SLINGSHOT

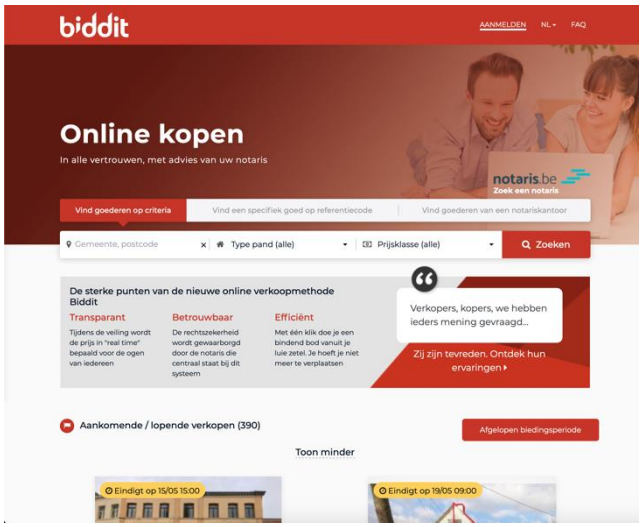
Reference case

Biddit

Today's public sales of houses is a complex procedure with long cycle times. The notary's fees and publicity costs make a public sale expensive for the seller.

In addition, the real value of the property is almost never achieved. Moreover, in the spring of 2018 the right of higher bid is abolished. Reasons enough to think about a more digital approach to look at this process.

FedNot, the Federal Association of Notary's, created a platform where online bids on public sales can take place. They want to create a transparent platform with a large offer on which the best price for the property can be obtained.



Slingshot has guided FedNot through the different phases of the project: from design over build to hosting and maintenance. During the build phase, we used Scrum with sprints of two weeks.

What did we do:

- Architecture and build the new bidding platform.
- Integration with existing FedNot back-ends.
- Integration with BOSA Authentication “ItsMe” & eID.
- Making the platform mobile friendly.

Tasks performed

Functional workshops and facilitation

For the facilitation we worked together with IMEC. In their workshops and analysis sessions, they emphasize the importance of always approaching the MVP deliverables from a lean perspective, pursuing short feedback loops, iterating quickly, avoiding waste and realizing ‘small product increments’.

Domain Driven Design

The functional domain can be divided into different functional business domains or bounded contexts. We currently distinguish between bounded contexts:

- Offer
- Auction (bidding, time management)
- Platform (registration and profile management of buyers, sellers and notary offices)
- Notifications and reporting
- Service provider for IAM-Development tasks.
- Infrastructure Operations and Support
- Infrastructure Service Provider
- Platform service providers

Each of these bounded contexts typically has its own so-called ubiquitous language, a context specific lexicon. Thinking in bounded contexts has the advantage of being ‘functionally isolated’ on the one hand. Of course, each part is an essential part of a whole thing. In addition, all parts are ‘loosely coupled’ with each other. This is an approach that can be translated into microservices architecture.

The domain driven design approach allows us to have multiple teams working in parallel on separate bounded contexts during the development process of this assignment. This is a successful strategy as it helps to meet the strict project deadlines.

Team composition

The choice for Domain Driven Design also reflected on the composition of the team. Consequently, both back-end and front-end development, UX/UI design, product ownership, release engineering, architectural guidance, testing, scrum mastership and project management are assured.

The trajectory consists of 18 sprints, each with a specific focus, arranged in a logical order, in which the functionality of the next sprint is analyzed during each sprint.

Before the recurring bi-weekly sprints, we started with the so-called kickstart sprint:

- The product owner collects the functional input for the first effective development sprint.
- The technical architect collects the necessary technical information to realize the interface with the existing notary platform.
- While the release engineer does the setup of the development cluster and continuous integration server, with build pipelines and initial deployment pipelines.
- The developers take care of the foundations of the first microservices with central login and monitoring facilities.

We realized an MVP in less than 6 months taking the feedback of user tests via customer panels into account.

Quality

Throughout the project we had a full-time tester at our disposal to continuously test the new functionalities. For each new functionality, the function analyst writes out a number of acceptance criteria. The end customer (FedNot) also performs a number of business test with each sprint validation. Halfway through the process, we set up the production environment. Once this was set up, we carried out a number of weekly performance tests. Around this period, we also started automatic testing. This allowed us to perform a number of regression tests with each new ‘build’. We also took care of pen tests (security tests). We called in an independent security service that was able to detect any security problem.