

Software technology

05 - The User Perspective

Dr. Attila Gludovätz

Online catalog – every week

- <https://catalog.inf.elte.hu/>
- Log in
- Username: yourUsername (@inf.elte.hu)
- Password: your email password
- Captcha: I generate a number for you...
- Lecture attendance is **not** optional! Max 3 misses and you are out

Customer or User?

- Is it the same?
- Lot small vs Few big customers
- Product launch: early or late?
- Competitors can copy everything...?
- Who will dominate the market? And Why?
 - User eXperience (UX) is important



UX - User eXperience Design (or UXD, XD)

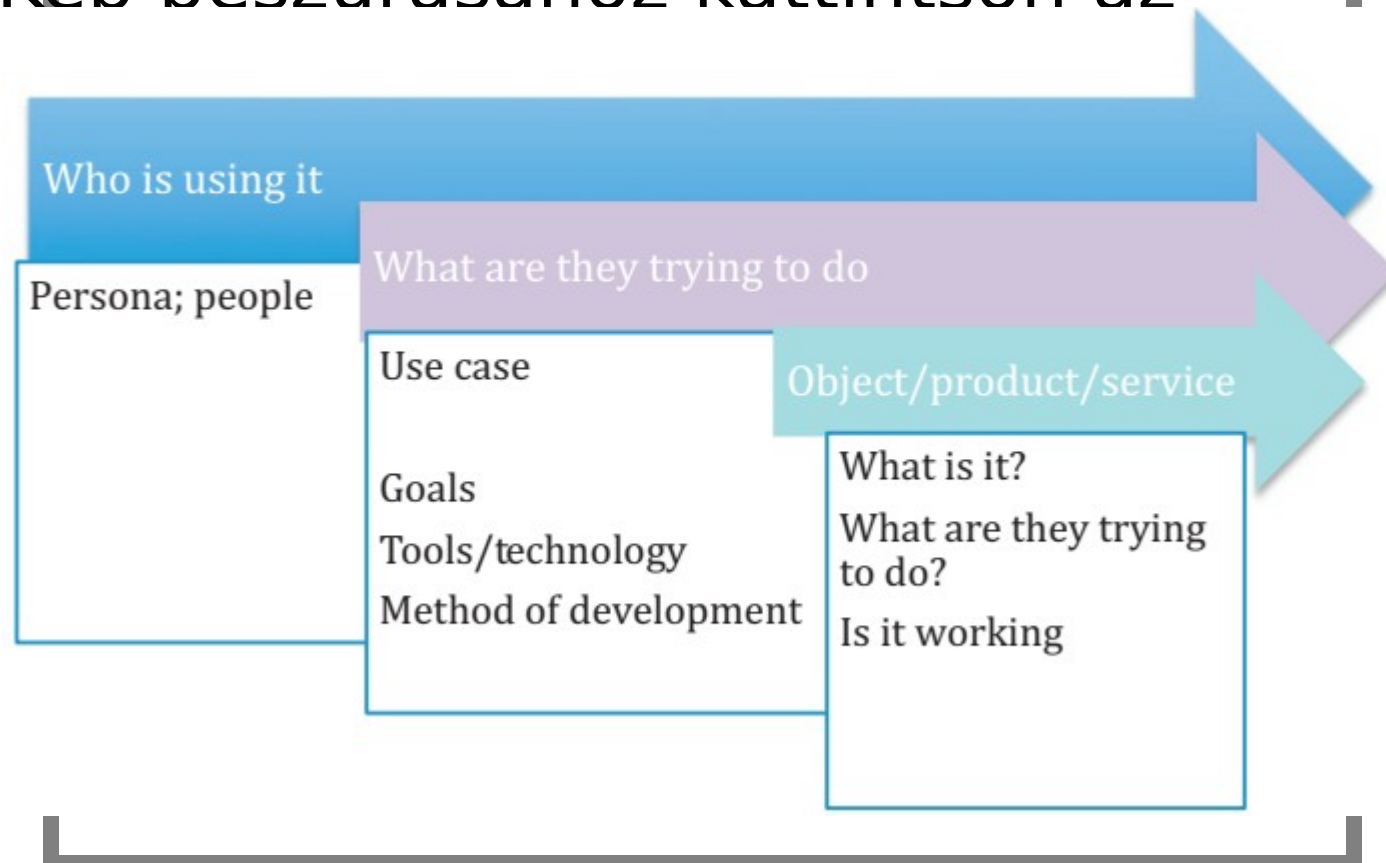
- Makes HCI (Human-Computer Interaction) more joyful
- Includes:
 - Visual Design (graphics, UI)
 - Structural Design (labels, finding stuff, interface organization)
 - Interaction Design (consistent patterns, intuitive behavior)
 - Usability (ease of achieving specific goal by specified user, includes Accessibility)

