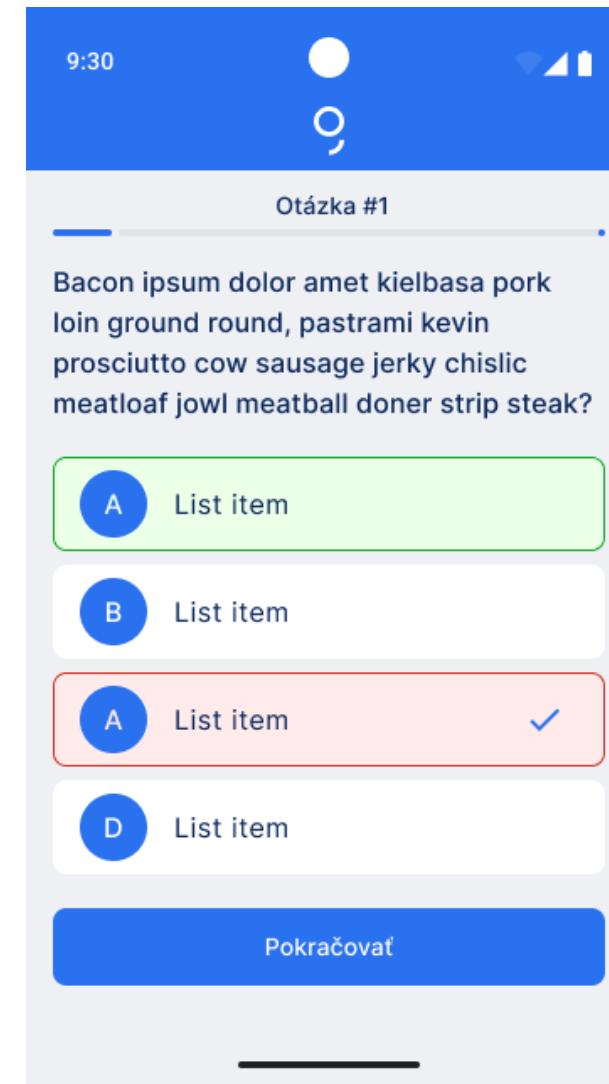
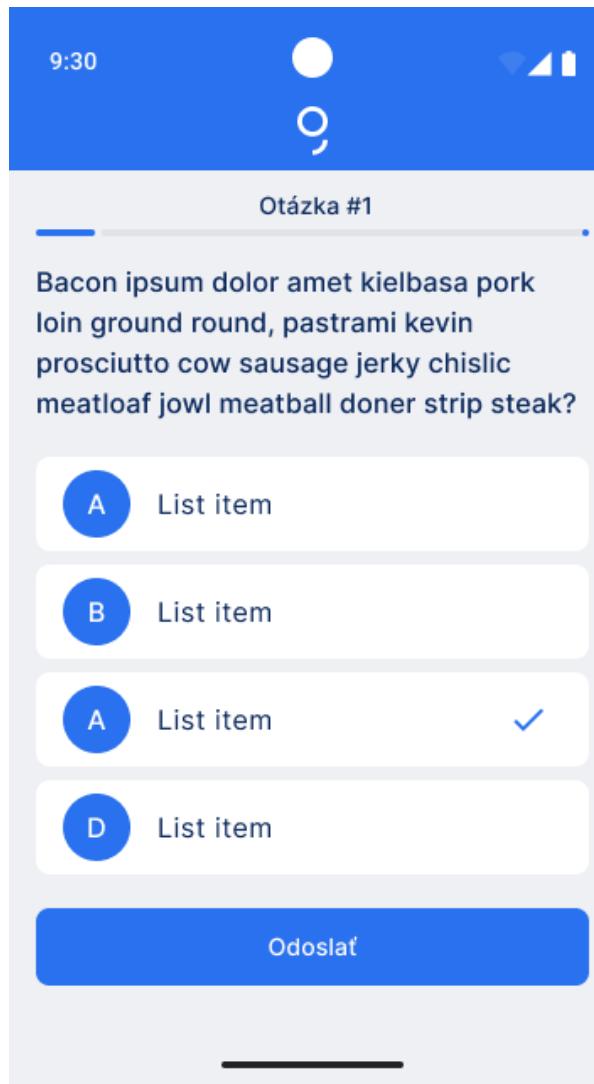




How IT in banking works

Tomáš, Martin, Martin
FEI STU, 20.3.2024

Demo application



Try it out!



Business Analysis

BA Drivers

- Company Strategy
- Group Strategy & Initiatives
- BA/PO Initiatives
- Inter-Team Dependencies
- Compliance & Regulations

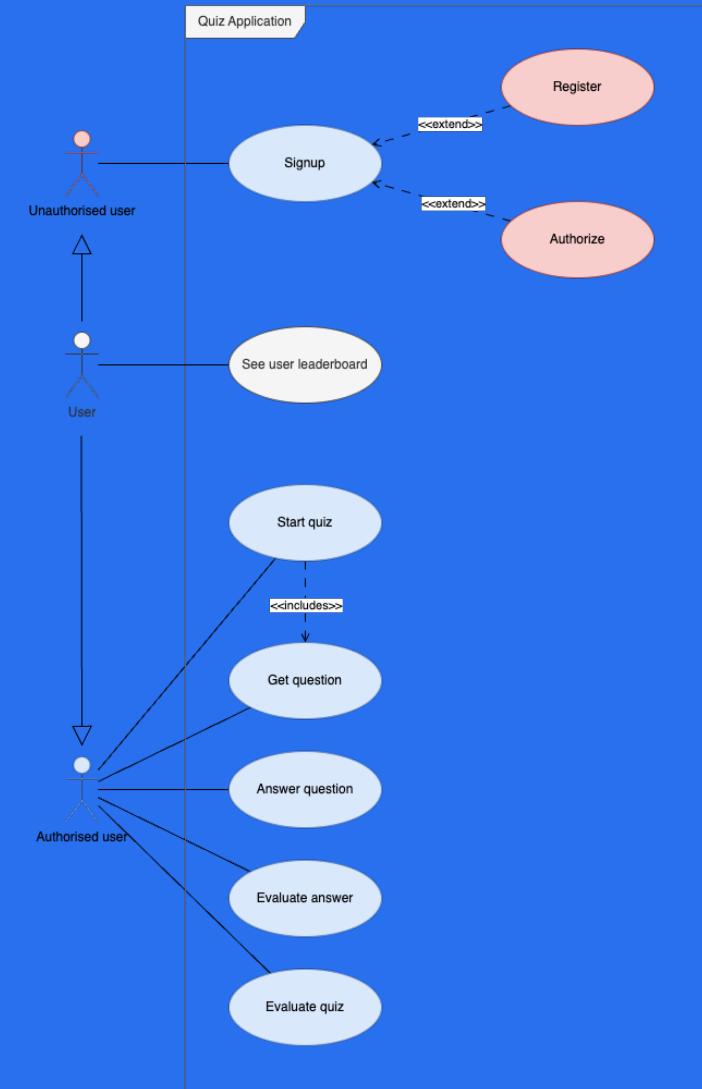


Business Requirements

- Solve business problem
- Achieve business objectives
- Business Assignment = Proposal for the change
- Quiz Demo - BA



