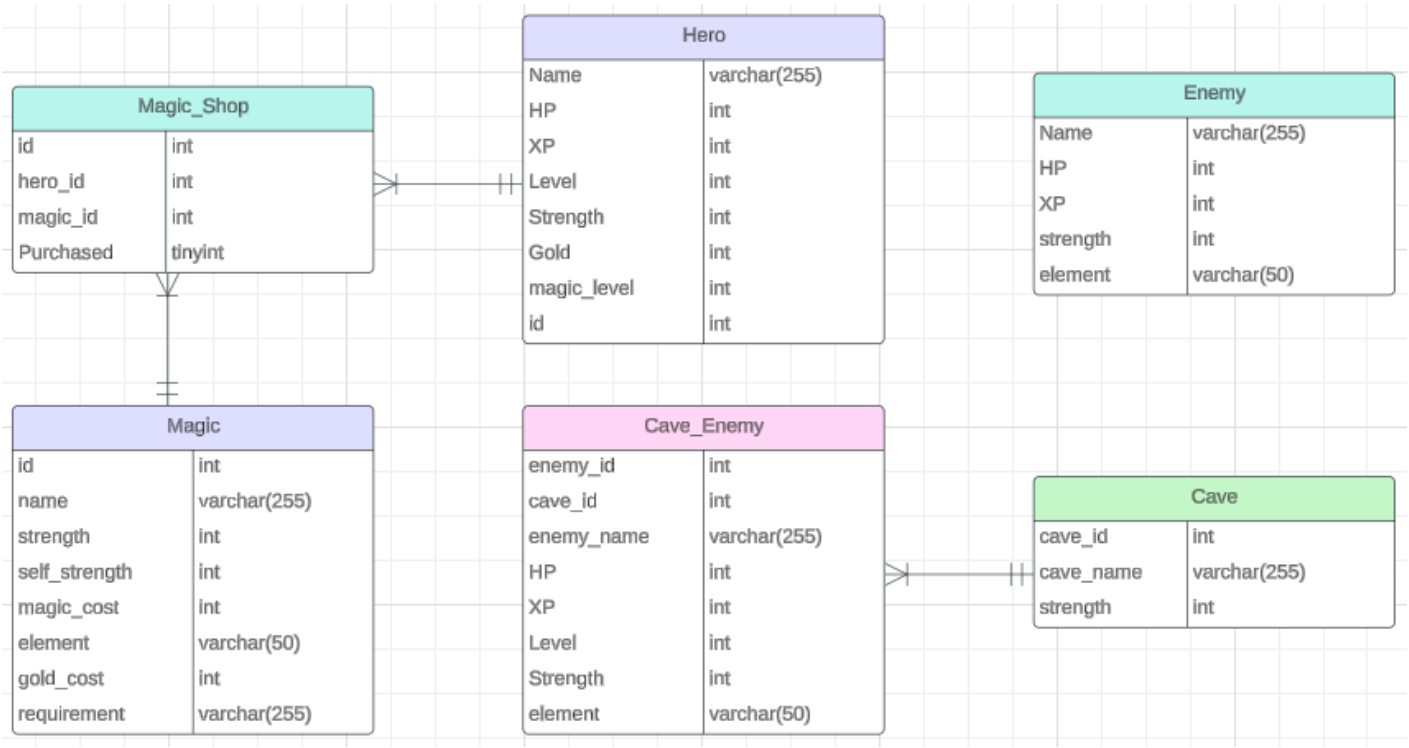
Ligeledes, hver gang spileren trykker for at slå, gives der en mulighed for enten at slå ved brug af magi eller slå uden magi. Alt efter valget skader hero og fjende hinanden med enten deres skadepoint samt magi_styrke og magi_selvstyrke eller bare deres skadepoint. Når hp for enten hero eller enemy rammer 0 eller under stopper kampen.

Hvis hero besejrer den første enemy så skal heroen fortsætter med at kæmper i mod det andre enemies som eksister i grotten.