UX



UX Process

Kép beszúrásához kattintson az



UX – Use Case Kodak Brownie camera


**You Push the Button;
We Do the Rest**

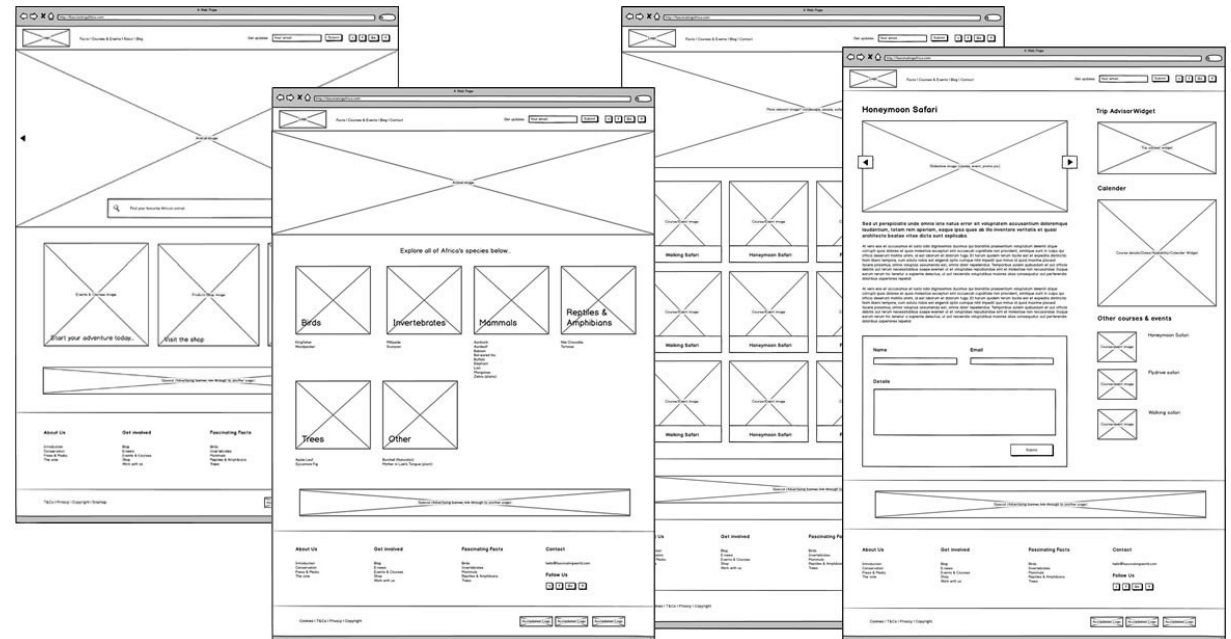
*Not everybody trusts
paintings, but people
believe photographs*

Kép beszúrásához kattintson az
ikonra



UX Practices

- Customer feedback, Surveys, User Interviews
- Problem sorting
- Wireframing 



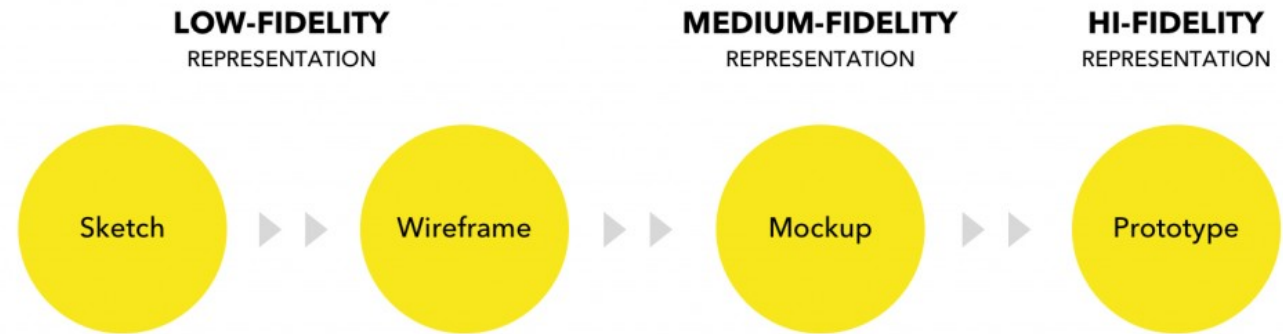
GUI mock-ups



What Is the Difference Between Wireframe, Mockup and Prototype?

<https://brainhub.eu/blog/difference-between-wireframe-mockup-prototype/>

Process of designing your first app

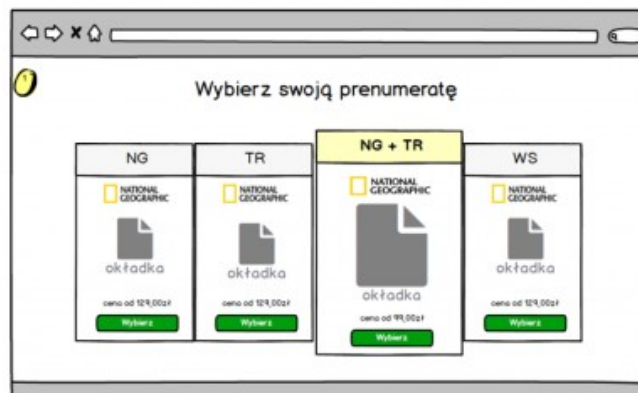


What Is the Difference Between Wireframe, Mockup and Prototype?

