

tuProlog Web IDE

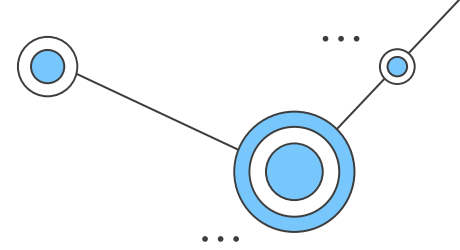
[Intelligent System Engineering]

Aa. 21/22

Alberto Donati - alberto.donati6@studio.unibo.it

Fabio Muratori - fabio.muratori2@studio.unibo.it

Introduction



tuProlog is a language and environment designed for logic programming

- Knowledge representation
- Reasoning
- Rule-based programming for artificial intelligence applications

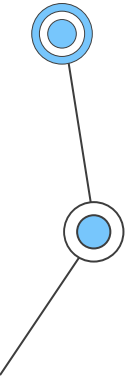
These are essential components of numerous AI applications

We developed a web-based IDE for the tuProlog platform

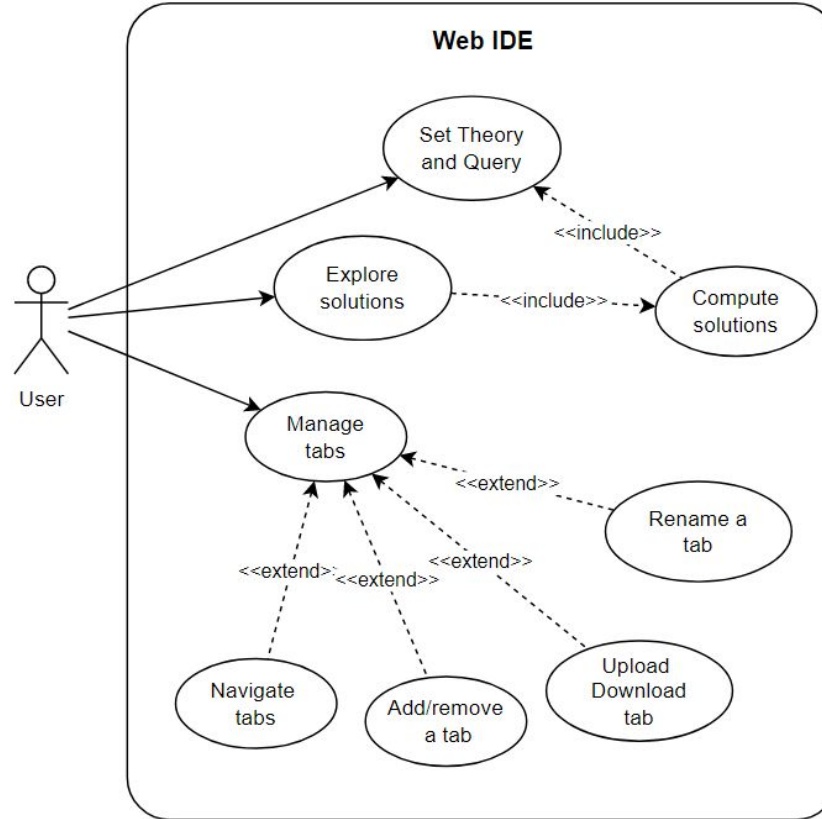
The project

- Builds upon an existing implementation of tuProlog written in Kotlin
- Must leverages web native technologies and best practices

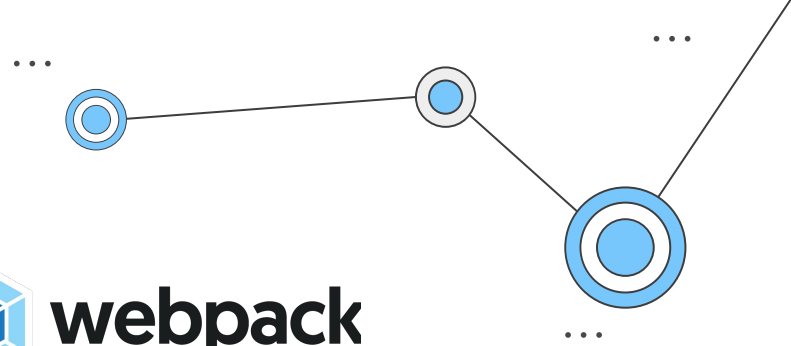
The integration of Kotlin and web-based frameworks is a focal point of our solution



