Digital product handbook

taiste

At its best, creating a digital product can be an exciting journey of discovery and creation. This document aims to describe some best practices for this endeavour.

During our 13 years in business, we have worked on projects big and small. We've both succeeded and learned some lessons the hard way. This is what these experiences have taught us.

Hopefully, this document is of use for anyone working on digital projects: product owners, project managers, developers and designers. Whether you're creating a new product or developing an existing one, we believe you can find something valuable here.

This document is updated as we live and learn new things.

01 Introduction

What is a digital product?

For us, it means any piece of software with a defined purpose. Every digital product reflects the context it's used in and the goals of its creators.

Creation of a successful digital product requires a combination of professional skills: Service Design, Digital Design, Software Development, Testing, Analytics and Marketing.

The end result can be a mobile app, web service or any piece of software. The possibilities regarding different platforms and use cases are endless.

Digital development: Horses for courses

Digital product development projects come in different shapes and sizes. For example the performance improvement of an established product is very different from the creation of a software product enabling a new and disruptive business model.

Each project requires a tailor-made combination of various efforts and suitable skills. The work effort distribution varies between the skills during the project.

Tools and perspectives

There are many processes for digital product development. Typically, a professional team applies these depending on what best suits the project. **This document however, does not aim to explicitly tell you which processes to use.**

Instead, we offer **a framework of things that need to be considered in every agile digital project, regardless of size and processes used**. Additionally, we present **various tools that you can mix and match**.

THE END USER IS AT THE CENTER

Design-driven development

Our philosophy for building digital products is **design-driven**.
This guides all ideas presented in this document.

Being design-driven means that the end user is at the center of our consideration from the get-go. This approach drives all phases of the project. A digital product must both serve its users and the business goals of its creators.

In digital product design, simple is beautiful and efficient. Ultimately, we believe that a design-driven approach is by far the most costefficient way of creating high quality digital products.

What is quality?

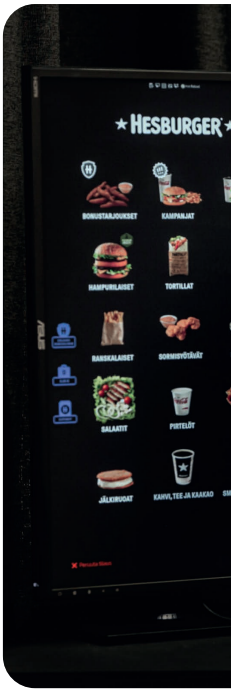
There is no shortage of digital products in the world; some are more useful than others. Our principle is to not produce digital waste. A high quality digital product has a clear reason to exist: **it serves the needs of its users and the goals of the organisation behind it**.

Quality also means the **intuitive user experience** enabled by a **robust technical implementation**. Elegance is achieved with complex problems turning into simple and beautiful solutions.

Quality assurance

Quality assurance is not a separate activity, but rather permeates the whole project. Quality is enabled by the right people using the correct processes and tools – backed by a sufficient budget.

This entire document aims to describe the fundamentals required to create high quality digital products. **Ultimately, the foundation of high quality rest upon the commitment of everyone involved. Every choice from big to small must reflect this.**



IT'S ALL ABOUT ADAPTING

On being agile

There is some variance in how agile is understood and applied to the actual work.

For us, the most important principle of agile is collecting feedback in different forms in every phase of development – and the willingness and capability to change direction based on it.

In design, this means validating the ideas with interviews and prototypes and other service design tools and methods. In technical implementation, it translates into peer review processes and careful testing loops. Analytics helps us to further collect data and measure it, both in terms of our user behaviour and KPIs.

02 Getting organised

GETTING STARTED

Building a foundation: The target and context

A successful digital project must begin by **establishing context**, **defining the targets** and **defining the available resources and understanding the limitations**. All of these need to be made clear to the entire team.