<https://brainhub.eu/blog/difference-between-wireframe-mockup-prototype/>

WIREFRAME

Structure + Functions + Content

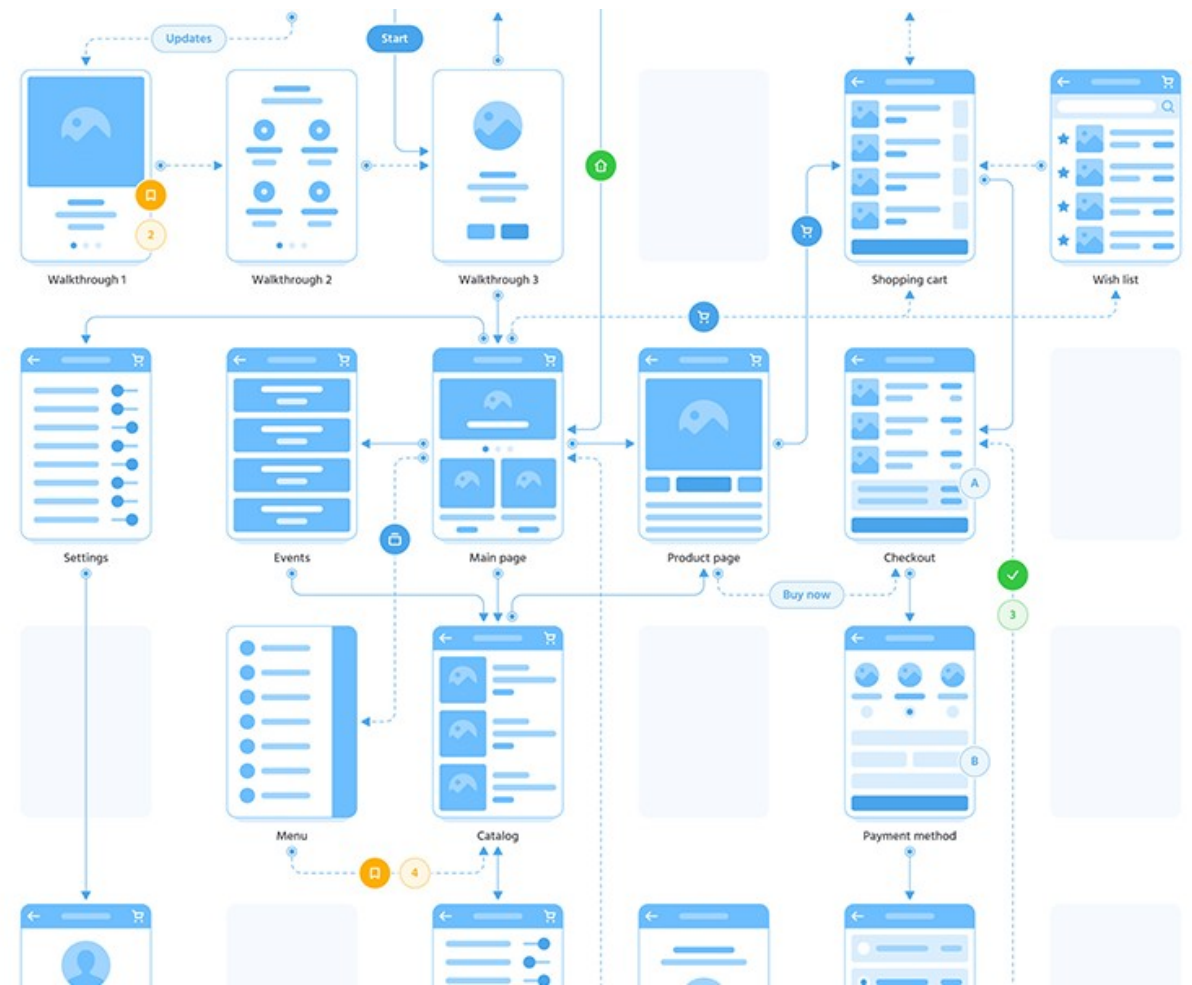
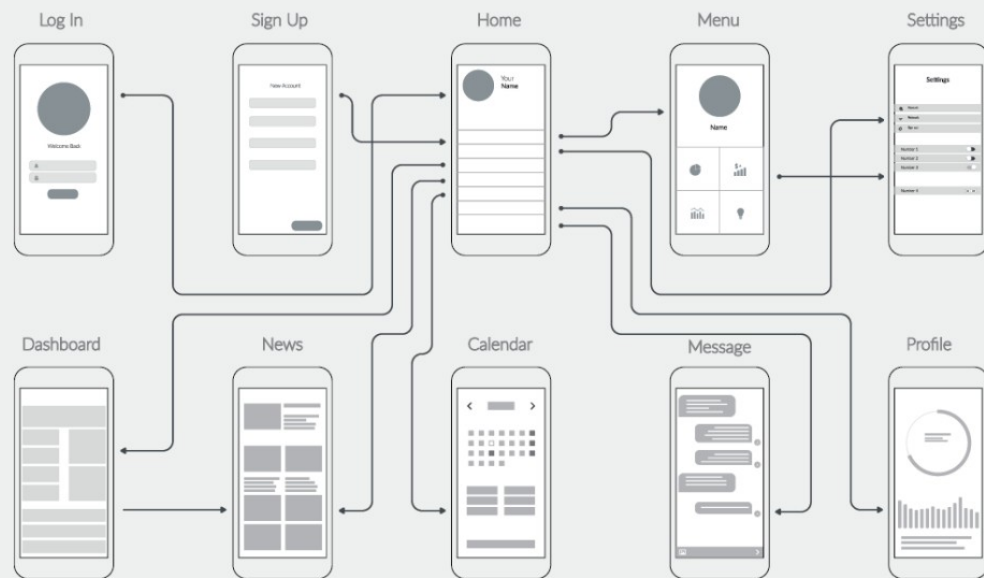


MOCKUP

Style + Colours + Right Content



Flowchart



User Persona


- **Persona = Person + Activity**
- Personas are the fictional characters created based upon the user research to represent the different user types that will use the service / product / site
- Creating personas helps to understand users, their needs, experiences, behaviors, and goals

User Persona

- The User Experience outcomes the best after the personas are well-defined, well-researched and well-analyzed
- The goal of personas is to create reliable and realistic representations of your key audience
- Effective personas represent a major user group for your website which would give a clear picture of the user's expectations and how likely they're to use the site
- Personas, however, fictional describe real people with backgrounds, goals, abilities, limitations, and values

User Persona

- Types of Personas (Goal-directed, Role-Based, Engaging, Fictional)
- Steps to design personas
- Developing Personas: Best Practice
- Elements of a Persona (pic✉)
- Evaluating User Personas

 <p>Peter</p>	<p>Works as product manager for a mid-sized company.</p> <p>Is 35 years old, holds a marketing degree.</p> <p>Has got experience working as a product owner on software products with agile teams.</p> <p>Has had some Scrum training.</p>	<p>Has managed mature products successfully. Now faces the challenge of creating a brand-new product.</p> <p>Wants to leverage his agile knowledge but needs advice on creating innovative product using agile techniques.</p>
---	--	--

User Persona (digital photography)



Daughter / Mom Marla

Age 40-55

Key Tasks

- Capture moments, take pictures
- Share pictures with family and friends via social media and email
- View pictures that are shared by family and friends
- Occasionally print pictures and put them on display at home or office

Motivations

Remember meaningful event
Relive events
Evoke warm memories

Frustrations

Can't find pictures when I want to
Too many pictures, no organization

Opportunities

Create organization of pictures as a by product of everyday events

User Persona (digital photography)



Teenager Theo

Age 14-18

Key Tasks

- Capture moments, take pictures
- Post pictures on high school web magazine
- View pictures that are shared by family and friends
- Post pictures on social media as a means of communication with friends

Motivations

Stay connected with friends
Communicate with peer group
Stay connected with family

Frustrations

Quality of pictures not consistent
Too many pictures, no organization

Opportunities

Create organization of pictures as a by product of everyday events

User Persona (digital photography)



Grandpa George

Age 70+

Key Tasks

- Capture moments, take pictures
- View pictures that are shared by family and friends
- Print out pictures to put into scrapbook