Hvis hero vinder får den xp, guld og dens hp bliver sat til 10 igen, hvis den er under efter kampen slutter. Taber heroen derimod, bliver hero slettet fra databasen.

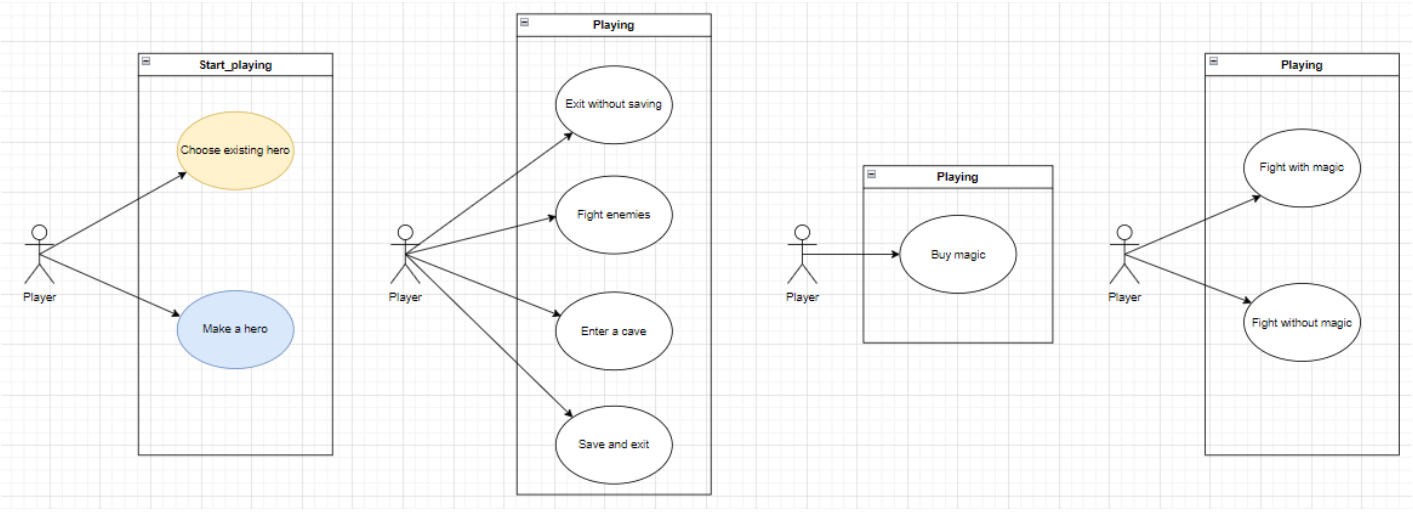
Der blev spurgt hele tiden om man vil fortsætte med at spille eller ej

ER-Diagram:



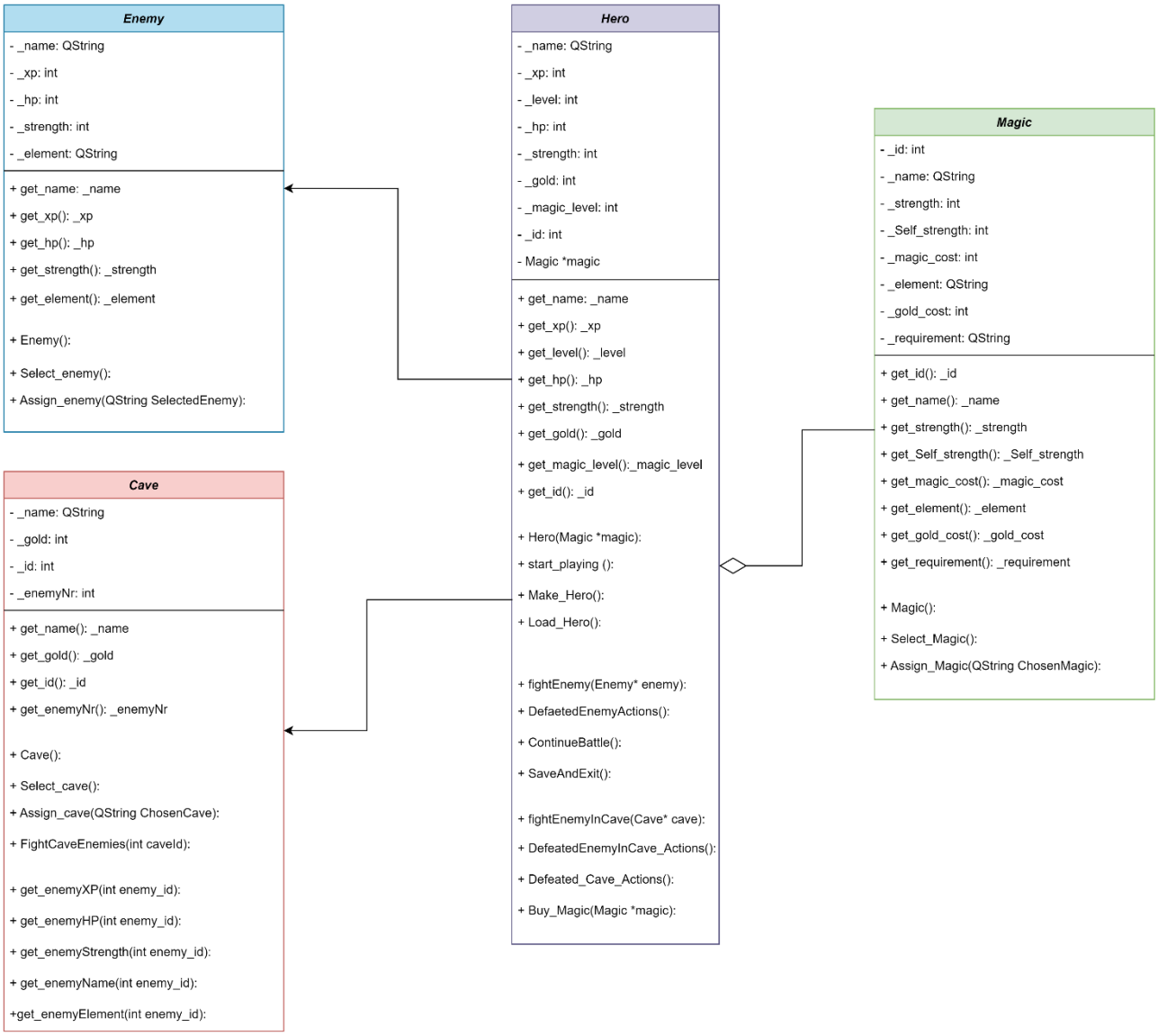
ER diagrammet viser relationen mellem de forskellige tabeller i databasen, hvor der kan ses at ”Cave_Enemy” bruger ”Cave” ved at have (cave_id) i sig. Herudover kan man se at der er ingen relationer mellem hero og enemy tabeller. Samt kan der ses at ”Magic_Shop” bruger ”Magic” og ”Hero” ved at have (magic_id OG hero_id) i sig.

UseCase diagram:



Diagrammet viser at ved starting af spillet. Så har spilleren to muligheder, make hero eller load en. Efter første valg, så kan spilleren vælger enten at går ud uden at gemme, kæmpe med en enemy, går ind i en grotte eller gemme og gå ud af spilet. Bagefter købes der magi. Den sidste valg er at spilleren vælger at bruge den købte magi eller ej.

Klasse diagram:



Diagrammet viser relationer mellem de forskellige klasser i koden. Her kan der ses, at der er hero-klasse har noget med enemy- og cave-klasse men ikke afhængig af disse. Derudover viser diagrammet, at hero klassen er opbygget af nogle elemeneter af Magic klasse ((f.eke. ved at konstruktoren har pointer til magi som input)). Men magic klasse kan eksistere uden hero klasse.

