Open Universiteit

Creating Interactive Visualizations of TopHat Programs

Mark Gerarts, Marc de Hoog, Nico Naus, Tim Steenvoorden

Task-Oriented Programming

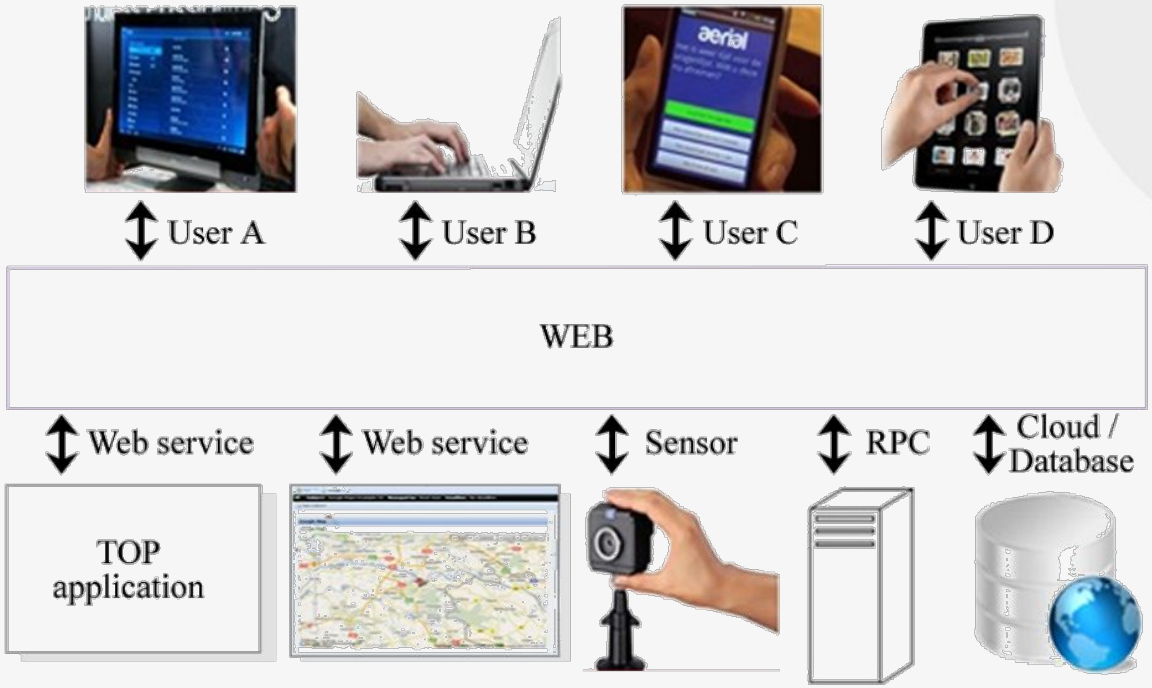
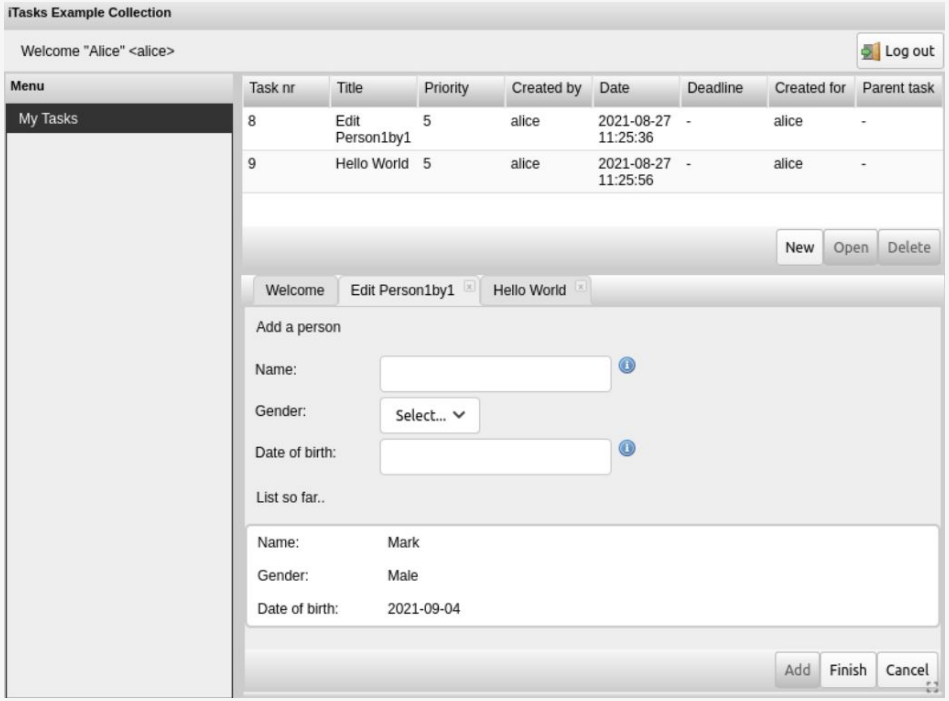