Motivations

Remember meaningful event
Relive events
Evoke warm memories

Frustrations

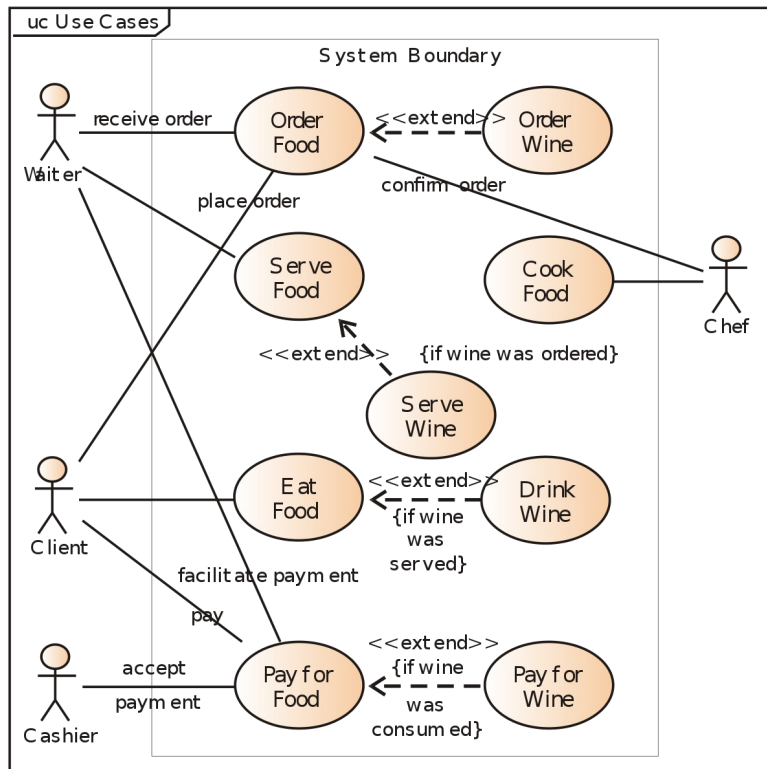
Does not own a digital camera
Can't find pictures when I want to
Too many pictures, no organization

Opportunities

Easy to organize digital photo album

UX Practices

- Scenarios (functional requirements)
 - Use Cases
 - User Stories
 - Storyboards, sequences
 - Simulations



- Personas are the most useful when paired with use cases which discuss in detail the process and steps of the task the user is carrying out using the object or the use case includes:
 - Goals;
 - User touch points with product, system, or technology;
 - Environment;
 - Actionable steps in order of execution;
 - User flow or journey;
 - Formulaic *"In order to do this, [persona] must first do this."*

Scenario: User Story

“As a <type of user>, I want <some goal> so that <some reason>.”

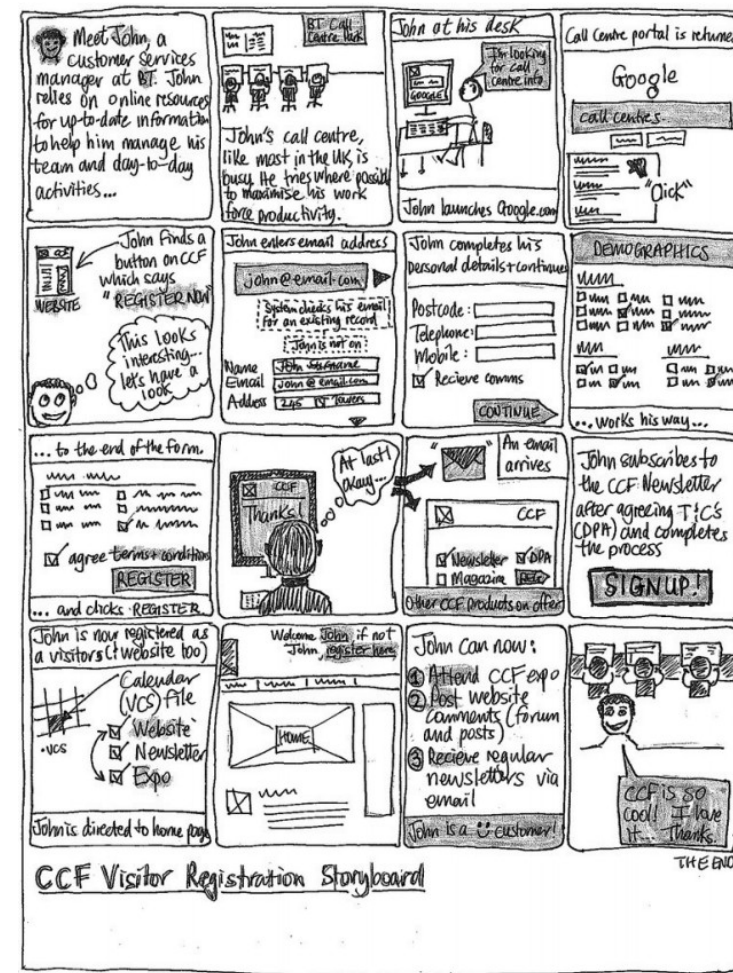
User Type	Epic	User Story
Mobile User	Registration	As a user, I can register for the application by entering my email, password, and confirming my password
		As a user, I will receive a confirmation email once I have registered for the application
		As a user, I can register for the application through Facebook
		As a user, I can upload a profile photo and add my name to my account
	Login	As a user, I can log into the application by entering my email and password
		As a user, I can log into the application through Facebook, if I previously registered with it
Web User	My Account	As a user, I can reset my password if I have forgotten my password
		As a user, I can view my personal information
		As a user, I can edit my profile photo
	Registration	As a user, I can edit my email. I will receive a confirmation email to my new email address.
		As a user, I can logout of the application from my account
		As a user, I can register for the application by entering my email, password, and confirming my password
	Registration	As a user, I will receive a confirmation email once I have registered for the application
		As a user, I can register for the application through Facebook
		As a user, I can upload a profile photo and add my name to my account