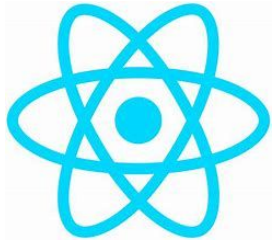
Introduction - Use cases



Introduction – Technologies used



webpack



Introduction – Kotlin

Kotlin

- Open source language
- Based on the JVM
- General purpose
- Multi paradigm

Kotlin Multiplatform

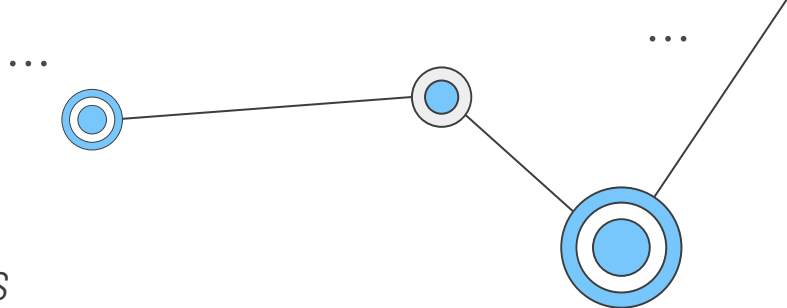
- Technology for cross-platform projects
- Modular architecture with shared and platform-specific modules
- Interoperability with platform-specific code and APIs
- Code reuse across platforms

Kotlin/JS

- Kotlin-to-JavaScript transpilation
 - Generate code for NodeJS platforms or web browsers
- Interoperability with Javascript
 - Allows the usage of existing Javascript modules

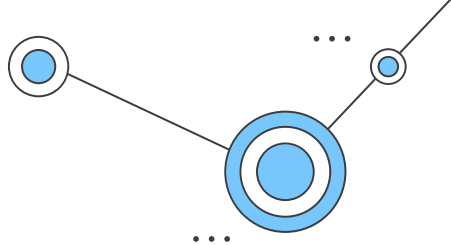
2p-kt

- A Kotlin-based and multi-platform reboot of TuProlog
- Collection of Gradle projects
 - Some of which are Kotlin Multiplatform and Kotlin/JS



We developed a new Kotlin/JS module inside the 2p-kt project

Design choices – Development

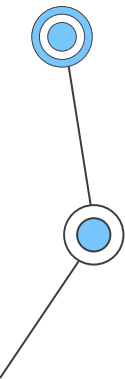


Different options to link our project to the current 2p-kt repositories

- Use 2p-kt as a npm dependency and develop a full Javascript application
 - Porting of Kotlin code to Javascript (through the Kotlin/JS Transpiler) is not easily understandable
- Use 2p-kt as a Gradle dependency and develop a Kotlin application
 - The available implementation does not have a set of new features we want to use
- Expand 2p-kt project with a new module and internally use other project modules
 - Allows developers to use a robust implementation of the MVC pattern where Model and Controller are already implemented

We decided to follow the third option because it allowed us to

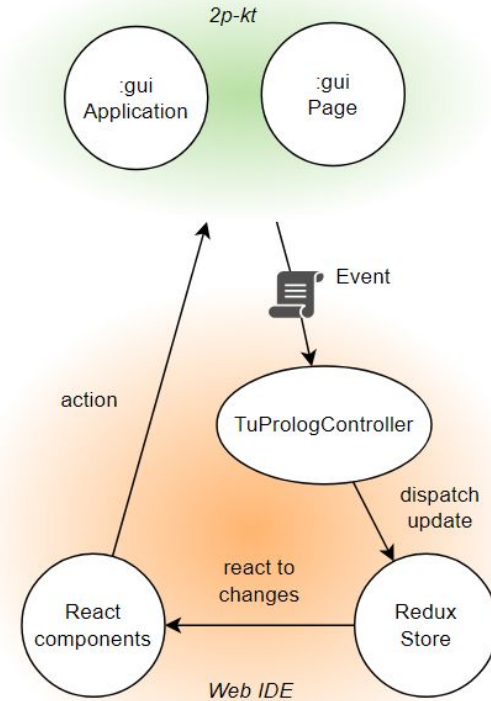
- Access the latest developments of the 2p-kt project
- Mainly focus on View components and how to integrate Kotlin with web based frameworks
- Follow best practices and patterns of modern web technologies