draw.io



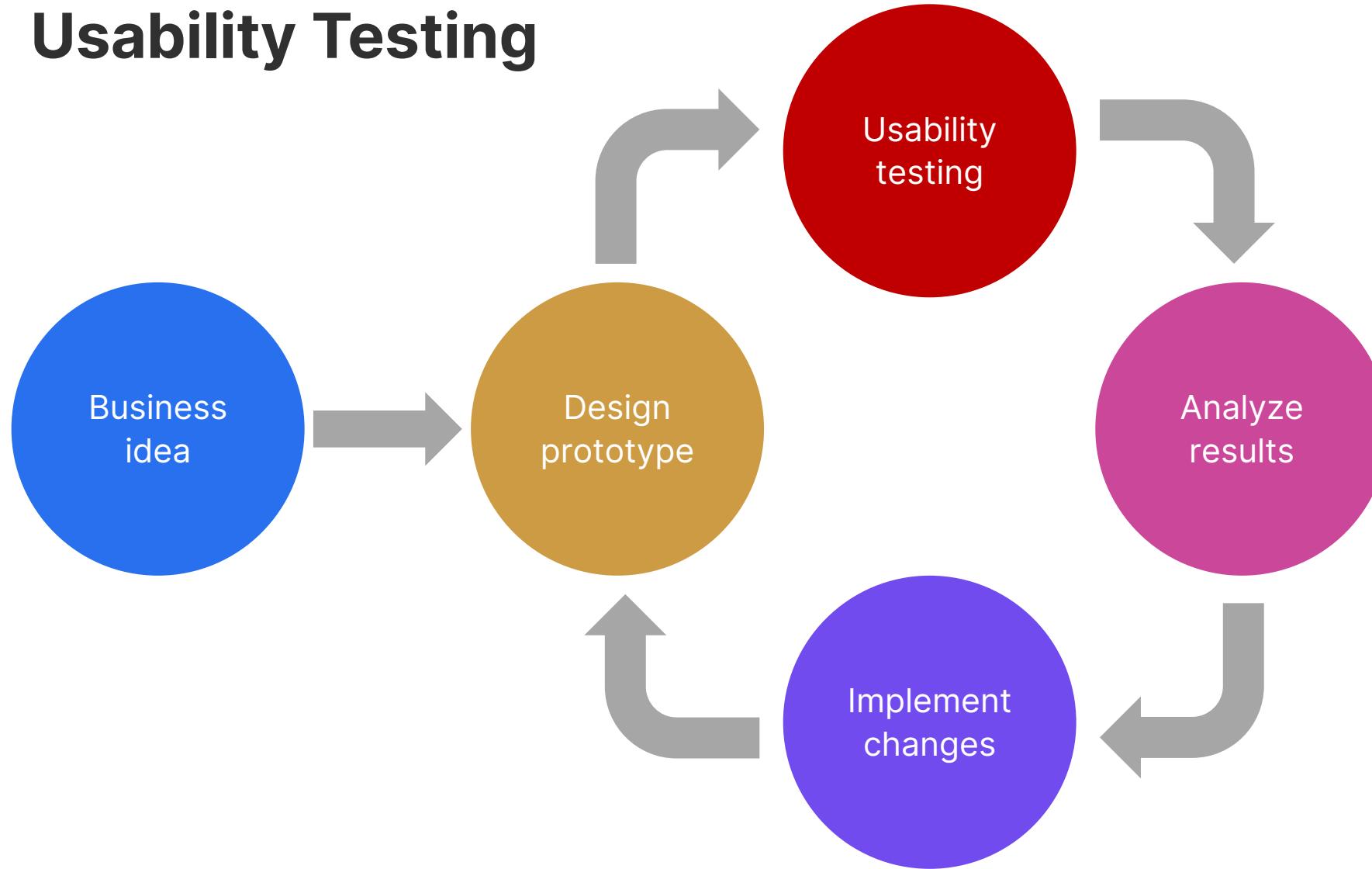
User Experience

1. Data analysis

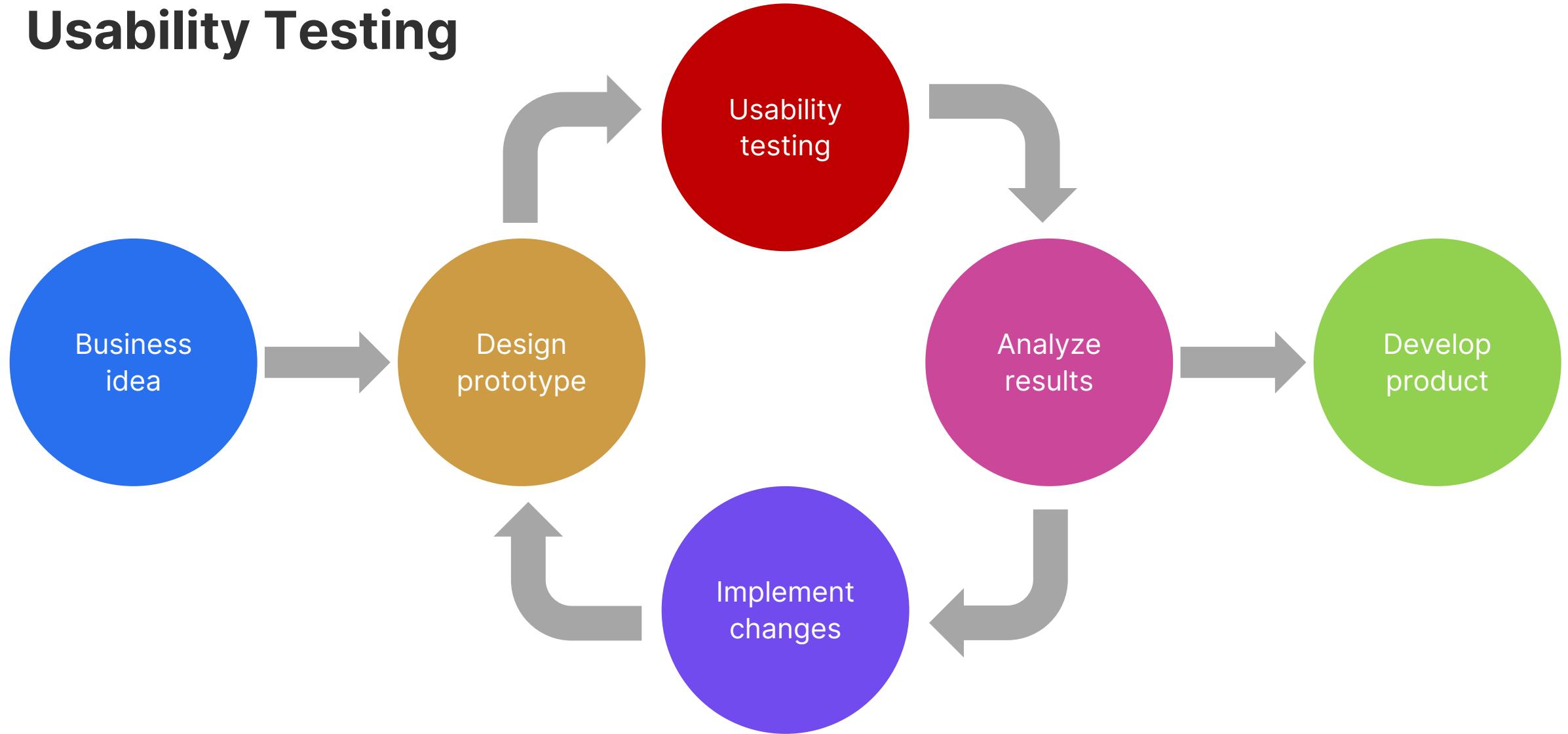
2. Quantitative research

3. Qualitative research

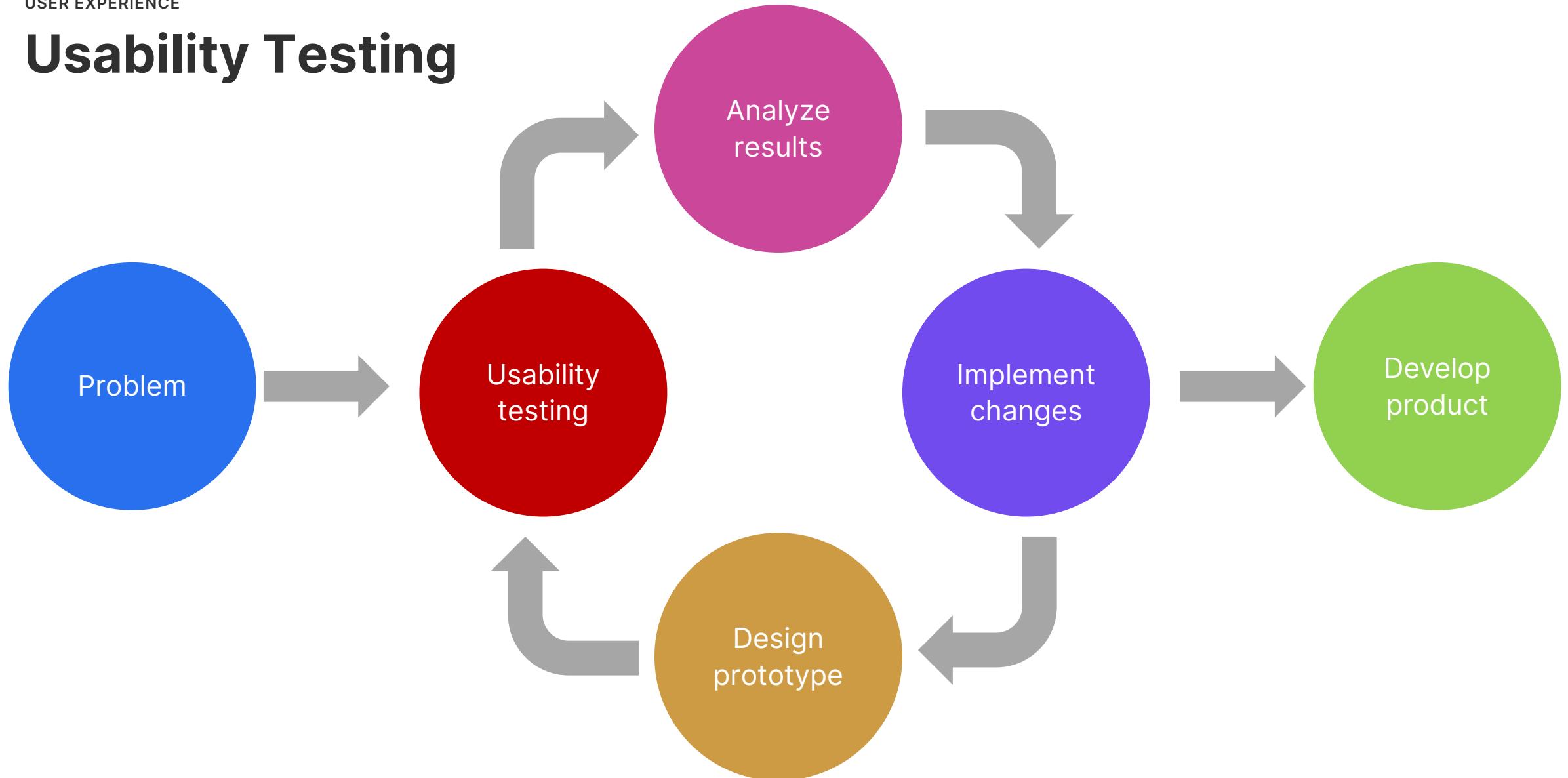
Usability Testing



Usability Testing



Usability Testing



Solutions Architecture

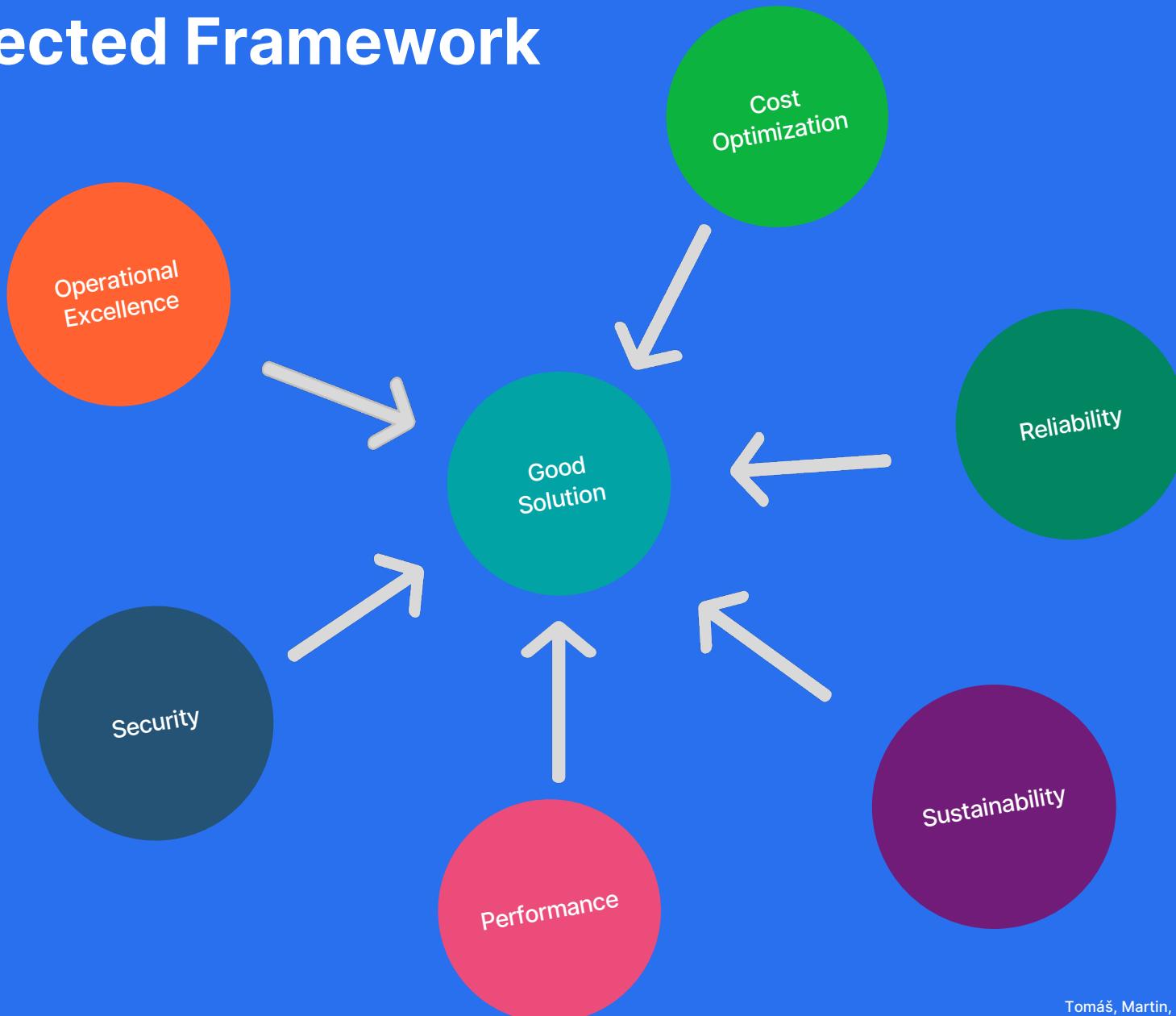
From Goal to Solution Design

- Understand Business Requirement
- Well-architected Framework – 5 (6?) Pillars
- Bottlenecks and Alternatives
- Describe and Draw Everything
- Repeat First Step Until the Solution is Done :-)

Understand Business Requirement

- What client said:
 - I want to use a online quiz application to understand client needs
- What client means:
 - 24/7 availability, simple and fast, configurable and reusable, reports on demand, more quizzes at once?, ...