Scenario: User Story

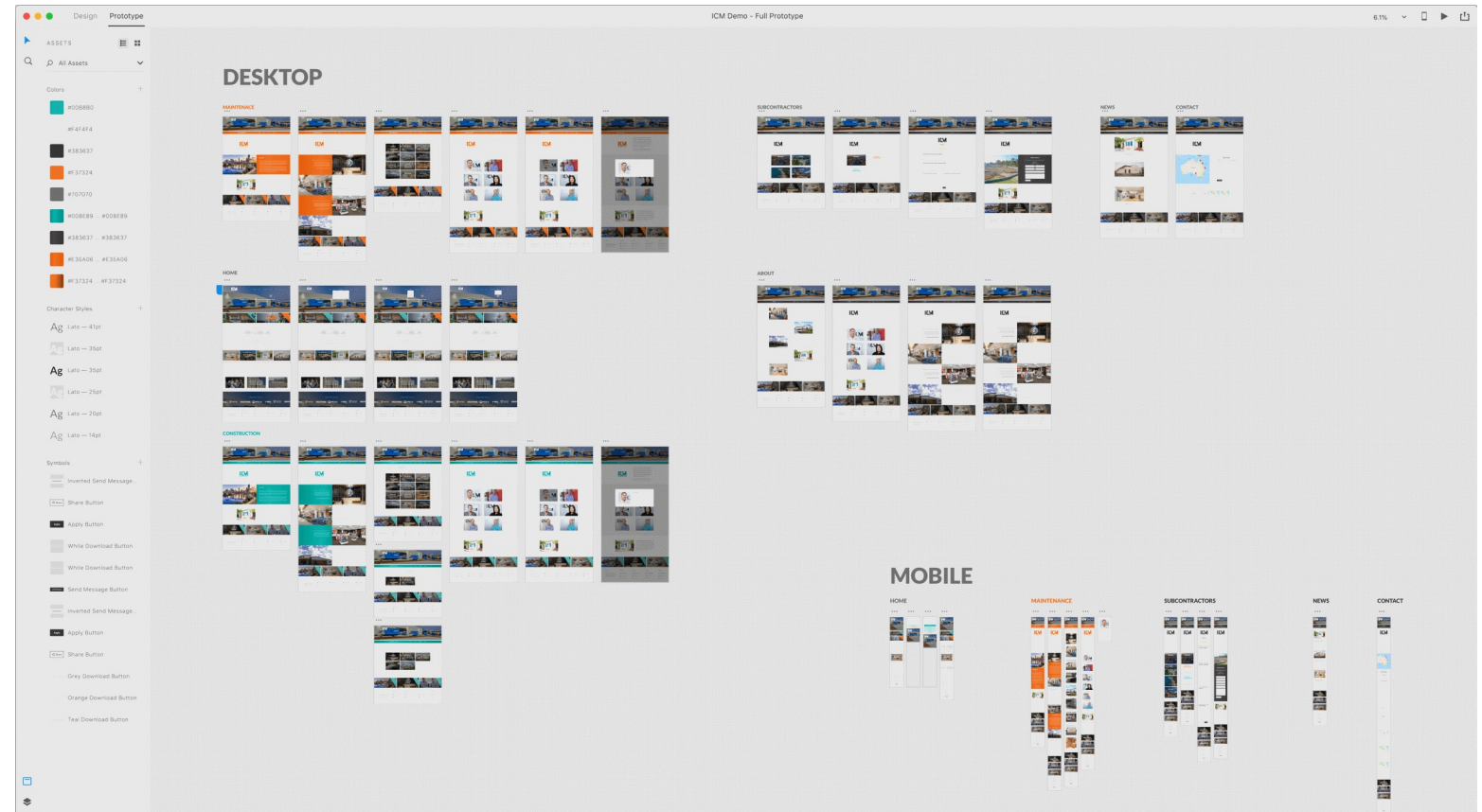
“As a <type of user>, I want <some goal> so that <some reason>.”

As a/an	I want to...	so that...
moderator	create a new game by entering a name and an optional description	I can start inviting estimators
moderator	invite estimators by giving them a url where they can access the game	we can start the game
estimator	join a game by entering my name on the page I received the url for	I can participate
moderator	start a round by entering an item in a single multi-line text field	we can estimate it
estimator	see the item we're estimating	I know what I'm giving an estimate for
estimator	see all items we will try to estimate this session	I have a feel for the sizes of the various items
moderator	see all items we try to estimate this session	I can answer questions about the current story such as "does this include"
moderator	select an item to be estimated or re-estimated	the team sees that item and can estimate it

Scenario: Storyboards, sequences

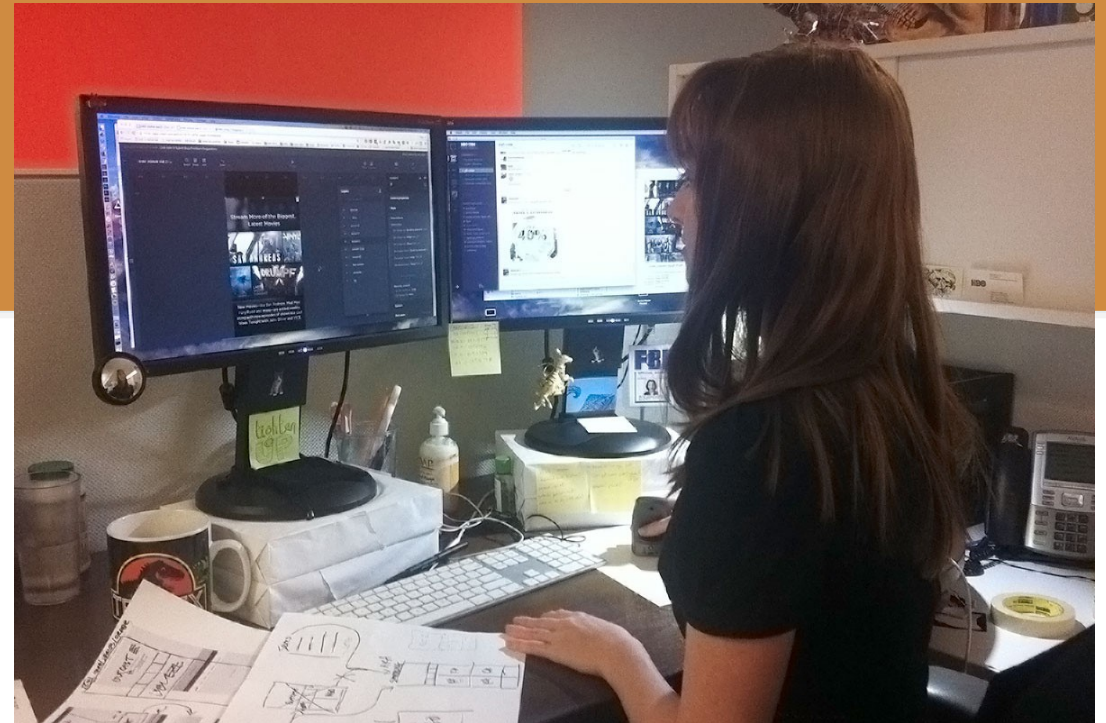


Scenario: Simulations Prototypes



UX Practices

- Prototypes (interaction prototypes)
- Usability testing (not market / qualitative research, **real testing!**)
- A/B testing
- *Logs and big data is really important here!*