Design choices – Components 1/2

The main components of 2p-kt we interacted with are

- Application - represents a container of Page objects and manages their lifecycle (creation, deletion, etc.)
- Page - a single tuProlog theory and query solver and contains other useful features such as execution
- Event - represents a change in the Page/Application state and these events must be handled by custom listeners

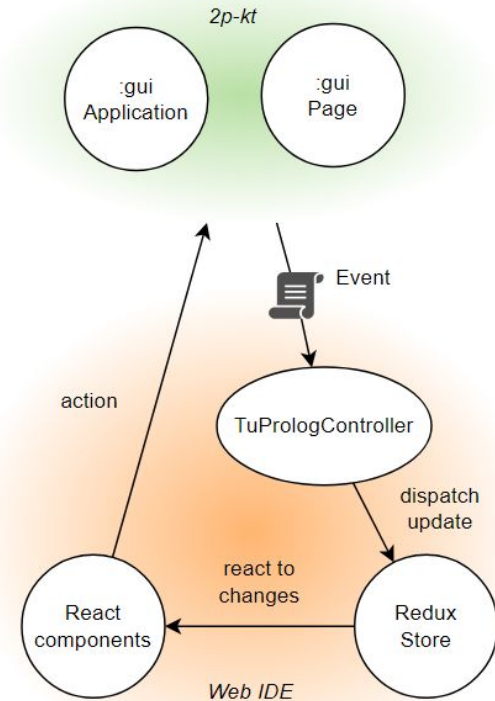


Design choices – Components 2/2

Our solution consists of

- React components - the structure and the representation layer of our web application
- TuPrologController - handle every event generated by the 2p-kt implementation and converts it into Redux state changes
- ReduxStore - the state of the application
 - When the state is modified, graphical components are automatically updated

This architecture allows us to separate application state, view representation and 2p-kt specific functionalities