Image source: An Introduction to Task-Oriented Programming, Achten Peter, Pieter Koopman, and Rinus Plasmeijer

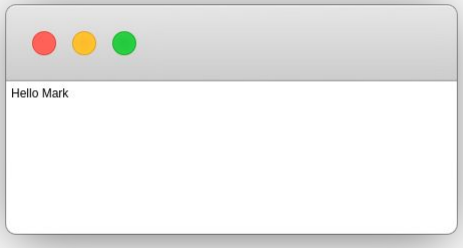
TOP framework

- *What* vs *How*
- Focus on the domain logic
- TOP framework handles all the rest (UI, authentication, …)

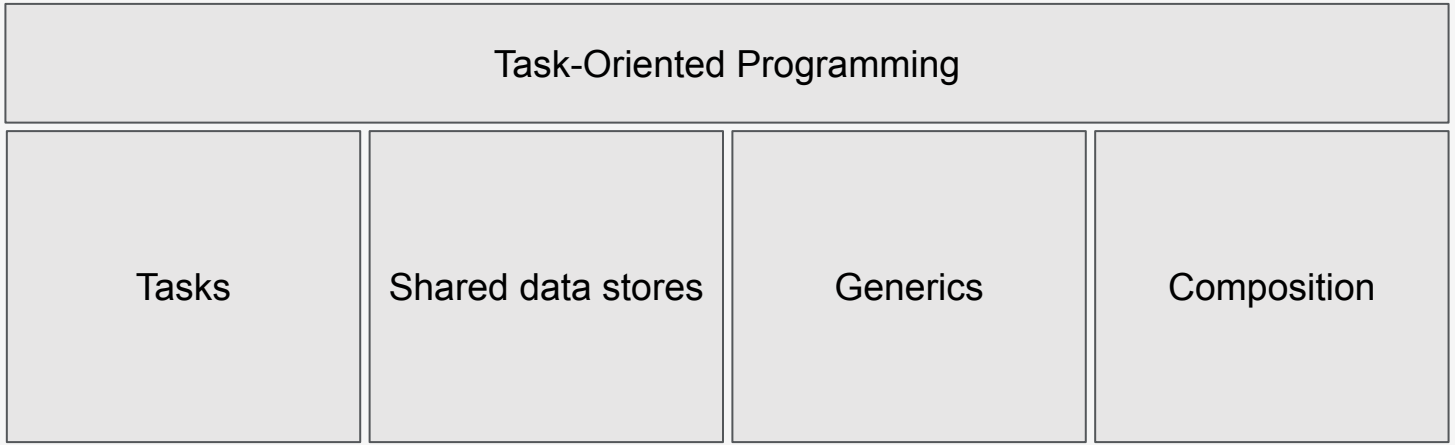


Hello iTasks

```
enterName :: Task String
enterName = Hint "What is your name?" @>> enterInformation []

greet :: Task String
greet =
 enterName >>!
   \result -> viewInformation [] ("Hello " +++ result)
```



Core concepts



Core concepts

Image source: TopHat: A formal foundation for task-oriented programming, Steenvoorden Tim, Nico Naus, and Markus Klinik

Current situation

iTasks

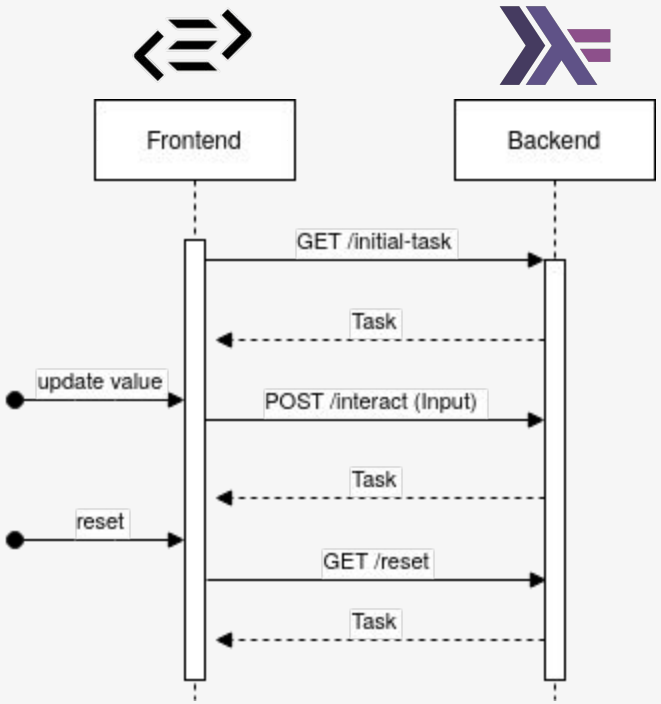
- Written in Clean
- Practical use
- Generates UI
- Lacks formalisation