Uddybbning af metoder:

Hero_class:

- De samme metoder som 2. iteration
- Følgen metoder blev tilføjet:
    - Buy_Magic(Magic *magic): som gør det muligt at købe magi for heroen samt opdatere databasen med de nødvendige opdateringer.
- Følgen metoder blev ændret:

- fightEnemy(Enemy* enemy) OG fightEnemyInCave(Cave* cave): ændring er at metoderne har mulighed fr at burge magi under simulation af spillet.
- + Hero(Magic *magic): konstruktor har pointer til magi som input, hvilket gør det muligt at bruge magi undervejs i klassen.

Enemy_class :

- De samme metoder som 2. iteration.
- Følgen metoder blev tilføjet:
  - get_element(): som er getter metode til den nye private variabel element

Cave_class:

- De samme metoder som 2. iteration.
- Følgen metoder blev tilføjet:
  - get_enemyElement(int enemy_id): som er getter metode til den nye private variabel element.

P.S.:

Ideelt skulle være tjek for element og requirement for når der skal bruges eller købes magi. Men pga. udfordringer i implementation i c++ programmet, kører spillet uden.

Git- log:

Grundet problemer med at ”lave” git-repo. rigtigt er der ikke så mange commit i starten dvs. efter 1. iteration blev færdiggjort.

```
commit 3aae0c8d58ac485f09dc207c6451f93badbf657a (HEAD -> master, origin/master)
Author: Ayman <aymante@gmail.com>
Date:   Fri May 31 00:12:58 2024 +0200

    1. , 2. & 3. Iteration

commit 6e9b3958dc30932dd2d7e0b6436c9789fe670cbe
Author: Ayman <aymante@gmail.com>
Date:   Fri May 31 00:03:13 2024 +0200

    1. , 2. & 3. Iteration

commit 899231d903382894b8772d9b6318513f22def7cb
Author: Ayman <aymante@gmail.com>
Date:   Thu May 30 23:19:12 2024 +0200

    1. , 2. & 3. Iteration
```

```
commit 6f50a806d0d1b4baadacf4867beb43f55beee159
Author: Ayman <aymante@gmail.com>
Date:   Thu May 30 22:34:26 2024 +0200

    1. , 2. & 3. Iteration

commit dbeec13555425b0a8cc0073cb22a1e241a74ba3f
Author: Ayman <aymante@gmail.com>
Date:   Thu May 30 19:37:57 2024 +0200

    1. , 2. & 3. Iteration

commit 6dd3087fdf04a38216062d8016fa99400fea3ab4
Author: Ayman <aymante@gmail.com>
Date:   Thu May 30 14:38:48 2024 +0200

    1. , 2. & 3. Iteration

commit b372ad2dca25b89461cb0c0697abaeccb102cb56
Author: Ayman <aymante@gmail.com>
Date:   Wed May 29 01:00:52 2024 +0200

    1. , 2. & 3. Iteration
```

```
commit c7feeed9015a6e27efbacf78a83ae9347a181e6a
Author: Ayman <aymante@gmail.com>
Date:   Sun May 26 23:32:58 2024 +0200

    1. & 2. Iteration

commit 57eaa7162669e8bcecb418a5227704bddc61dc2b
Author: Ayman <aymante@gmail.com>
Date:   Sun May 26 23:30:07 2024 +0200

    1. & 2. Iteration

commit ff1ae140772ed263c20283a2b45cb6b5dd0f6153
Author: Ayman <aymante@gmail.com>
Date:   Sun May 26 23:16:41 2024 +0200

    1. Iteration

commit bdf232d6e7a766b981cf5e5a14b24280cbc3f73e
Author: Ayman <aymante@gmail.com>
Date:   Sun May 26 22:32:33 2024 +0200

    1. Iteration

commit e24c789c94a35b57d6949cf70e41d97fd2468d58
Author: Ayman <aymante@gmail.com>
Date:   Sun May 26 22:24:18 2024 +0200

    1. Iteration
```