* 1. *Wat hebben we goed gedaan?*
  2. *Wat ging minder goed?*
  3. *Wat zullen we in de volgende sprint (of na sprint 3, volgend project) anders doen?*

*Sprint 0 “backlog” retrospective*

De eerste sprint behandelde het invullen van de product backlog en het maken van de wireframes.

Bij het opstellen van de backlog voor onze applicatie, ondervonden we grote moeilijkheden in het correct inschatten van de werkuren. Het viel ons op dat we voor bepaalde items ofwel veel te hoog inschatten. Vooral het concretiseren van bepaalde te implementeren functionaliteiten bleek een moeilijke denkoefening. Soms hadden we een item te hoog ingeschat, maar in een latere fase (Sprint 1) konden wie die dan opsplitsen, waardoor er eigenlijk meer tijd nodig was dan vooraf bepaald.

Een algemene verdeling opstellen voor de verschillende toekomstige sprints verliep wel vlot. Doordat we ook een algemeen JAVA klassenschema opstelden in deze sprint, hoewel niet vereist, hielp dit ons om al een duidelijk onderscheid te maken tussen wat bij sprint één hoorde en wat eerder voor sprint twee en drie werd opgesteld.

Voor de creatie van de wireframes zaten we enkele keren samen aan de ronde tafel en gooiden we allerlei ideeën in de groep. We bereikten al snel een consensus over de vorm en uitgetekende functionaliteit van onze applicatie. De manier waarop we de wireframes opstelden en uitwerkten is voor ons zeker voor herhaling vatbaar.

*Sprint “front-end”retrospective 1.*

De tweede *sprint* behandelde de echte frontend van onze applicatie. Het verzorgen van de HTML en Javascript. Een complexe taak waarbij heel wat werkuren verricht dienden te worden. En waarvoor nog maar eens duidelijk werd dat het maken van scrum board items en ze correct kunnen inschatten pas goed zal beginnen werken met ervaring. We splitsen al meer te algemene items op.

Het leren werken met Git zorgde voor enorme vertraging: de eerste drie dagen werkten we zelfs niet met Git omdat het voor de front-end quasi geen meerwaarde opbracht. Elk groepslid werkte aan zijn eigen stukje van de HTML en Javascript. En aangezien de meeste functionaliteiten qua grootte niet vereisten er tegelijk aan te werken, bracht het gebruik van Git niet op. Pas in het tweede deel van de Sprint, waar de JavaScript tamelijk complex groeide, zagen we al snel de voordelen van Git in, en begonnen we er ook gebruik van te maken. Aan het einde van de sprint had iedereen Git goed onder de knie. We kozen om Git met de GUI te gebruiken, welbepaald met het programma GitKraken. Als *freeware*, voor niet-commercieel gebruik, een uitstekend Git programma. We verkiezen de GUI ten allen tijde boven de CLI Git omdat overzicht, wat zo belangrijk is voor de Git ervaring, ontbreekt in de CLI.

*Wat we meenemen voor de volgende sprint*

We begonnen deze sprint eigenlijk met elke bepaalde items op zich te nemen zonder veel overleg over de structuur van de front-end. Op het einde van de sprint vernamen we dat we in de HTML en javascript heel wat zaken zoals plaatsing van ID’s en klassen (die de semantische HTML elementen aanspreken) moesten veranderen. Het ontbrak ons, ondanks een degelijke backlog, om een gestructureerde planning op te stellen. Dit pakken we aan in de volgende sprints waarbij we het steeds het grotere geheel en de structuur van de backend in het oog zullen houden.

*Sprint 2 “backend part one” retrospective*

*JAVA is a dream: a nightmare of epic proportions* – Groep 7.

Voor sprint twee pakten we het radicaal anders aan: overleg en communicatie in de groep worden onze sleutelwoorden, *pair programming* en *mob programming* maken we eigen tot ons steeds groter wordend gemeenschappelijk lingo[[1]](#footnote-1). Het werk aan de backend, t.o.v. de front-end, ligt veel intensiever. Gekoppeld aan de stijgende examenkoorts en deadlines voor andere taken, krijgt dit hoofdstuk van het project een totaal andere dimensie. Een rustig kabbelend beekje vs. een kolkende en oorverdovende waterval. Er bestaan kortgezegd evenveel vergelijkingen voor de radicale tegenstelling backend vs. front-end als de angstaanjagende JAVA excepties.