<https://www.uxpin.com/customer-success/hbo>



Clic rate : 52 %



72 %

Usability testing

- The basic idea is to evaluate a product, service, or system by having a real person interact with it and test it
- The usability judged on various factors such as is effectiveness, efficiency, and overall satisfaction of the users during their experience
- Usability tests look to evaluate many user experience dimensions of a product, service, or technology to understand those dimensions

Usability testing

- Test methodology looks to evaluate several factors including:
 - **Design:** *how well the functionality fits into the users' mental model of the system?*
 - **Learnability:** *how quickly a new user can learn to use the system? How easy is to remember how to use it in the future?*
 - **Efficiency:** *how fast a user can accomplish tasks?*
 - **User satisfaction:** *self-report of how the user views the system.*
 - **Errors:** *how severe and frequently does a user make a mistake?*

Empirical Methods: A/B testing

- Usability evaluation methods, as well as business and market research A/B tests are used to compare two designs
- Occasionally this method is used for more than two designs or products
- In any event, the evaluation method uses the same protocol for both A and B; the only variable is the different design
- A/B evaluations are well suited for comparing elements of design such as:
 - Colors
 - Layouts
 - Components, such as buttons, banners, and other affordances
- A/B evaluations are done in both qualitative and quantitative studies

UX Practices

- Web tool: heat map
 - <https://heatmap.com/>



Lean UX – the Agile UX

- Lean UX is a practice that focuses on the actual experience that is being designed and less on the specific process deliverables such as reports
 - Both *Agile development* and *Lean developing processes* inspire **Lean UX**
- Lean UX focuses on the design stage of product development, getting feedback early and integrating it quickly
- *Lean product development is built on the foundation of Lean manufacturing, which looked to improve product manufacturing through the elimination of waste, while improving quality and managing the process toward perfection*

Lean UX – the Agile UX

- The **Lean software development** is based on the seven principles listed below:
 1. Eliminate waste, taking away anything that does not add value to the user
 2. Amplify learning
 3. Decide as late as possible
 4. Deliver as fast as possible
 5. Empower the team
 6. Build integrity in
 7. See the whole

Lean UX – the Agile UX

- The **Lean UX** movement is the next step in the evolution of the field of UX
- It makes sense, to integrate the team, to provide a process that brings software development and UX work closer together
- Lean UX is built on three principals:
 1. Design thinking, which applies design principals to all aspects of business
 2. Incorporation of Agile development principals
 3. Lean startup methods
- Incorporating design, development and business processes with UX makes sense because it brings the evolution of UX, almost, to a full circle

Lean UX – the Agile UX – summary

- Cross-functional teams
 - UX expert, designer is part of the team
 - Involved in the same iterations, development cycles
 - Common problems (especially prototypes, GUI, A/B testing)
- MVP (Minimum Viable Product) determination

Lean UX – the Agile UX – summary

- Iterates on
 - Who uses the product?
 - How is it used?
 - What is the user's goal?
 - Functionalities ordered in importance / user value
 - Identifies risks
- Continuously communicates with developers
- Trendy (see all major product successes lately!)

Story-driven Modeling

- **Story-driven modeling is an object-oriented modeling technique**
- Other forms of object-oriented modeling focus on *class diagrams*
 - Class diagrams describe the *static structure of a program*, i.e. the building blocks of a program and how they relate to each other
 - Class diagrams also model data structures, but with an emphasis on rather abstract concepts like types and type features

Story-driven Modeling

- Instead of abstract static structures, story-driven modeling focuses on concrete example scenarios and on how the steps of the example scenarios may be represented as **object diagrams** and how these object diagrams evolve during scenario execution

Story-driven Modeling

- Story-driven modeling proposes the following software development approach:
 1. Draft scenarios (textual)
 2. Mock-ups
 3. Storyboarding
 4. Class diagrams
 5. Algorithm design
 6. Implementation
 7. Testing

Just for fun :)

Try it?!

<https://userinyerface.com/>



User Inyerface

a bagaar frustration

Hi and welcome to User Inyerface,
a challenging exploration of
user interactions and design patterns.

To play the game, simply fill in the form
as fast and accurate as possible.

NO

Please [click](#) HERE to GO to the next page

Questions?