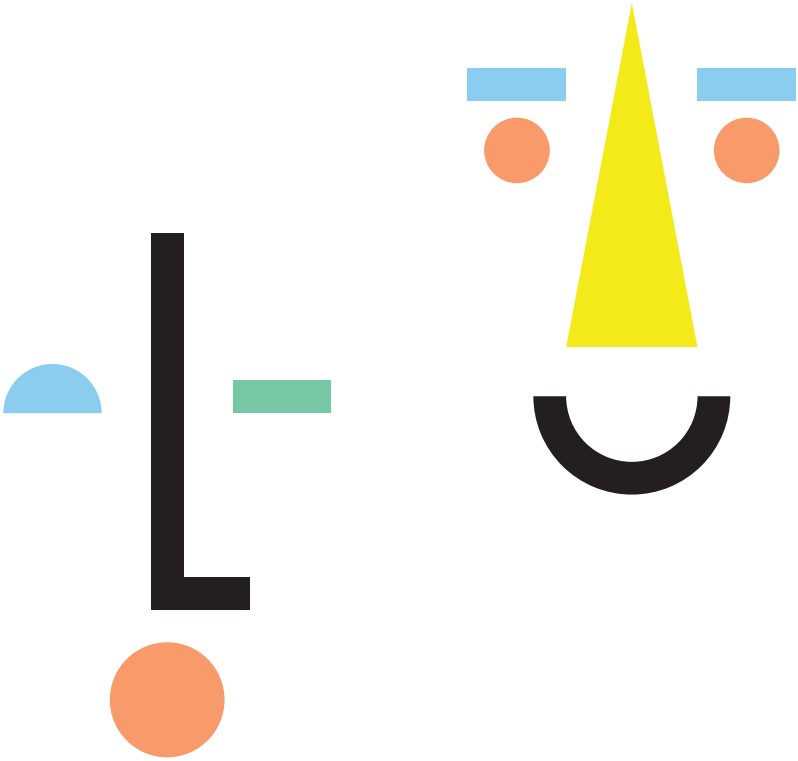
Only after these steps have been performed the appropriate set of actions can be chosen.This applies to all shapes and sizes of projects, and is perhaps the most important step when starting a project.

Building the team

The size and composition of the team depends both on the project and project phase. We follow the principle of having work pairs: everyone in the project should have someone to support their work.

A project should have a **Product Owner**, representing the client, who can make decisions to support the defined higher level goals.

The **Project Manager** is responsible for the deliverables. Other **team members** are selected based on the skills required.



Skills required for building a digital product

Digital product development is a multi-disciplinary effort. A good team composition is a balanced combination of various areas of expertise. Individual team members can have multiple roles.

Product Ownership

A Product Owner represents the client's voice in the project. He/she has in-depth knowledge of the client's business and how the digital product fits in it.

Project Management

A Project Manager will oversee the entire project. He/she is responsible for deadlines, KPIs and reporting, as well as arranging follow-ups with each project member.

Service Design

Service Designer(s) are responsible for the user research and making sure that the design of the product is done with the focus on the end-users.

UX/UI Design

UX/UI Designer(s) work to turn the research and business goals into user experiences that are intuitive and free of clutter. They constantly work with the developers to ensure that the UX is in line with the technical implementation.

Front End Development

Front End Developer(s) work on the interactive layer of the technical implementation. Anything the user can see, touch or click has to work smoothly and be built in a way that allows flexibly making changes.

Back End Development

Back End Developers(s) are the architects of what happens under the hood of the digital product. How the product functions with servers and APIs is crucial and can get complex depending on the requirements of the product.

Analytics & data

Gathering the right data tells us how well we've succeeded and informs us of the changes that need to be made in the following iterations.

Content creation

Content guides the user and sets the tone of the product.

Marketing

Marketing helps the product reach its intended audience and grows it over time. The work can be both textual and visual and involves deep knowledge of the marketing platforms.

Testing

Testing makes sure that the solution is clear of major bugs, secure and fulfils all of its functional requirements.

Building a foundation: Teamwork

Successful teamwork in projects has the same building blocks as any human relationships: mutual respect, trust, openness and the will to stay motivated and negotiate.

In all cases the communication between all the skill holders must be ensured from the start. This communication is a skill that requires practice.

Making decisions

The ability to make decisions is a paramount to ensure success. During a project there will be literally hundreds or thousands of big and small decisions to be made. All of these have to support the overall vision. The vast majority of decisions should be and usually are team decisions.

The party with the ultimate responsibility must however have the final say, just like a captain on a ship. But a wise captain always engages the team to maximise commitment and contribution.

Defining the target

Defining the desired outcome and the measures of success are the foundations of a digital project. How should success be measured? How do we know we have achieved our goals?