Well-architected Framework

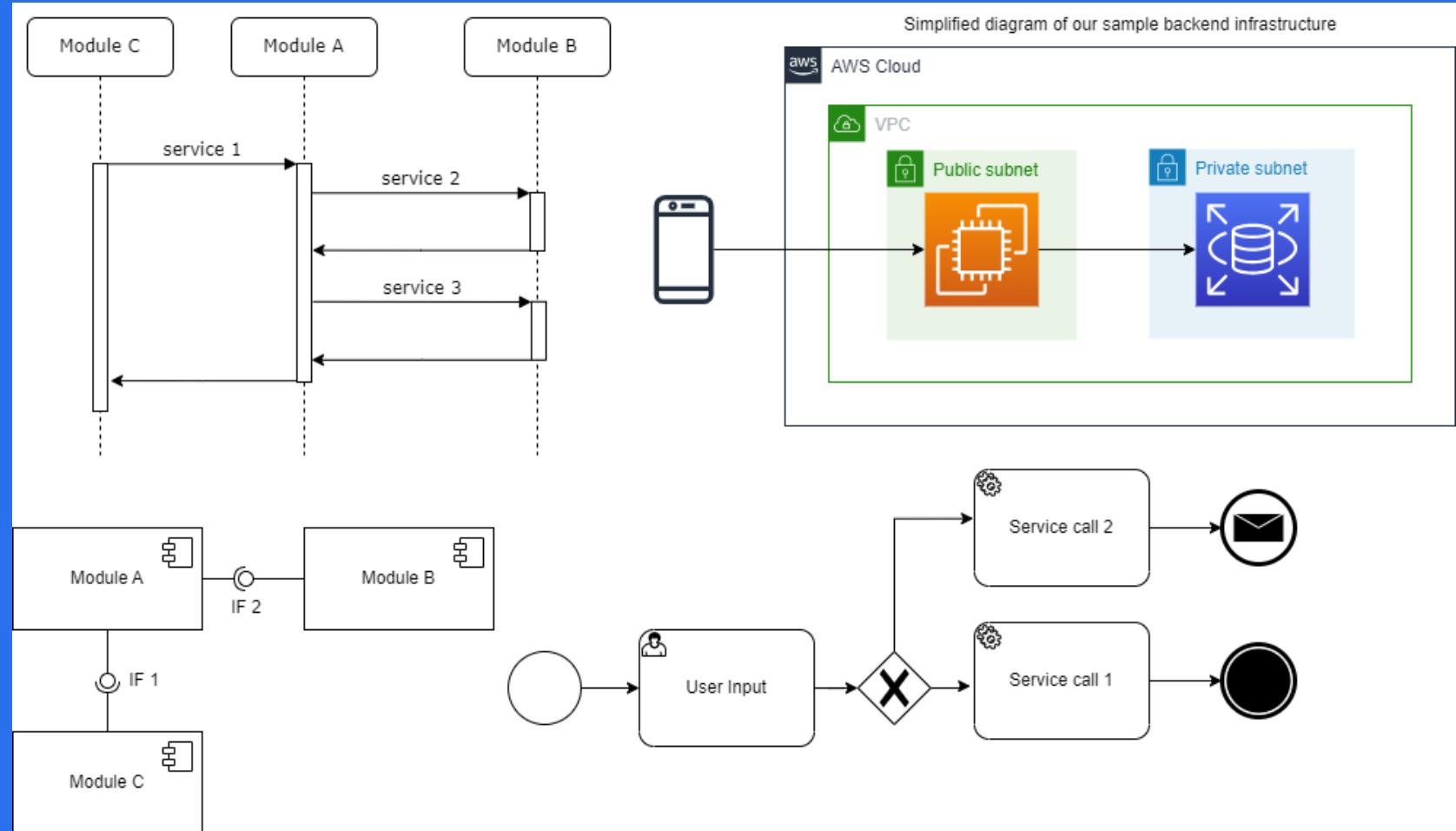


Bottlenecks and Alternatives

- Each requirement affects another requirements
 - Security vs. Performance
 - Robustness vs. Price
 - Reliability/Sustainability vs. Time to delivery
- Each solution has at least 2 more alternatives

Describe and Draw Everything

- UML and diagrams are the common language of every project participant



IT Analysis

Finding the solution

- Business Requirements Analysis
- Solution Design



BA



ARCH



IT
ANA



PM



BE
DEV



FE
DEV



TEST

Artefacts

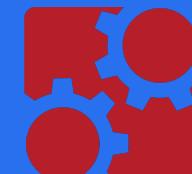
- Impact Analysis



- Functional Specifications



BPMN
Business Process Model and Notation



Quiz Server 1.0.2 OAS 3.0

Quiz Server API - DEMO.

Servers
[https://ec2-3-76-40-32.eu-central-1.compute.amazonaws.com:\(port\) - The Quiz API server](https://ec2-3-76-40-32.eu-central-1.compute.amazonaws.com:(port) - The Quiz API server)

Computed URL: <https://ec2-3-76-40-32.eu-central-1.compute.amazonaws.com:8443>

Server variables
port 8443 v

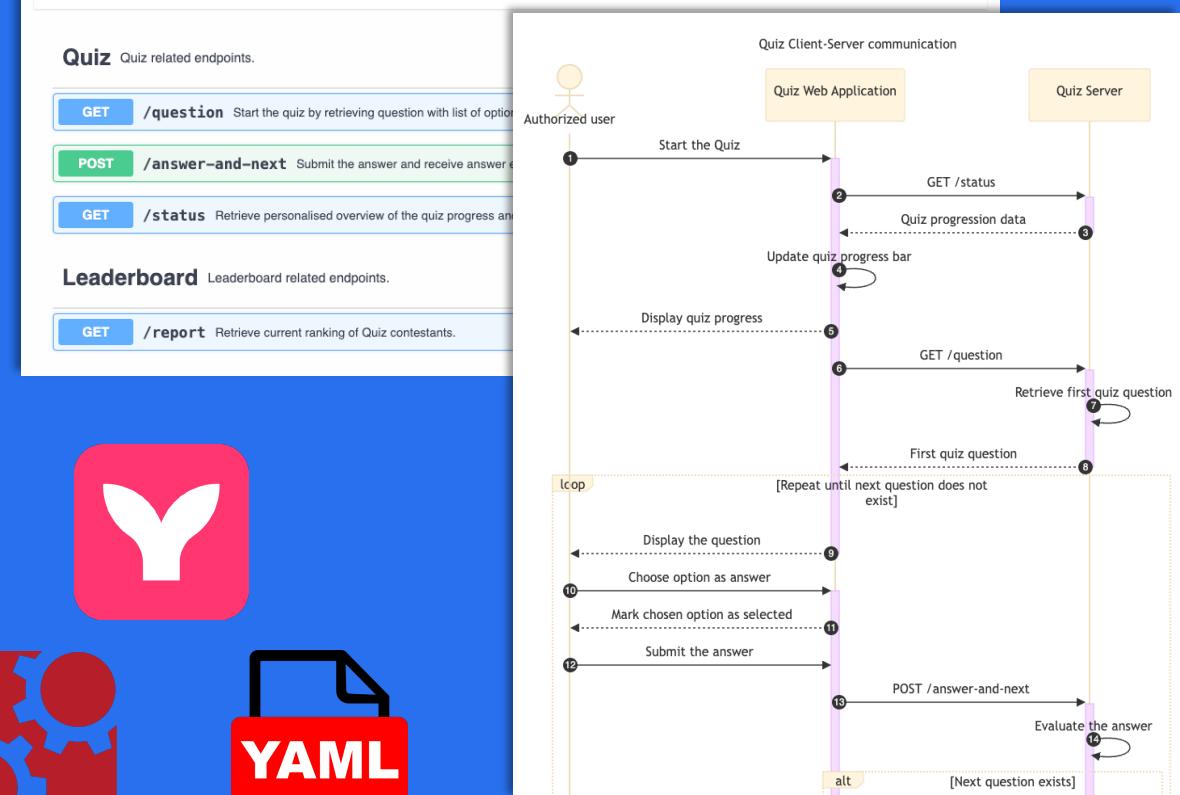
Authorize

Quiz Quiz related endpoints.