Daarenboven hebben we de eerste dagen heel wat tijd gespendeerd aan de voor 90% afgewerkte front-end (hier kwamen de zgn. “*done but not finished”* casussen op). We werkten hier bewust in paren aan: één paar stortte zich op die laatste vervelende *itch cases[[2]](#footnote-2)*, terwijl het andere paar de beginselen van de JAVA front-end op zich nam.

Een lang uitgerekte sprint, verdeeld over verschillende losse dagen, zorgden occasioneel voor lange inwerkperiodes en verwarrende momenten. Als anekdote kan de dubbel geschreven test van **<insert name of member of group here>** als goed bewijs dienen van de soms absurde aard van deze sprint.

Maar we boekten vooruitgang! De magische connectie met de database, JAVALIN communicatie: samen met de *brothers in arms* Jackson en Maven leggen we langzaam maar zeker de mysterieuze JAVA codekronkels bloot. Voor ons allen toch, de één wat minder dan de andere, was wat zelfstudie vereist. JAVA connectie met SQL is voor de meesten van ons geen natuurlijk gegeven.

Gestructureerd werken was niet alleen aangewezen, maar simpelweg een must. Want het wordt nu wel echt gemakkelijk voor de niet in het *Javanees[[3]](#footnote-3)* onderrichte leek zich te verliezen in het doolhof der klassen dat uiteindelijk *Howeststone* dient te worden.

*Wat nemen we mee naar de volgende sprint*

Focus op de essentie: het onderscheiden wat echt belangrijk is en wat dus, met oog op het eindresultaat, niet mag ontbreken voor het Minimum Viable Product[[4]](#footnote-4).

*Sprint 3 “backend part two” retrospective*

In de derde sprint dienden we de frontend en de backend én de database aan elkaar te linken zodat de applicatie een gestroomlijnd karakter krijgt. We kregen de CLI goed aan de praat, vervolledigden de benodigde klassen voor een werkend basisspel en zorgden voor de juiste implementatie aan de frontend van de *abilities*.[[5]](#footnote-5)

Voor deze sprint zat de communicatie echt goed; we werkten in grote mate samen aan grote problemen in de code. De vooruitgang verliep gestaag en hadden goed zicht op ons beoogde einddoel. Spijtig genoeg probeerden we ook *spell cards* (kaarten met zgn. magische effecten) te implementeren. Afgewerkt in de SQL database en deels op poten gezet in JAVA konden we deze issue niet afwerken wegens tijdgebrek.

Wat we kunnen meenemen en beter doen in het volgende project valt op te maken aan de hand van een algemene eindbalans van dit project. Zeker de werkverdeling van de verschillende sprints kon beter. In de eerste sprint beseften we namelijk nog niet hoeveel meer werk de backend zou verwachten. Een strakkere planning voor bepaalde te implementeren ***feathers***, tijdig feedback geven en het grotere geheel in het oog houden zodat de scheiding tussen belangrijke en minder belangrijke issues voor iedereen duidelijk is.

1. ander woord zoeken / zin herschrijven [↑](#footnote-ref-1)
2. de normale gang van zaken ontwijkend; implementatie van een oplossing / methode die in 90 % van de gevallen werken zal, maar waar een kleine minderheid mee zal botsen. [↑](#footnote-ref-2)
3. Een fantastische verwijzing naar de JAVA programmeertaal, vrucht van lector M. de Wael [↑](#footnote-ref-3)
4. SCRUM terminologie, cfr cursus PM [↑](#footnote-ref-4)
5. Met de *abilities* worden de speciale aanvallen of verdedigingsmogelijkheden van de kaarten bedoeld: *windfury* is zo’n voorbeeld van een *ability* waarbij de kaart tweemaal in één beurt mag aanvallen i.p.v. de standaard eenmalige aanvalsmogelijkheid. [↑](#footnote-ref-5)