What is the core challenge? Do similar solutions already exist? Should the solution even be a digital product? Answering these questions before we decide to move forward is essential.

Project management principles

The work is typically split into a hierarchy of goals to be achieved. The naming conventions vary, but they can be e.g. milestones > epics > tasks.

The work is split into sprints. A sprint is a defined fixed amount of calendar time or work effort. Typically a sprint can be for example 2 - 10 weeks.

A sprint has a predefined set of outputs that should be achieved and related tasks. There can also be several concurrently ongoing sprints in a project. The sprint plans with associated timetables are the high level project plan.

Kick-off meeting

The kick-off is the first project meeting. This meeting has often been preceded by a round of quotation requests and offers or internal business plans. The target is to get to know each other and create an understanding of the project context and goals – and get the train moving.

A typical kick-off meeting can last between a few hours and a full working day. The next steps and preliminary time schedule are agreed upon.

Kick-off checklist

Company introduction
Introducing the ins and outs of the client's business

Introducing the project team
Roles and personnel from both the vendor's and client's side, responsibilities and contact persons, CVs of the vendor's project team members and references relevant to the project at hand

Introducing the project
Our initial plan for what we are about to create

Goals
What are we aiming for, how do we measure success?

Planning the routines
Weekly, meetings, communication, documentation, version control

Schedule
Project plan and timetable

Ground rules
Contract, budget, responsibilities, resources

Next steps
Next meetings, action points, etc.

Risks
Recognising risks and categorising them on the basis of likelihood and consequences


19:36
32 h
START
Mountain biking
Trail
Battery mode
SUUNTO

03 Design

Finding what matters

Design is about finding the essence of the required solution and giving it a functional and aesthetic form. An investment into good design enables cost-efficiency as only the essential is developed into the final product. There is no clear borderline between design and implementation in the context of digital products.

The philosophy behind finding the essential has been summarised by one Winston Churchill:

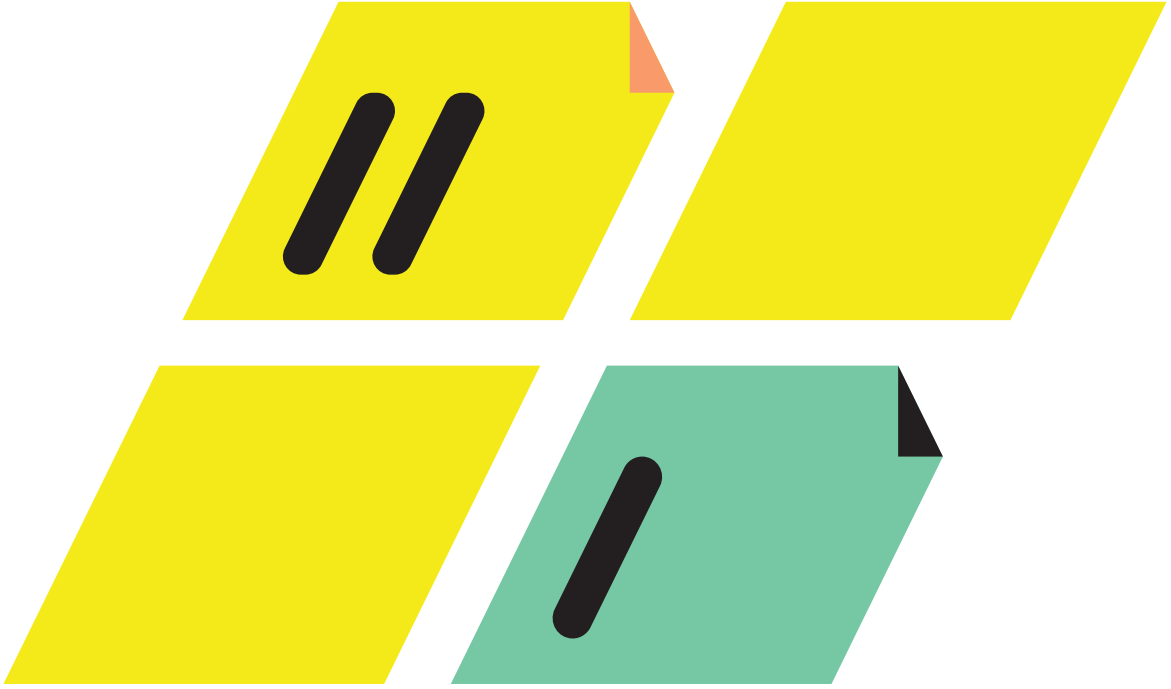
"If you want me to speak for two minutes, it will take me three weeks of preparation… If you want me to speak for an hour, I am ready now."