- GET /question Start the quiz by retrieving question with list of options.
- POST /answer-and-next Submit the answer and receive answer evaluation.
- GET /status Retrieve personalised overview of the quiz progress and ranking.

Leaderboard Leaderboard related endpoints.

- GET /report Retrieve current ranking of Quiz contestants.

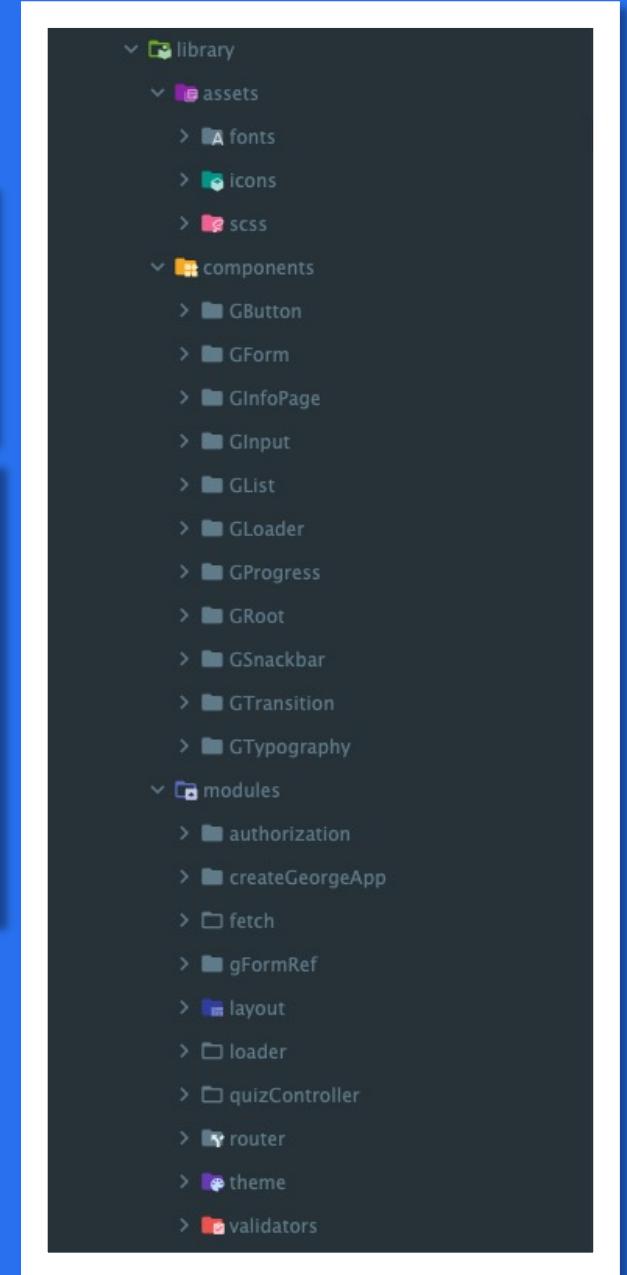
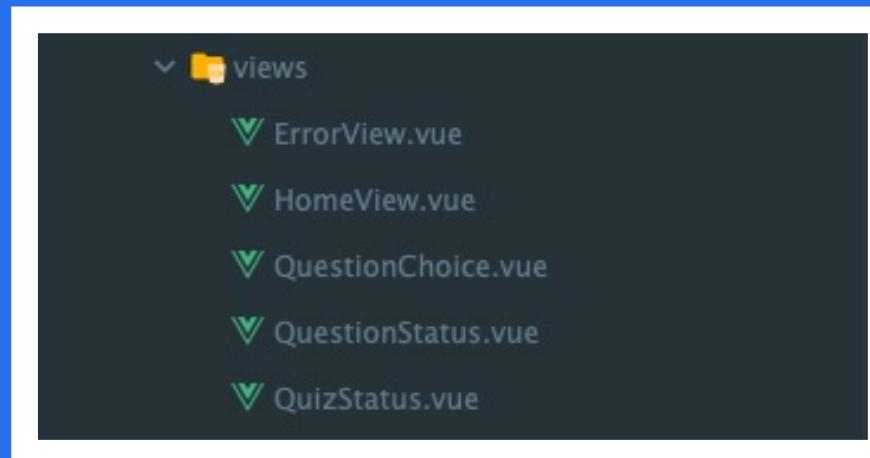
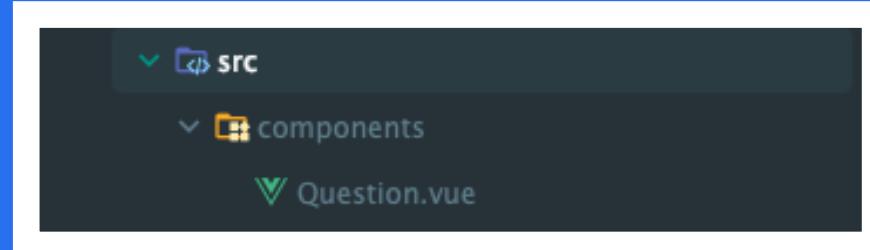
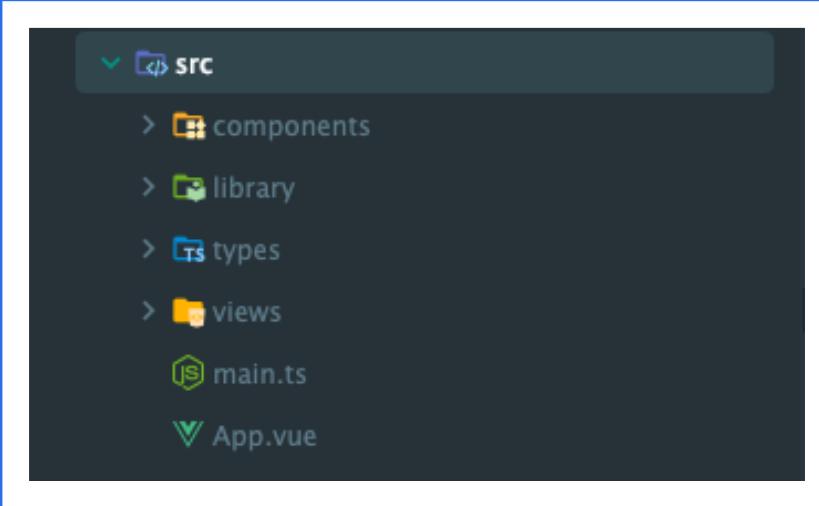