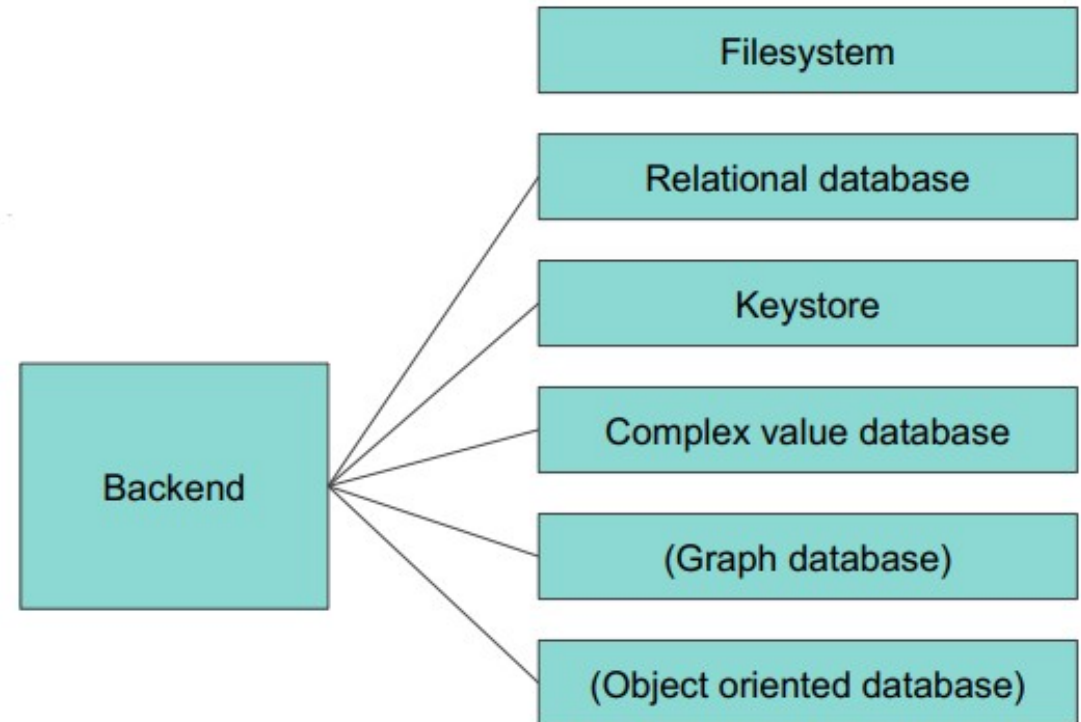
- ...
- Or write me an email to gla@inf.elte.hu

Persistence

- What is the primary role of the backend?
- To implement the business logic.
 - Login / logout
 - e.g., ask for new chat messages (information that belongs to the logged in user)
- To act as a central information store.
 - Store information in long terms - persistence
 - To store the information related to the logged in user (access control)

Persistence

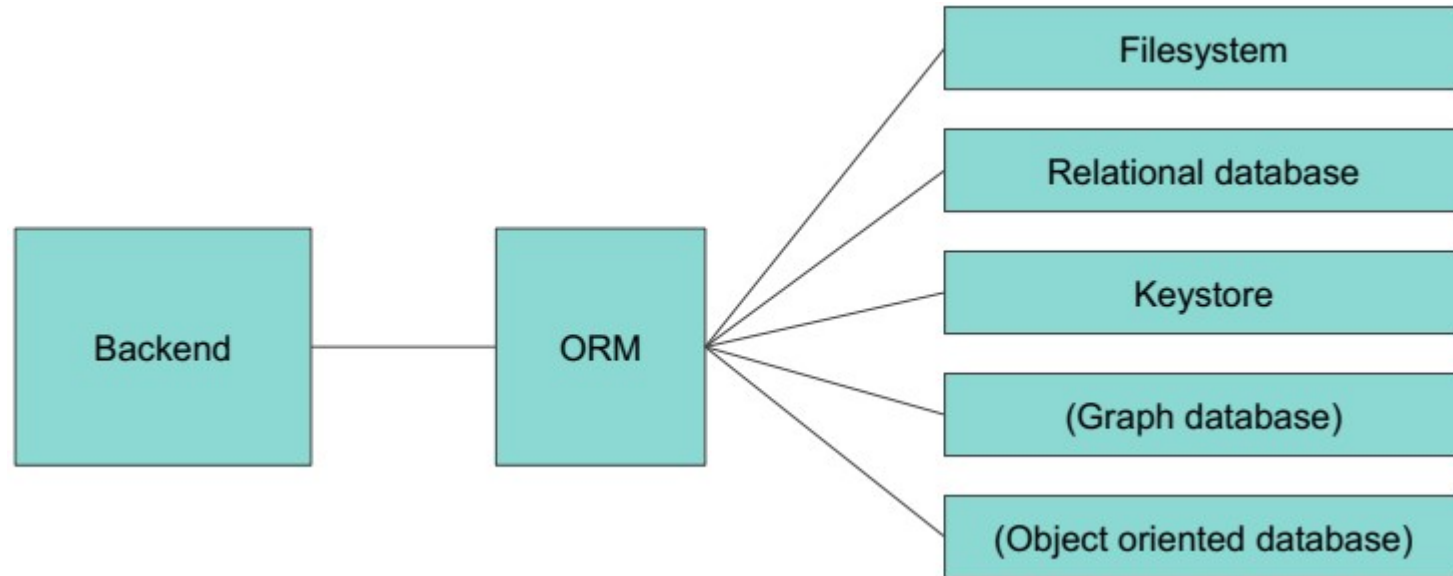
- Where to store information



Introduction

- ORM has a history: https://en.wikipedia.org/wiki/Object-relational_mapping
- There are several implementations:
 - https://en.wikipedia.org/wiki/List_of_object-relational_mapping_software
- Object relational mapping - entity relations are mapped to the relational structure
- Typically RDBMS is involved to store structured data, but other mappings are also feasible / exist
 - Keystore
 - OODBMS (identity)

ORM in the architecture



Specialized Services

- Enterprise level business software frameworks provide specialized / higher level services
- I will explain it in the case of Java, as I work with Java. I present you concepts / paradigms / techniques which typically exist in several programming languages
- Java Enterprise Edition (J2EE)
 - https://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition
 - Persistence - Java Persistence API 2 (JPA2)
 - Dependency injection
 - Logging
 - Sending emails

Class Relationships

- Entity relations are mapped to the relational structure
 - The classes are beans and implement the Serializable interface
 - The classes are annotated with the @Entity annotation
 - The relations between the classes are defined
- Some annotations are
 - @MappedSuperclass, @Entity
 - @Id, @OneToMany, @ManyToOne, @ManyToMany
 - @Table

Conversion

- The J2EE container collects the information (basically the class relationships)
 - Generates the entity relationship model (based on the annotated classes)
 - Converts it into a relational model
 - Synchronizes the model with the database (creates tables, keys, indices)
- You can ask JPA2 to update the underlying database
- (Pitfall: You can also ask JPA2 to reset your database at start)

Advantages

- No DDL, you define the data by software code
- No SQL, you use HQL to query class instances
- No SQL, ORM does the type conversion implicitly, thus the RDBMS is more integrated (type checks)
- No SQL, you do it “fullstack”, only Java