Design thinking and lean

Our design process is heavily inspired by both **Design Thinking** and **Lean** methodologies.

For us, Design Thinking is all about being solution-oriented – the main role of design should always be mapping real issues and finding ways to solve them.

From Lean, we're most influenced by the principle of involving both the clients and the end users in the iteration and testing process.



Co-design

Co-design workshops that involve both us and the customer are a key component of our process. This way, we make sure we share the same vision and take advantage of the fact that the customer knows their own business field the best.

In the first workshops, we define the **target group(s)**, **goals of the product** and **hypotheses** – and go through **the potential obstacles** known at that point. Typically from there, we move on to **creating user profiles** and **user paths.**

Co-design tools

Co-design tools are a powerful way of arriving at a shared vision for the product and making informed decisions regarding the design of the product.

Lean business model canvas

The Lean business model canvas is one of the most widely used brainstorming tools that helps gather the basics of the project in one easy-to-read template.

Featuring basic information like Opportunity, Solution, Competition, Value Proposition and Key Metrics, it helps to solidify the purpose, goals and challenges of the product we're about to build.

User journeys

User journeys visualise the users' interaction with the service over time. They can include descriptions the various phases of the user journey, what actions are taken and how different emotions and situations might affect the experience.

User personas

User personas provide more understanding of the people who will be using the product. These personas may include descriptions of the following: demographic information, financial information, marital status, lifestyle, descriptions of use situations as well as goals, hopes & fears related to using the service.

Prioritisation matrix

Prioritisation matrix is a tool for evaluating the different features of the project, making it more clear what to focus on.

Trigger cards

Trigger cards are a creative ideation tool that can help the team move forward with brainstorming by posing various "what if" questions.

Research and service design

For us, Service Design is a seamless part of design. Depending on the project, we collect more in-depth data with **observation and field studies**. Typically, we conduct **interviews** and **surveys** to gain a clearer understanding of the user needs and challenges.

User testing is also a normal part of our process, helping us validate both bigger ideas and highlight individual issues within the user paths.

The projects can also include **service audits**, where we research an test the usability or accessibility of the product.

"Creating user personas is an effective tool. But by itself, it can lead to poor representation of your users. I'd suggest always inding a balance between the internal service design work and maintaining contact with the users. People are always more complex than we think, and collecting data from them brings the whole team close to the mindset of our real-world audience."

Sanni Karlsson, UX Designer

UX design

Designing UX is the process of collecting data, prioritising and presenting it in a format that offers an smooth experience for the user. Often, the best user experiences are almost unnoticeable in their intuitiveness.

Typically, UX design starts with creating **information architecture** (what information is necessary and how should it be organised?) and moves on to **wireframes** and **prototypes** that visualise the flow of the service. With these tools, the product can be tested with the users and other stakeholders before writing any code.

UX design tools

From information architecture to interactive prototypes, UX tools bring the planning and ideas to life.

Information architecture

Information architecture is about what information to present and how. Finding out what information is essential to the product and organising it in a way that's intuitive for the user to navigate is a key component of a smooth user experience.

User testing

User testing should not be an afterthought. Rather, it is often a good idea to incorporate it with the design process from as early on as possible.

In the end, only the data from the actual users can reliably determine whether we're making the right design choices.

Service blueprint

A more comprehensive description of what the product does, service blueprint explains what should happen in the service at various different levels during the user journey.

