



Customer's mission is to provide an excellent shopping experience. The customer establishes a transparent and seamless interaction between buyers and sellers to create the best deal for both parties.

Price Comparison and Personal Offers platform

The Price Comparison and Personal Offers platform instantly provides an overview of all shops and their offers for a product that a customer is looking for. Personalized offers can be made just-in-time taking into account a customer's profile and market data.

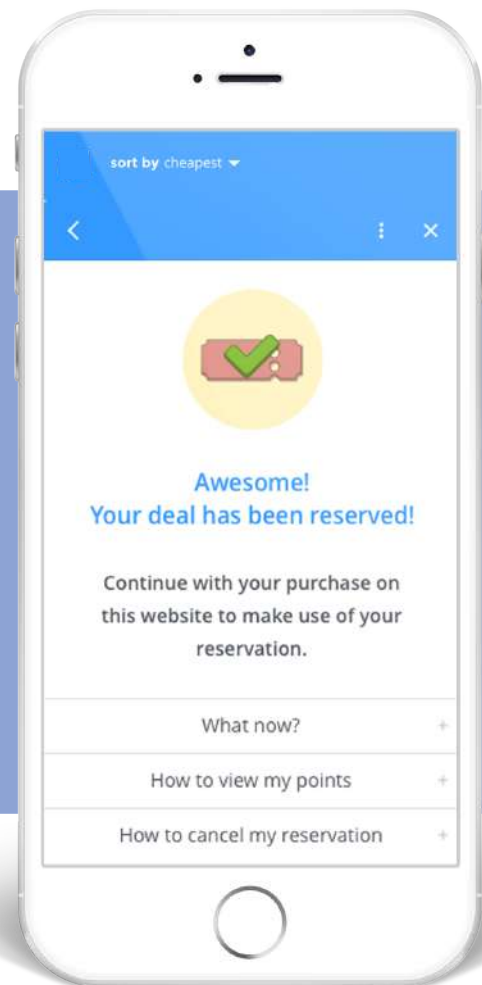
Project Description

The ***instinctools Company** was involved in a price comparison project with a unique capability to make personal offers to customers based on their profile and current market conditions. The Price Comparison platform aggregates offers from all shops, and shows them to the customer using a browser add-on and mobile application while the user is looking for the product.



The Personal Offers platform allows shops to offer personalized deals using a rule engine, taking into account a customer's profile and competitor's offers. This allows shops to implement a loyalty program and compete for best offer in terms of price, delivery terms or additional value.

The whole process takes a split second. Therefore, the best deals are presented to a customer while he is looking into a product and is ready to buy it right away.



Challenges

- ✓ **Redesign and refactor existing system**, going from a monolith application to micro-services architecture to support growing number of offers and impressions.
- ✓ **Implement online and offline deal making process**, reliable tracking transactions and accounting commissions.
- ✓ **Monitor and orchestrate micro-services** to provide high availability and low latency service to customers.
- ✓ **Support existing system** functionality lacking original developers' knowledge.

Solutions



We delivered **online and offline deal-making process** by collecting requirements, designing a new system, planning a change to the existing product and successfully integrating it together. The existing functionality has been maintained throughout the process. It was covered with automatic tests and redesigned to support a growing number of requests.

To support operations, we implemented **monitoring of backend services** while providing reliable tracking of transactions and making user's behavioral data available for data analytics.