Frontend Development

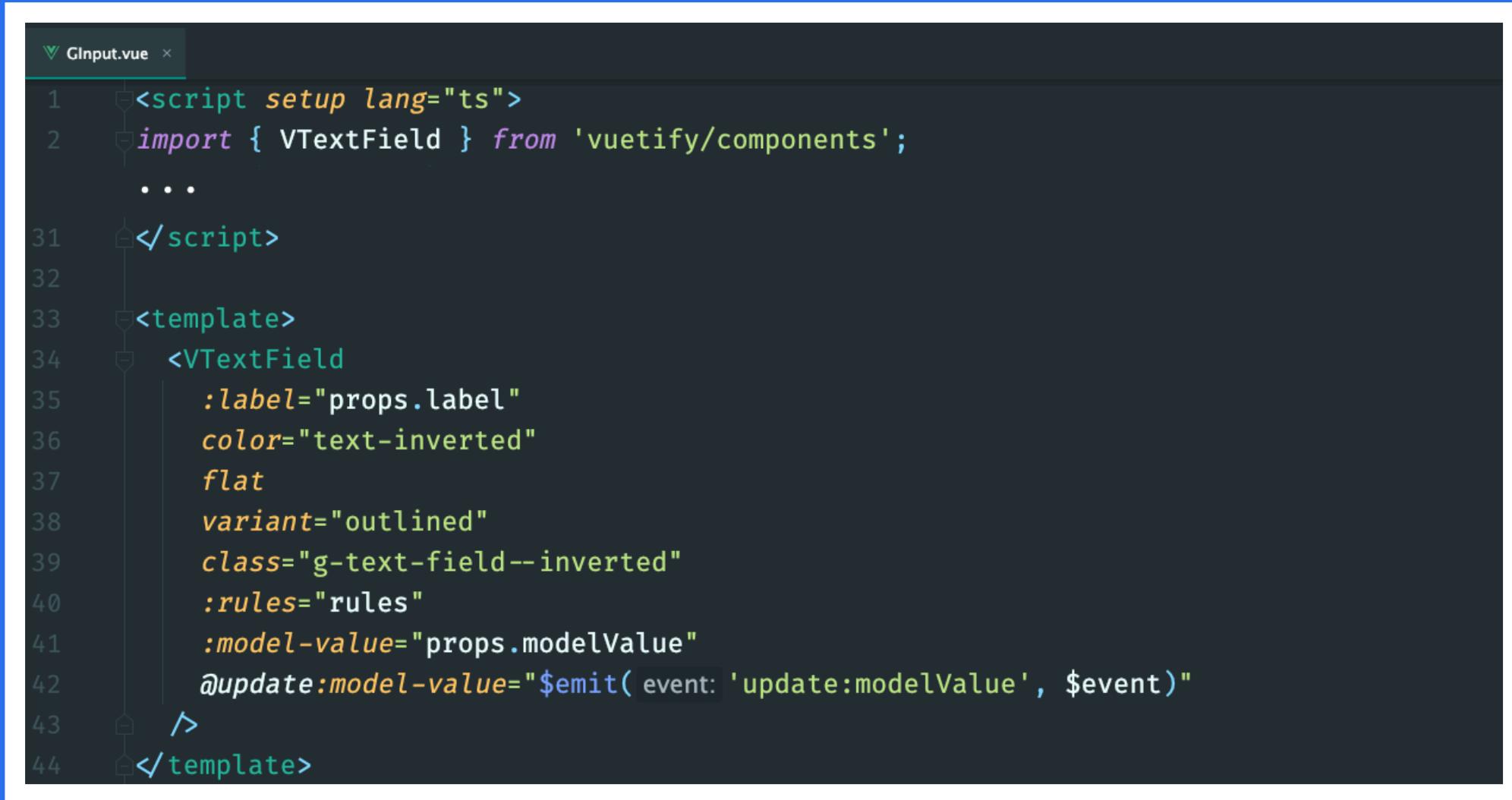
Idea behind our Frontends

- Vue.js + Typescript
- Headless FE managed by BE
- Abstraction
- Modularization
- Composability

File structure



Abstraction



```
GInput.vue
1 <script setup lang="ts">
2 import { VTextField } from 'vuetify/components';
3 ...
31 </script>
32
33 <template>
34   <VTextField
35     :label="props.label"
36     color="text-inverted"
37     flat
38     variant="outlined"
39     class="g-text-field--inverted"
40     :rules="rules"
41     :model-value="props.modelValue"
42     @update:model-value="$emit( event: 'update:modelValue', $event)"
43   >
44 </template>
```

No impact on APPs when changing 3rd party libraries

The image shows a code editor interface with two files open:

- HomeView.vue**:
A template-based component. Lines 35-38 show the structure:

```
35 <template>
36   <GInfoPage title="Georgeov Kvíz">
37     <GText color="text-inverted" class="mb-6">
38       Pre pokračovanie, vyplňte prosím nasledujúce údaje:
39     </GText>
40   </GInfoPage>
41 </template>
```
- QuestionChoice.vue**:
A script-based component. Lines 41-25 show the code:

```
41 <script setup lang="ts">
42   import { quizController } from '@library/modules/quizController/quizController';
43   const { quizState, answerQuestion } = quizController();
44   ...
45   ...
46   const completeView = (newValue: string) => {
47     answerQuestion( {selectedOptionId}: { selectedOptionId: newValue } );
48   };
49 </script>
```

Modularization

The screenshot shows a code editor with three tabs open, illustrating modularization:

- QuestionChoice.vue**:

```
13  export const authorization = () : {...} => {
14    const { fetch } = fetchFactory();
15
16    const setAuthToken = (token: string) => {
17      const loader = createLoader();
18
19      const showLoader = () => {
20        loader.show();
21      };
22
23      const hideLoader = () => {
24        loader.hide();
25      };
26
27      return {
28        showLoader,
29        hideLoader,
30        loaderState: readonly(loaderState),
31      };
32    };
33
34    return {
35      authorization,
36      setAuthToken,
37      showLoader,
38      hideLoader,
39      loaderState: readonly(loaderState),
40    };
41  };
42
43  
```
- authorization.ts**:

```
13  export const authorization = () : {...} => {
14    const { fetch } = fetchFactory();
15
16    const setAuthToken = (token: string) => {
17      const loader = createLoader();
18
19      const showLoader = () => {
20        loader.show();
21      };
22
23      const hideLoader = () => {
24        loader.hide();
25      };
26
27      return {
28        showLoader,
29        hideLoader,
30        loaderState: readonly(loaderState),
31      };
32    };
33
34    return {
35      authorization,
36      setAuthToken,
37      showLoader,
38      hideLoader,
39      loaderState: readonly(loaderState),
40    };
41  };
42
43  
```
- layout.ts**:

```
14
15  const state = reactive<LayoutState>({variant: 'default' ... });
16
17  export const layout = () => {
18    const setLayoutVariant = (variant: LayoutVariant): void => { ... };
19
20    return {
21      setLayoutVariant,
22      layoutState: readonly(state),
23    };
24  };
25
26  
```