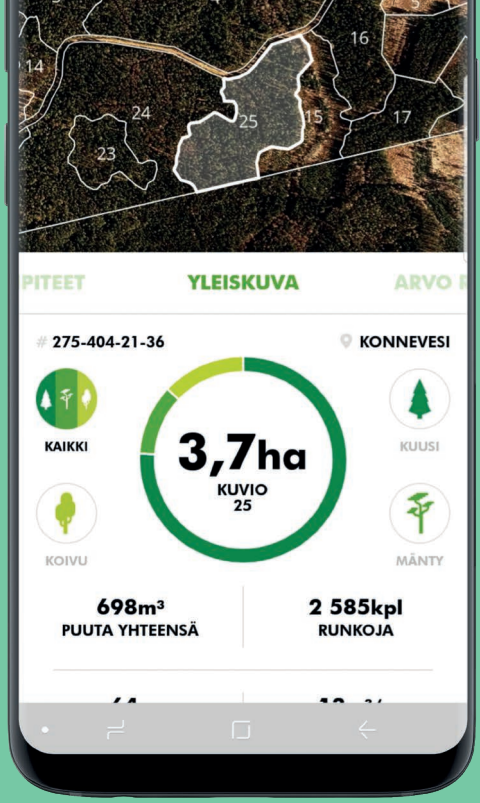
Wireframes

Wireframes present the flow between the different views of the service. It's an effective tool in finding out whether moving within the service follows a clear logic: the user should never get stuck or lost.

Prototypes

With various prototyping tools, it is possible to create clickable versions of the service before moving into production.

This is great for testing, since at this point it is much faster and cheaper to make even drastic changes to the UI when compared to coding.



THE DESIGN PROCESS

UI design

UI design is much more than just visualising wireframes. Rather, modern user interfaces are built using components and dynamic styles.

Design systems can go beyond being component libraries – they can be a design philosophy that enables a more consistent customer and brand experience across whatever digital channels a client may use now or in the future.

Our tool of choice for UI design is **Figma**, due to its co-working capabilities and flexibility. Over the years, we have built our custom plugins for advanced prototyping and translating styles and components directly into reusable code.


Dickies

04 Implementation

Choosing the best approach

A digital product is always a reflection of its technical implementation and vice versa. Implementation and design cannot be separated from each other. Both must be present from the start.

Technical implementation can be done in many different ways, utilising different technologies or approaches. Some are, however, a better fit than others.

The chosen methods are always a compromise between various factors. The choices should be made to ensure they stand the test of time and are easy to upgrade based on feedback from real-world use.

JARKKO LAIHO, SENIOR SOFTWARE DEVELOPER

Technological principles

Small is beautiful

From information architecture to interactive prototypes, UX tools bring the planning and ideas to life.

Simple is elegant

Overengineering is a common trap. Don't introduce needless complexity. Don't try to solve problems that you don't have.

Re-use is smart

Reinventing the wheel is not efficient or economical. Do it right once, and it keeps paying dividends.

Stable technologies over fashionable technologies

You don't want to rely on "the new hotness" in production. The boring, battle-tested option is nearly always the right one.

YAGNI (You Ain't Gonna Need It)

Build what is needed now, not what might be needed in the future. Build it smart so it's easy to extend later.

Light and informal (mostly) over heavy and formal

Unless lives are on the line, you don't need exhaustive change management or cumbersome quality control processes. Common sense and code reviews are the way to go.

Aspects that affect technology choices

There are multiple facets that need to be taken into consideration when designing the technical architecture. Here are some of the principles we incorporate in our work.

Required functionality and platforms

Based on the different user requirements there are multiple possible ways to solve the problem, each providing its own set of benefits and drawbacks.

We strive to use the most suitable tools for the task, and only a necessary amount of fundamental building blocks. Dependencies are liabilities so we tend to be conservative about them.

Available in-house competences

The competences of the project team should always be taken into consideration when choosing the technology stack for the project.

Future proof

On the back end side, we tend to be very conservative – use VPS, automate via Ansible, and keep things as simple and dependency-free as possible. For most solutions, being cloud-provider independent is also a good idea.

Existing technologies in use

The customer might have their own technology stack in use – we always try to fit in whilst incorporating our own tools of choice if possible. If the impedance mismatch is too big, it is our responsibility to the honest about it and possibly skip the project altogether.

Budget constraints

All technology choices should be realistic in relation to the budget. Where substantial budget constraints exist, we go with the principle of "test manually and create a good product" rather than try to force unit an integration testing into a budget that is too tight.

Scalability

Most services tend to work just fine by vertically scaling when needed: in practice, this means getting more powerful computers rather than building a cluster right away.

The former is suitable for most cases in the long run as well – while the latter brings in additional complexity that might not be justified for the use case.

Security requirements

We have a good automation of setting up services via Ansible that have the basic security principles baked in. Making use of these tools means things will be pretty safe from the get-go.

Reliability requirements

We tend to create clusters with VPSs and provide reliability through basic distributed system principles.