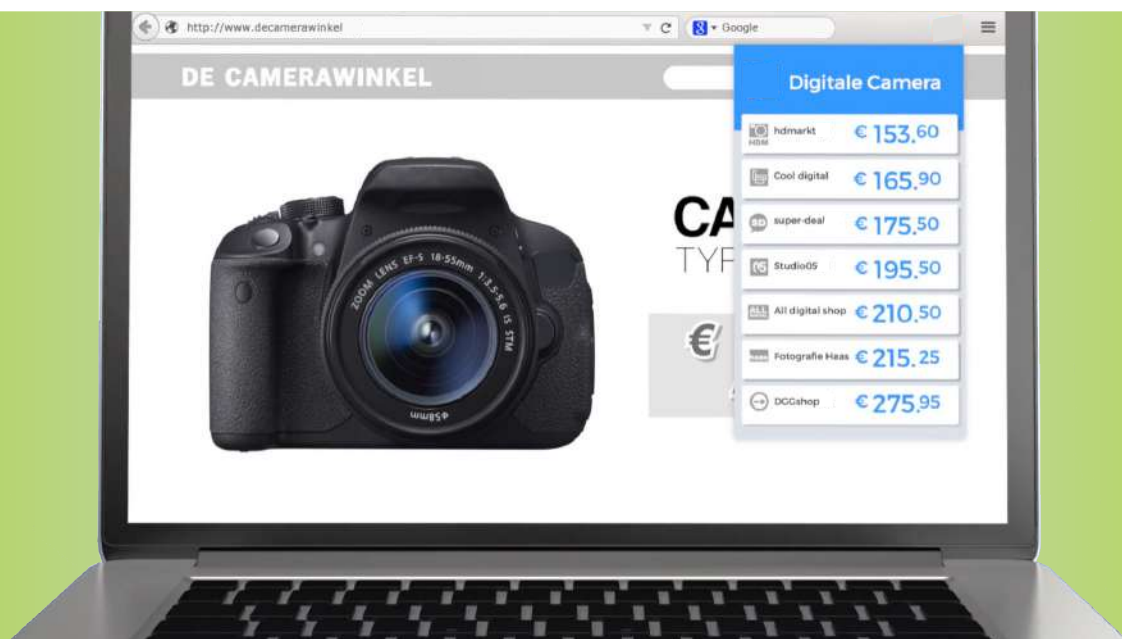
We have done **major refactoring to unify approaches** used throughout the micro-services, which lowered costs for operations and maintenance.

We also have:







- **fixed existing security** issues and **implemented best practices** to protect the business from common security threats, which was confirmed by penetration testing done by an independent contractor;
- **implemented event tracking** for behavioral analytics;
- **managed cloud resources** and implemented automation which reduced operating costs;
- **set up DevOps practices** and implemented continuous integration and delivery;
- **redesigned a system** to enable horizontal scaling.

Key Features

The client has obtained the desired functionality in time to run partnership programs with online and offline shops. **The delivered platform was cloud-ready and modular**, allowing the scaling of hot spots and handling business growth and peak load.



The benefits of the delivered platform are:

-  **Ability to handle increasing traffic.**
-  **Monitoring the system** in a real-time mode.
-  **Gathering of customer behavioral data** for analytics purposes.
-  **Integration** with third-party analytics products.
-  **Vendor-free deployment** to cloud, no lock-in to AWS.
-  **Performing the expected functions** fast and accurately.

The client was pleased with our fulfillment of the requirements and the recommendations of the affiliate partners. We are currently continuing to maintain the stable operation of the platform.

Technologies Used

- **JavaScript** was chosen as a common language for the platform. The existing system used a mixture of CoffeeScript and pre-ES5 code. We used **Babel** to bring the expressiveness of ECMAScript 2016 to every service.
- **Node.js** and **React** were chosen as two pillars to establish a common ground and allow, among other things, server-side rendering and code reuse between web application, browser add-ons and mobile application created in React Native.
- The system was deployed to **Amazon EC2** using **Terraform**, **Ansible** and **Docker**, with core components independent from **AWS** platform offerings. The delivered solution is not locked to AWS.
- **Amazon Kinesis**, **Redshift** and **S3** were used to build analytics platform open to third-party analytics products, e.g. Tableau and Amazon Quicksight.
- **MongoDB**, Elasticsearch and Redis were used as operational data storages.
- An increasing number of offers and data crunching tasks were gracefully handled by a scalable pool of workers getting jobs from the **RabbitMQ** queue.
- Continuous integration and delivery were implemented around Git repositories using the Gitflow model. The integrated setup of **Jenkins**, **Docker**, **Slack** and **GitHub Status API** was used to implement DevOps practices.
- **npm** was used for public and private package registry. The combination of semantic versioning, GitHub Releases and private npm registry allowed the sharing of knowledge inside a distributed team.
- **Prometheus** and **Graphana** were used to build a backbone for backend monitoring. Sentry was used on both the frontend and backend to catch runtime errors from all environments.



Node.JS



GitHub



React



RabbitMQ



npm



Elasticsearch



Redis



Prometheus



Graphana



Sentry



Docker



Babel



Webpack



Ansible



Jenkins



Terraform



MongoDB



Amazon Web Services