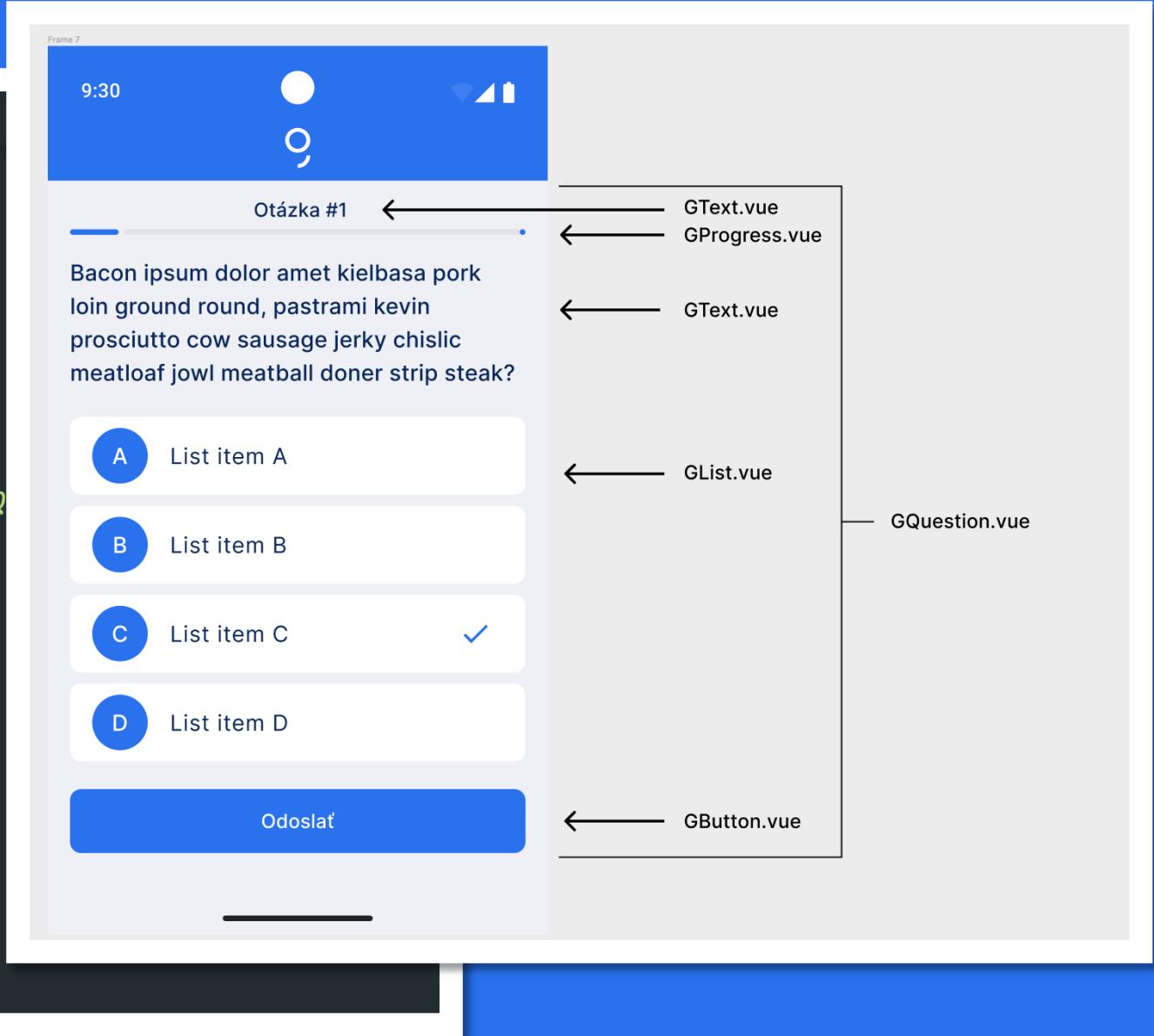
Composability

```

quizController.ts x

45  export const quizController = () :{...} => {
46    const { protectedFetch } = fetchFactory();
47    const { showLoader, hideLoader } = loader();
48    const { push } = router;
49
50
51    const starQuiz = async () => {
52      showLoader();
53
54      const response = await protectedFetch<QuizQ
55        url: '/question',
56        method: 'get',
57      );
58
59      await quizStatus();
60
61      state.questionData = response.data;
62      await push(Route.QuestionChoice);
63      hideLoader();
64
65    };
66
67    ...
68
69  };
70
71  ...
72
73  ...
74
75  ...
76
77  ...
78
79  ...
80
81  ...
82
83  ...
84
85  ...
86
87  ...
88
89  ...
90
91  ...
92
93  ...
94
95  ...
96
97  ...
98
99  ...
100 ...
101 ...
102 ...
103 ...
104 ...
105 ...
106 ...
107 ...
108 ...
109 ...
110 ...
111 ...
112 ...
113 ...
114 ...
115 ...
116 ...
117 ...
118 ...
119 ...
120 ...
121 ...
122 ...
123 ...
124 ...

```



  Classes,  Factory functions

```
function createCounter(): {delayedIncrement: () => void, increment: () => void} {
    let count: number = 0;

    function increment(): void {
        count++;
    }

    function delayedIncrement(): void {
        setTimeout(() => {
            increment();
        }, 1000);
    }

    return {
        increment,
        delayedIncrement
    };
}

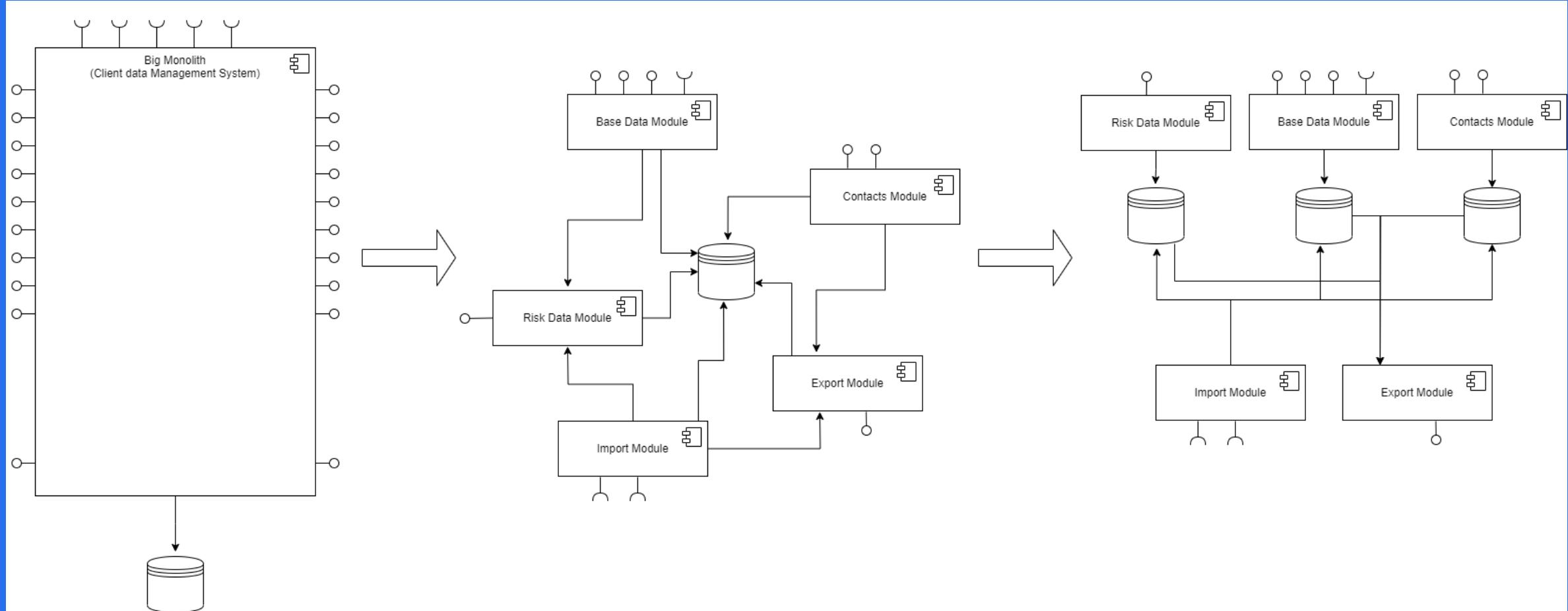
const { delayedIncrement } = createCounter();
delayedIncrement();
```

Backend Development

Modern App in a Large Modern Bank

- From (Distributed) Monolith to Microservices
- Domain Driven Design & Hexagonal Architecture
- Technology Stack that Helps Us
- Modern App is Never Done