Disadvantages

- HQL is not very flexible, e.g., with grouping. This is the reason, why there are two interfaces
- You can query also by code (criteria API)
- An additional abstraction layer. Now you work not only with transactions but also with sessions
- More memory is consumed, as Hibernate maintains a cache of entity instances
- FetchType - Needs a lot of fine tuning, as loading an entity may loads all the related entities

Object Oriented Database Design

- You can define your database with a hierarchy of classes
- Inheritance works
 - Separate table per class
 - Joint tables for the classes
 - A single table for all the classes
- Interfaces also work - Java aspect

IdentifiableEntityInterface

```
public interface IdentifiableEntityInterface extends Serializable {  
    public void setId(Long id);  
    public Long getId();  
}
```


IdentifiableEntity

@MappedSuperclass

```
public class IdentifiableEntity implements IdentifiableEntityInterface {  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO, generator =  
"IdGenerator")  
    @TableGenerator(name = "IdGenerator")  
    private Long id;  
    @Override public Long getId() { return id; }  
    @Override public void setId(Long id) { this.id = id; }  
    ...  
}
```

NamedEntity

```
@MappedSuperclass
public class NamedEntity extends IdentifiableEntity implements
NamedEntityInterface {
    @Column(nullable = false) private String name = new String("");
    @Override public void setName(String name) { this.name = name; }
    @Override public String getName() { return name; }
    ...
}
```

EmailTemplate

```
@Entity(name = "Alternative_Table_Name")
public class EmailTemplate extends NamedEntity {
    @Column
    private String subject = "";
    @Lob @Column(name = "html")
    private String html = "";
    public String getSubject() { return subject; }
    public void setSubject(String subject) { this.subject = subject; }
    ...
}
```

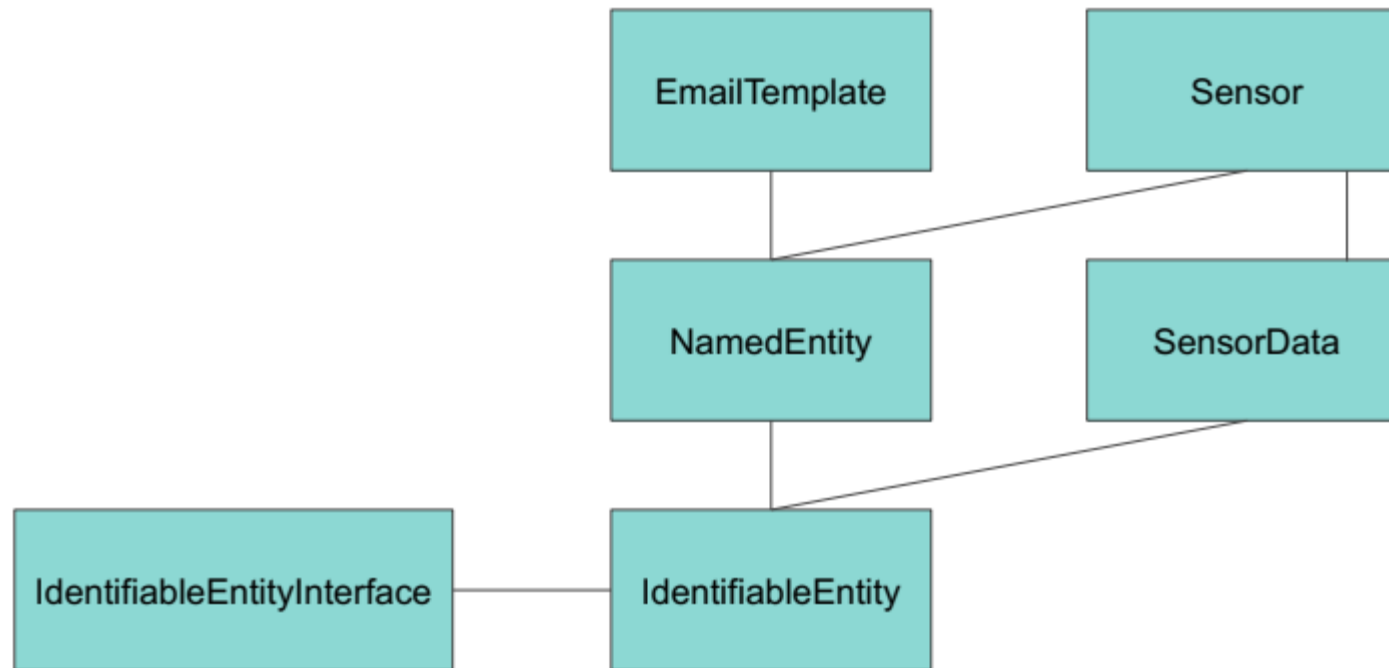
Sensor

```
@Entity
public class Sensor extends NamedEntity {
    private static final long serialVersionUID = -1539975510265701496L;
    @Column(nullable = false)
    private String token;
    @OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL,
        orphanRemoval = true, mappedBy = "sensor")
    private Set<SensorData> data;
    ...
}
```

SensorData

```
@Entity
public class SensorData extends IdentifiableEntity {
    private static final long serialVersionUID = -5399393509549115510L;
    @CreationTimestamp
    private Date timestamp;
    @ManyToOne(cascade = { CascadeType.MERGE })
    private Sensor sensor;
    private double value;
    ...
}
```

Class Relations



Hint

- When defining the class relationships take a look at the generated RDBMS model
- You will get a feedback on your data structure
- It holds especially for container types (set, list, map)

ASP.NET - Entity Framework

- Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects
- It eliminates the need for most of the data-access code that developers usually need to write
 - What is Entity Framework?
 - <https://www.entityframeworktutorial.net/what-is-entityframework.aspx>
 - Creating an EF data model for an ASP.NET MVC application
 - <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application>

Laravel - Eloquent

- The Eloquent ORM included with Laravel provides a beautiful, simple ActiveRecord implementation for working with your database
- Each database table has a corresponding “Model” which is used to interact with that table
- Models allow you to query for data in your tables, as well as insert new records into the table
- <https://laravel.com/docs/8.x/eloquent>

Questions?

- ...
- Or write me an email to gla@inf.elte.hu