Technological flexibility

The best way to be flexible is to stick to best practices within the given framework as closely as you can.

When we create new abstractions, we don't do it because we are, for example, "unsatisfied with the way things are in Django". Rather, it needs to be justified by the use case and solve a problem at hand that would be harder in other ways. Sticking close to our tools is worth biting the bullet a bit.

Performance requirements

Using the proper algorithms and making sure we have indexed the database tables goes a long way. We don't try to optimise before we know where the bottlenecks of the system under development will be.

Performance requirements

We try to stick to the components that are already widely used and known to work as expected. Naturally, there are also some projects where experimenting with new components poses a lower risk.

Competence development and research

Some projects allow more explorative and brave technological decisions. In cases like these, it is crucial to make sure that everyone involved understands the potential benefits and pitfalls of the approach chosen before moving on.

Technical & architecture design

All projects tend to share some aspects in their fundamental design. We try to find the solutions where we separate the system to autonomous units that can work independently in a subtask as much as possible.

Early on, it is important to understand the data as well as the data flow patterns in the system. This helps modelling the rest of the solution around this knowledge.

We suggest employing a relational database as the fundamental data persistence layer. However, when there are strong offline requirements with multiple users interacting with the same dataset, we'd go with PouchDB/CouchDB based solutions.

If there is need for multiple services, the inter-process communication is mostly built around message queue abstractions, be it RabbitMQ or Redis. In some cases, we use HTTP REST APIs as the communication layer as well – but only in cases where the API was used by other clients too, and there was no evident way to make it queue based.

Depending on the need up the stack, we assess at least the interactions between the client and back end services to determine if we should go with a REST or GraphQL based solution.

Creating the environment

Typically, we create the environment based on an existing project. We have experimented with creating templates, but have found it difficult to implement cost-efficient and one size fits all solutions this way.

As we have done hundreds of projects, we have reference projects available for most needs. We recognise this is not the case for everybody.

This is a topic of investigation for us – we're always interesting hearing about better alternatives.

Task management

In bigger projects involving many people we always organise the work in Kanban-style agile process. The sprint is planned and stories are created in Shortcut which has an in-house built integration to GitLab making sure they are in sync.

We suggest providing an estimate for any issue in Shortcut. If it exceeds four days, try splitting the work into multiple issues instead to get a better grasp on what actually needs to be done.

The backlog is organised in priority order – the developer can just pick the task from the top of the list when the previous task is merged.

Implementation cycle

Once the developer picks the highest-priority issue to implement, the work is started in a new branch that can be created automatically in GitLab.

The developer then proceeds with the implementation and asks for input or advice from project members, the client or colleagues along the way. When the implementation is completed, a reviewer is assigned to the issue and a code review is conducted. Once the findings have been resolved, the code gets merged to the relevant branch and the CI/ CD pipelines take care of the deployment.

Quality control and testing

If the budget allows, we always suggest covering at least the most vital aspects of the solution with unit- and integration tests. That, along with a good plan for manual testing, is sufficient for most small-scale projects.

When deciding on creating UI tests, it is always preferable to implement them when the UI is to close to being complete.

Going live ⟶

A project should have at least the basic CI set up in e.g. GitLab to make sure the work gets packaged, (tested) and deployed to environments properly. We have a good set of CI configurations that can be incorporated in new projects and are easily worth the investment.

We also add a basic set of metrics and monitoring to the backend services, along with Sentry and other tools that make it possible to proactively find issues and system errors before the users might even know about them.

Monitoring

Having monitoring and error reporting for both the back end and front end makes it possible to quickly react to errors in production.

The specific set of data to be collected and monitored depends on the service. From a technical standpoint, it should be chosen on the basis of what best allows us to stay on top of things when preventing and dealing with possible issues. Certain metrics are always relevant: CPU and memory usage, instance load, number of successful requests etc.

Once we have created the CD pipelines, it is easy to update the system after the problem is identified.