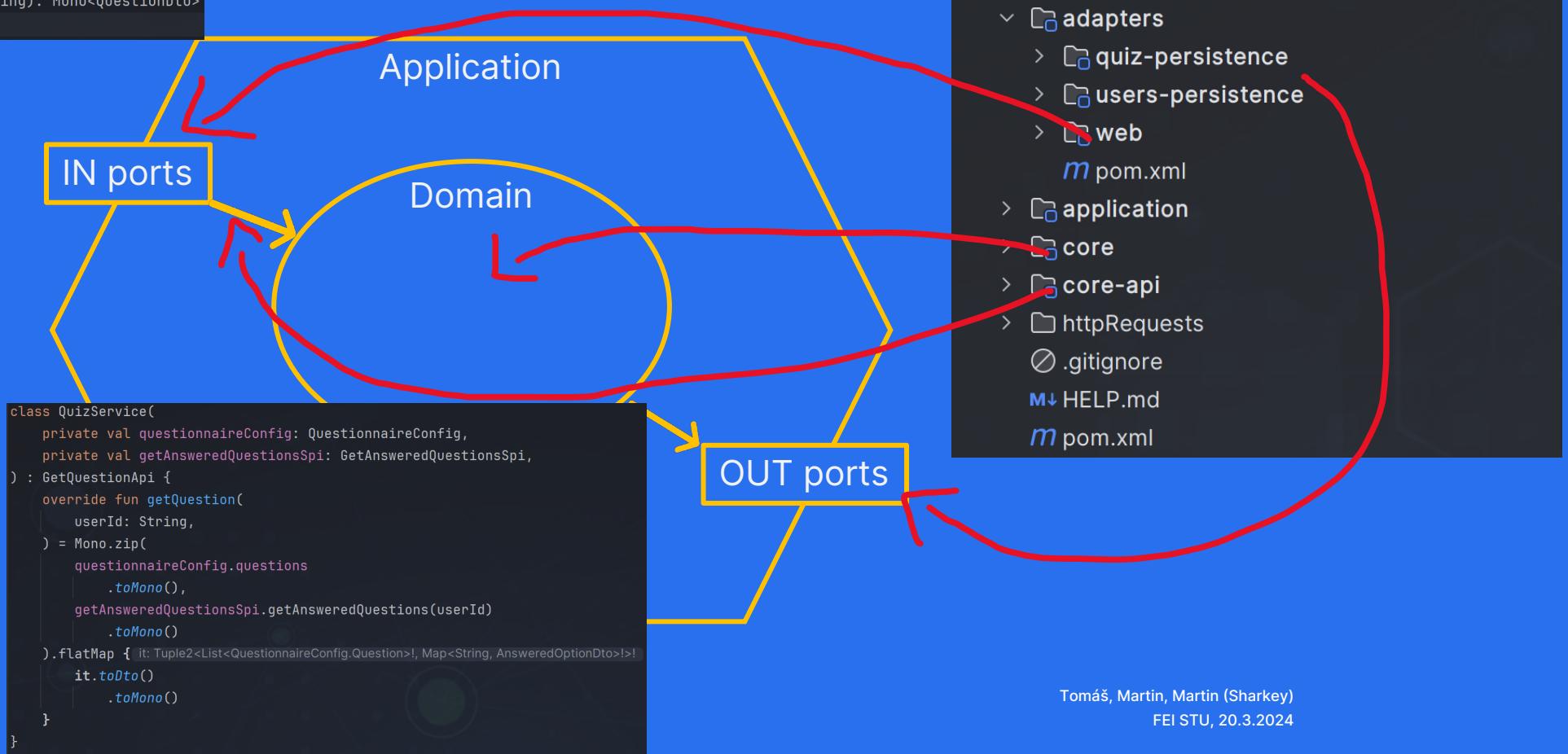
From (Distributed) Monolith to Microservices



Domain Driven Design & Hexagonal Architecture

- Application domain is the bridge between code and business requirement
- Domain layer knows nothing about in/out ports

```
fun interface GetQuestionApi {
    fun getQuestion(userId: String): Mono<QuestionDto>
}
```

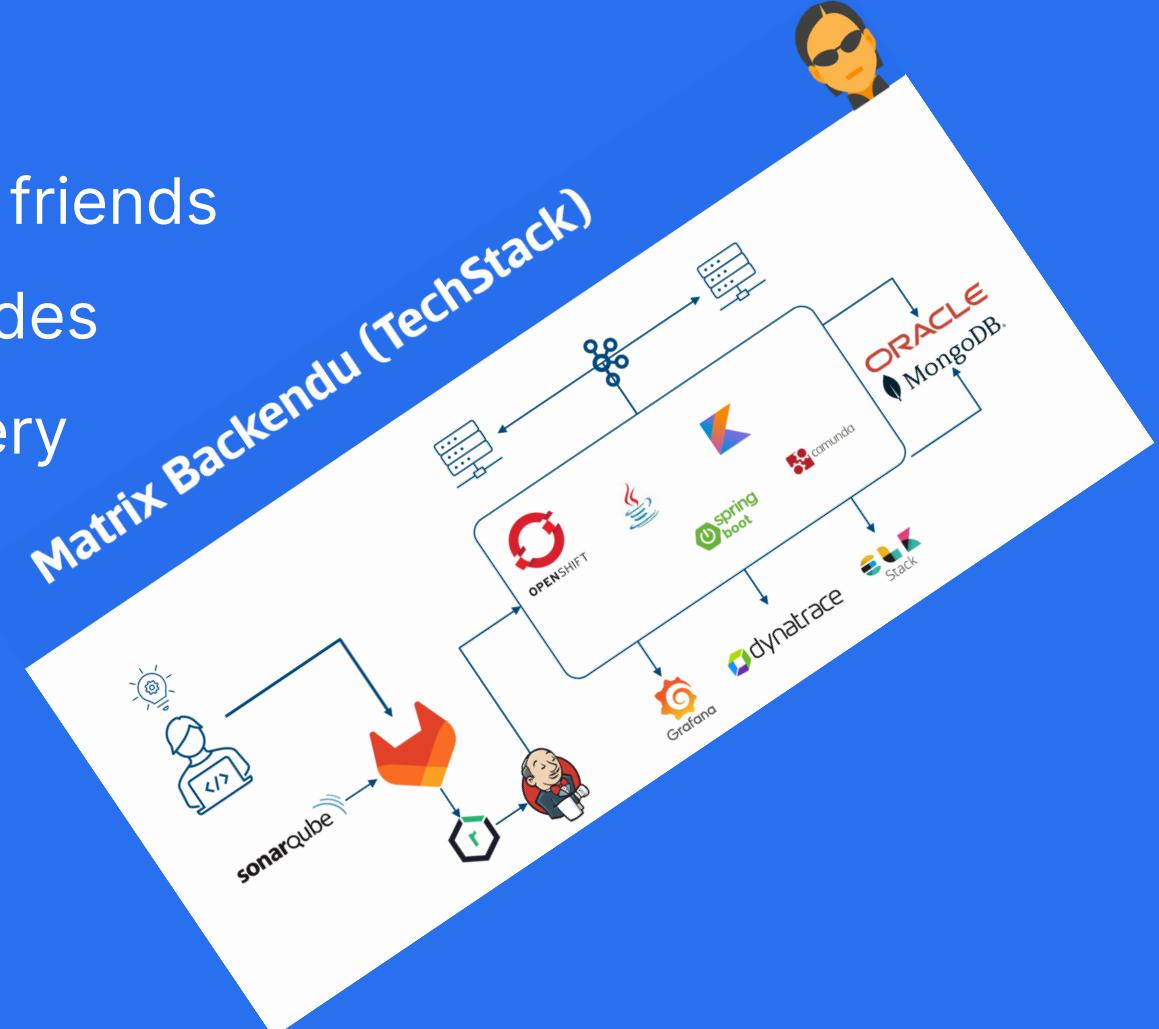


Technology Stack that Helps Us

- Java & Kotlin
- Spring boot
- Reactive programming
- BPMN engine when is needed
- Avoiding untrustworthy 3rd party libs and technologies

Modern App is Never Done

- DevOps engineers are developers best friends
 - Continuous improvements and upgrades
 - Automation and continuous delivery
- Development is not just writing code
 - Monitoring
 - Observability
 - Reporting, Audit,



SLOVENSKÁ Š
sporiteľňa

Thank you!



Tomáš, Martin, Martin (Sharkey)