TopHat

- DSL implemented in Haskell
- Verifying behaviour
- No UI, only CLI
- Formal reasoning

To what extent is it possible to add an interactive web UI to a TopHat program without a translation to iTasks, and without the use of Clean-specific constructs?

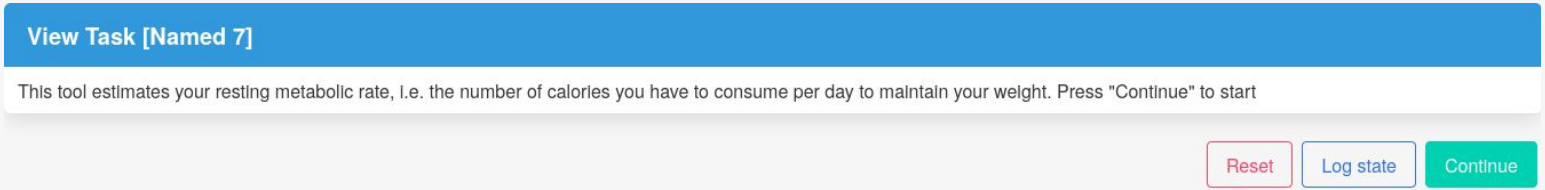
Architecture



Calorie Calculator

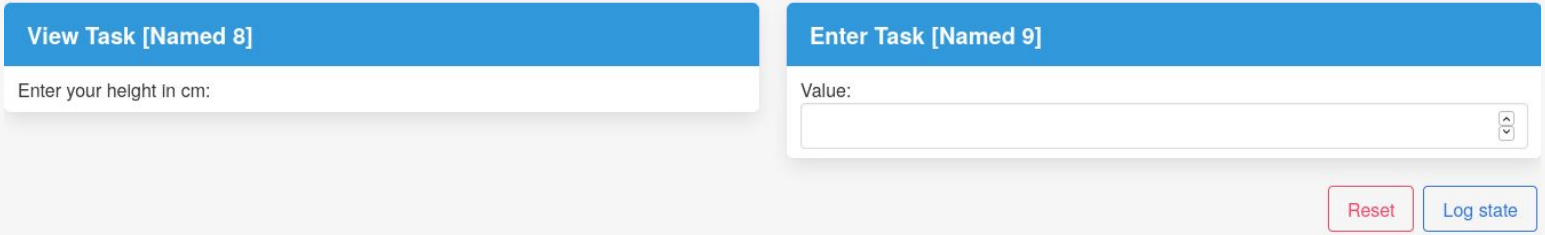
```
calculateCaloriesTask :: Task h Text
calculateCaloriesTask =
 introduction >>? \_ -> do
   (_, height) <- promptHeight
   (_, weight) <- promptWeight
   (_, age) <- promptAge
   (_, gender) <- promptGender
   (_, aLvl) <- promptActivityLevel
   let calories = calculateCalories gender aLvl height weight age
   view ( "Your resting metabolic rate is: "
            <> display calories
            <> " calories per day."
        )
```

Calorie Calculator



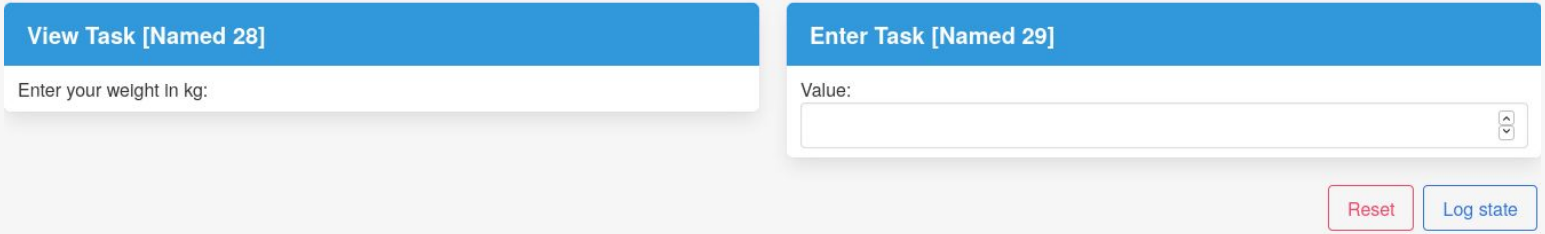

```
introduction :: Task h Text
introduction =
 view
   <| unlines
     [ "This tool estimates your resting metabolic rate,",
       "i.e. the number of calories you have to consume"
       "per day to maintain your weight.",
       "Press \"Continue\" to start"
     ]
```