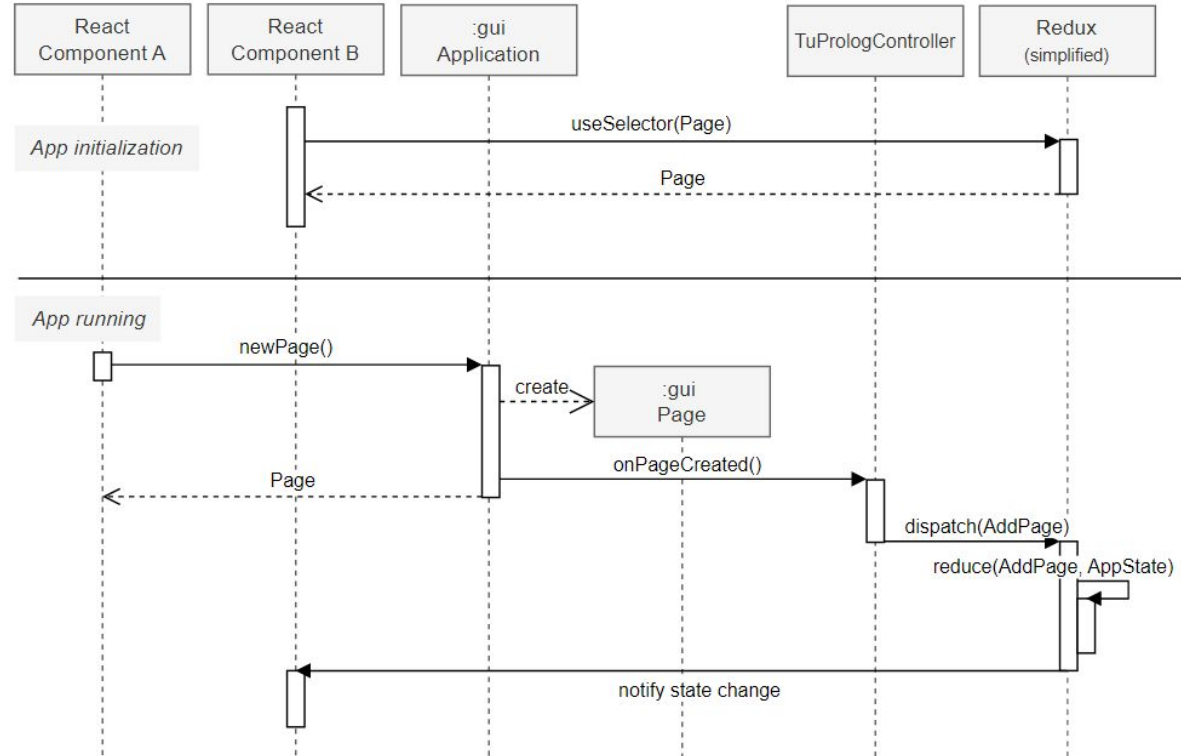


Design choices – Interactions

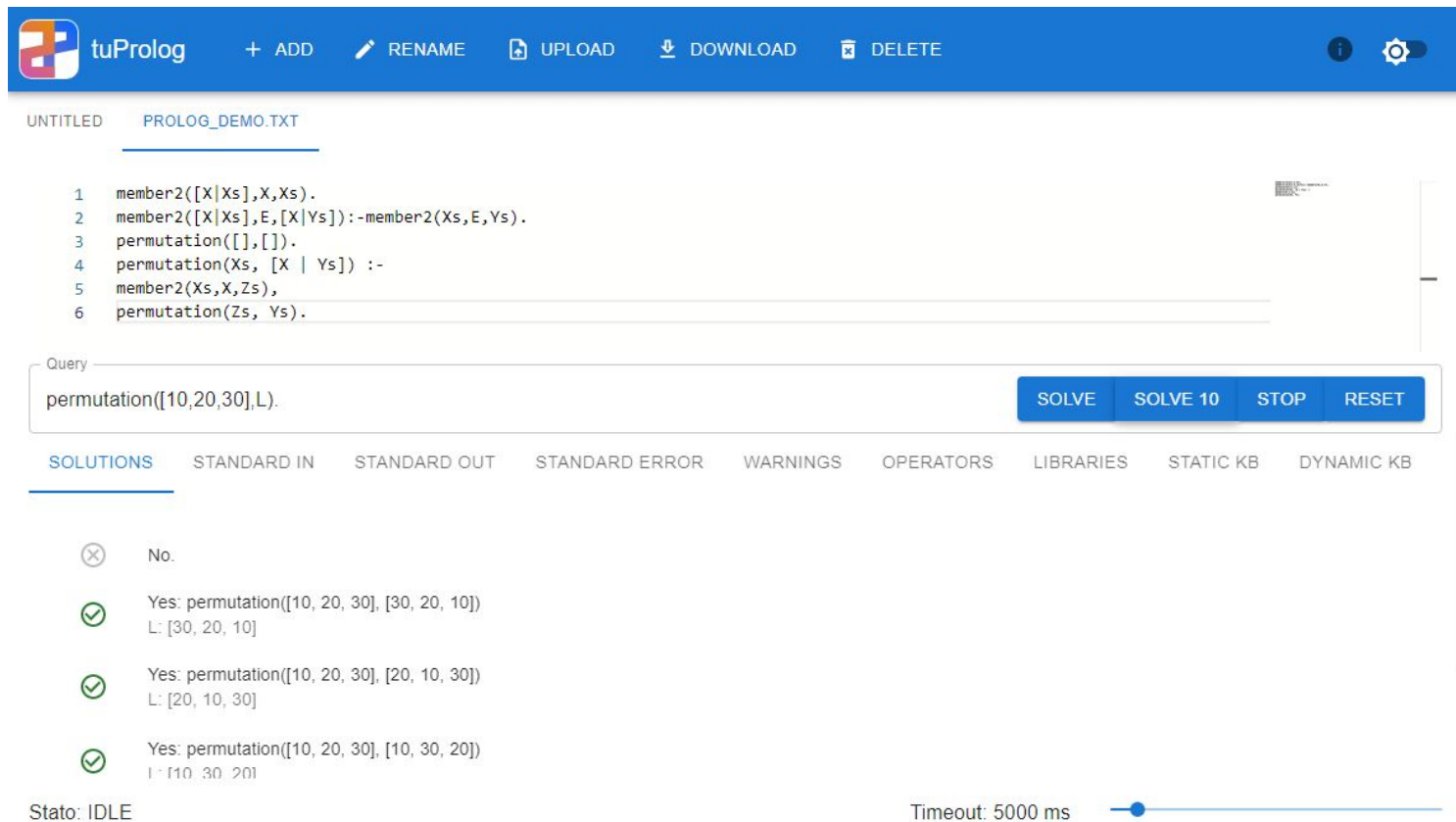
Initially, each React component that requires access to the application state has to register itself to the Redux state manager