15:35
29.6.2022
612345
Kulta | 0313
Kerrosateria
9,80 €
Kerroshampurilainen
— Poista: Viipalekurkku
Norm. ranskalaiset
Coca-Cola 0,4 l
2 x Juustohampurilainen
(2,60 €)
5,20 €
Myyty
15,00 €
Maksettu
0,00 €
Summa
15,00 €
Muokkaa
Lisää
Poista
Määrä
Ateriat
Hampuri-laiset
Tortillat Kebabit
Sormiruoat
Dipit
Salaatit
Kylmät juomat
Kuumat juomat
Pehmytit Jälkiruoat
Lapset
Kampanjat Tarjoukset
Omat
Kupongit
Foodora Wolt
Kerros-ateria
Mega-ateria
Jätti-pekoni-ateria
Kana-ateriat
Kana-ateria
Minikana
Kasvisateriat
VEKE®-juusto-kasvis ateria
VEKE®-soija-kasvis ateria
Lasten ateria
Lasten ateria

05 Testing

Testing requirements

Testing is an integral part of every phase of digital development. No software is bug-free but requirements vary when it comes to reliability: bugs in aviation software are different than bugs in a mobile game.

Defining these requirements and the testing process are essential for results that meet the expectations of all parties.

A key aspect of testing is to enable problem detection after the deployment of the digital product.

Project testing plan

With a test plan, we set the rules and goals for testing that best serves the project.

We define what is **in or out of scope** for our testing efforts, **who is responsible for organising and implementing** the different areas of testing, what **environments** we need take into account while testing and what the **testing schedule** looks like.

Time and budget will always set some constraints to testing. The test plan should reflect a shared understanding of what quality means in the context of the project.

Device & operating system test matrix

There's a large number of operating system and device combinations to consider when testing mobile apps. For web apps, the same applies to browsers and browser versions.

To determine which use cases are relevant to prioritise, the **device & operating system test matrix** (for mobile apps) and **device & browser test matrix** (for web apps) comes in handy.

Based on the current market data and the individual requirements of the product at hand, we create a map of probable OS/device/version combinations. These can then be prioritised based on time and budget.

Testing checklist

Test plan

Processes, responsibilities, defining goals and quality. Excel/Numbers document of user interactions within the software. Forming an internal test group.

Testing instructions

Creating a testing instructions document and distributing it. Listing known issues.

Distribution to the client

Forming a client test group. Distribution with a selected prototyping tool and teaching the client to use it.

Functional testing

End-to-end

Documenting the test results

Creating and updating a test transcript.

Moving the issues to GL

Highlighting the issues and adding them to GitLab

Compatibility testing

Different devices, browsers, operating systems etc.

"Hunting for bugs has the tendency to become increasingly time-consuming as the errors become more rare and specific. As bug-free software doesn't exist, knowing when to stop is essential. Resilience in digital product often also means that the software knows how to break as gracefully as possible in case something still goes wrong."

Oscar Salonaho, Managing Director

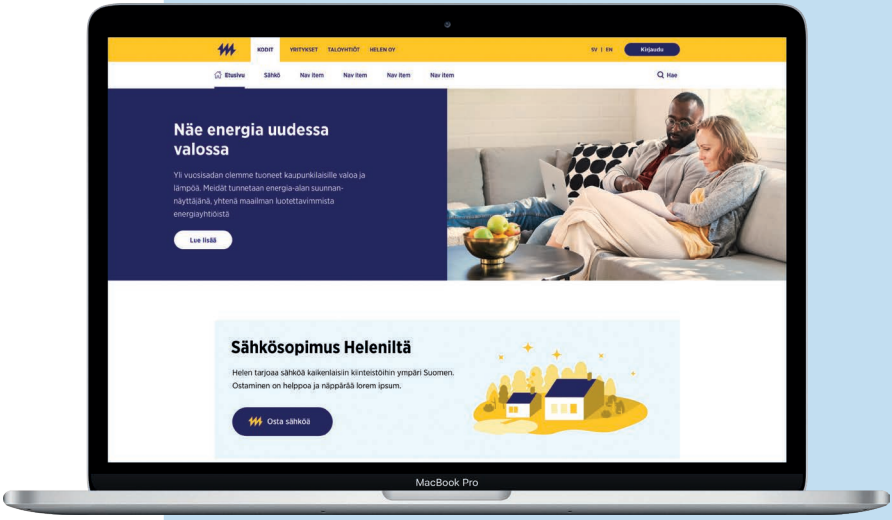
06 Analytics

Collecting data

A deeper dive into analytics is outside this document's scope. Here, we'll focus on a more general checklist that can be expanded upon depending on your project's needs.

Reasons for collecting data include:

- ⟶ Measuring success/KPIs
- ⟶ Monitoring functionality
- ⟶ Improving the product



Don't miss out

The first rule of data is: **you can never get back what you didn't collect**. Thus, the product should be integrated with the right analytics tools right away.

Having the data from a longer time period makes it possible to utilise it in more sophisticated contexts later on. When you start receiving data from other sources (ie. digital marketing), make sure to integrate those with your analytics too.

On the technical side, collecting error logs is crucial for fixing bugs and usability issues along the way.

Basic analytics

Creating an analytics setup should start with a discussion: what are our KPIs? How do we measure usage patterns? What actions count as conversions? Can we link the marketing channels with our analytics to measure the success of our campaigns?

> The most used analytics platforms (ie. Google Ads, Localytics) are packed with enough features for almost any needs, but it is **up to the user to make sure the setup and integrations work as intended**.
>
> Once the basic analytics setup is ready, it is a good idea to **bring the essential metrics to a dashboard** for easy access.

07 Risk management

Preparing for the what ifs

A digital project always comes with a set of risks, some of which are more likely than others.

The risks can be internal or external. Most software has integrations with other services and is susceptible to any problems that these third party products might encounter.

A **threat and security analysis** helps to both mitigate these dangers and to have a plan ready in case something goes wrong. Preparation and back up plans are essential in avoiding these threats.

Risk handling

Risk control includes preparation and procedures for action when problems are encountered. The OODA loop, originally developed by the US air force, is a very capable risk handling tool. It includes four steps that must be implemented by the organisation responsible.

- ⟶ **Observation**
  Observing the problem, preferably in advance
- ⟶ **Orientation**
  Determining the severity and extent of the problem
- ⟶ **Decision**
  Making the appropriate decisions
- ⟶ **Action**
  Executing the appropriate actions

Threat and security analysis checklist

Risks with third parties

The risks with third parties can range from temporary downtime to data corruption. These are best avoided by decentralising the service between several service providers and having private backups of the critical data (if possible). Third-party services are also prone to changes.

It's often a good idea to build the technical architecture so that it is not too dependent upon the implementation details of any individual third party product.

Software quality risks

Unit and integration tests are a necessity because they enable making changes and further development with lesser risk of any component or product breaking down. Budget-wise, this should always be taken sufficiently into account to avoid problems.

Moreover, there should never be a situation where only one person truly knows a piece of code. This can be avoided with collaborative peer review practices within the development team.

Force majeure risks

Software development is not immune to force majeure situations (accidents, vandalism, war, environmental catastrophes).

The world can be unpredictable: it is crucial to have a process for creating offsite backups that are always up to date and have the safety measures of the physical premises to be taken care of.

Security risks

DDoS attacks, information thefts and related criminal activities are always a possibility. To combat this, one should always avoid the "everything behind one locked door" phenomenon by having unique secure configurations everywhere.

Security-related DevOps knowledge is a good asset within a development team. Encrypting sensitive data and storing databases on separate, higher-security servers is often a necessary practice.

Social risks

Social risks include issues with personal relations, key personnel leaving the companies and problems in work culture. In digital projects, it's best not to have rockstars but rather teams that work together as units and document their work in a way that makes it easier for new people to jump in.

Culture-wise, enabling and monitoring the wellbeing of the team and encouraging open discussion is the key. Social issues should always be a high priority and the focus on measures that help avoid such situations in the first place.

Data loss

Data can be lost due to corrupted files or physical damage. Having the data available in several (safe) locations both digitally and in the physical world makes sure the team can rest easy.

08
In conclusion

IN CONCLUSION

Where do we go from here?

The most important aspects of a successful digital product are clear goals, right skills, proper tools – and last but definitely not least: teamwork.

When these pieces are in place, a digital project can be an inspiring journey from an idea to a finished product. **Let's get to work!**

Did the ideas presented here resonate with you?

Do you need help with your digital product?
Don't hesitate to get in touch!

Eeppi Nieminen
New business
eeppi.nieminen@taiste.com
+358 40 841 7566

About Taiste

2009

FOUNDING YEAR

Offices in Turku and Helsinki

100+

CLIENTS SERVED

From a wide array
of industries

~55

PROFESSIONALS

A balanced group of
designers and developers

6.7 M€

TURNOVER 2022
(estimate)

Steady growth

taiste