The TuPrologController handles 2p-kt updates and translate them to Redux state updates

When the state of the 2p-kt application changes, the new state is propagated by the Redux state manager to every subscribed component



Usage example



The image shows the tuProlog web interface. At the top is a blue header bar with the tuProlog logo, a '+ ADD' button, a 'RENAME' button with a pencil icon, an 'UPLOAD' button with a document icon, a 'DOWNLOAD' button with a download icon, and a 'DELETE' button with a trash icon. On the right of the header are an information icon (i) and a settings icon (gear). Below the header, there are two tabs: 'UNTITLED' and 'PROLOG_DEMO.TXT', with the latter being selected. The main area contains a code editor with the following Prolog code:

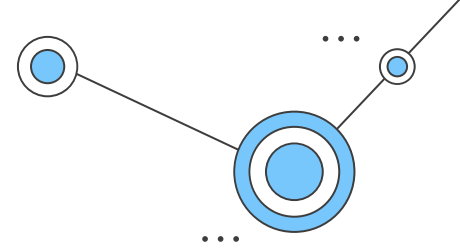
```
1 member2([X|Xs],X,Xs).
2 member2([X|Xs],E,[X|Ys]):-member2(Xs,E,Ys).
3 permutation([],[]).
4 permutation(Xs,[X|Ys]):-
5   member2(Xs,X,Zs),
6   permutation(Zs,Ys).
```

Below the code editor is a 'Query' input field containing 'permutation([10,20,30],L).'. To the right of the input field are four buttons: 'SOLVE', 'SOLVE 10', 'STOP', and 'RESET'. Below the query field is a horizontal menu with the following items: 'SOLUTIONS' (selected), 'STANDARD IN', 'STANDARD OUT', 'STANDARD ERROR', 'WARNINGS', 'OPERATORS', 'LIBRARIES', 'STATIC KB', and 'DYNAMIC KB'. The 'SOLUTIONS' panel shows the following results:

- ⊗ No.
- ✓ Yes: permutation([10, 20, 30], [30, 20, 10])
L: [30, 20, 10]
- ✓ Yes: permutation([10, 20, 30], [20, 10, 30])
L: [20, 10, 30]
- ✓ Yes: permutation([10, 20, 30], [10, 30, 20])
L: [10, 30, 20]

At the bottom left, it says 'Stato: IDLE'. At the bottom right, it says 'Timeout: 5000 ms' with a slider control.

Conclusions



What did we learn

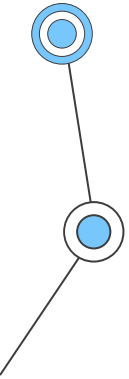
- Gained valuable web development experience using React, Redux, and Kotlin/JS
- Familiarity with the structure of 2p-kt and effective use of its controller
- Ability to create robust and efficient web applications

Future developments

- Implement a tree view feature for visualizing solutions
- Add export functionality for saving solutions in different formats
- Include a debugger for effective issue troubleshooting
- Enhance libraries management in the tuProlog environment
- Enable code linting and decoration for Monaco Editor
- Deploy the application on a real domain

Ideas

- Develop a MAS where each browser page is an autonomous Agent



Thanks!

[Intelligent System Engineering]
Aa. 21/22

Alberto Donati - alberto.donati6@studio.unibo.it
Fabio Muratori - fabio.muratori2@studio.unibo.it

CREDITS: This presentation template was created by [Slidesgo](#), including
icons by [Flaticon](#), infographics & images by [Freepik](#) and illustrations by
[Stories](#)