Calorie Calculator



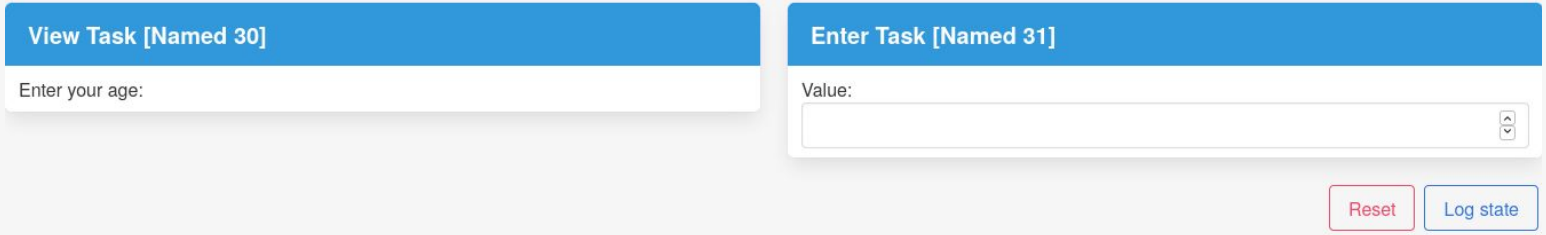

```
type Height = Int

promptHeight :: Task h (Text, Height)
promptHeight = view "Enter your height in cm:" >< enter
```

Calorie Calculator



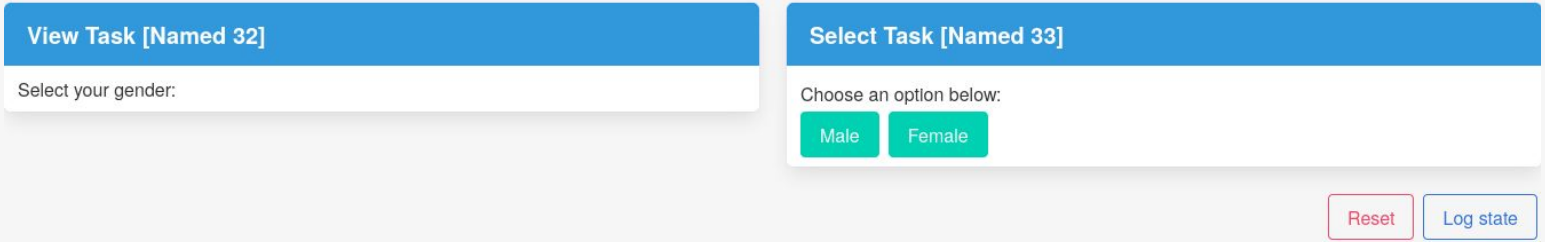

```
type Weight = Int

promptWeight :: Task h (Text, Weight)
promptWeight = view "Enter your weight in cm:" >< enter
```

Calorie Calculator



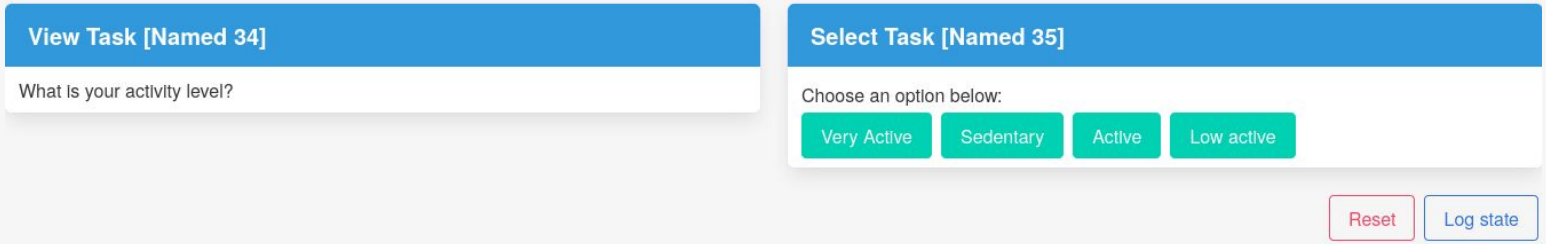

```
type Age = Int

promptAge :: Task h (Text, Age)
promptAge = view "Enter your age:" >< enter
```

Calorie Calculator



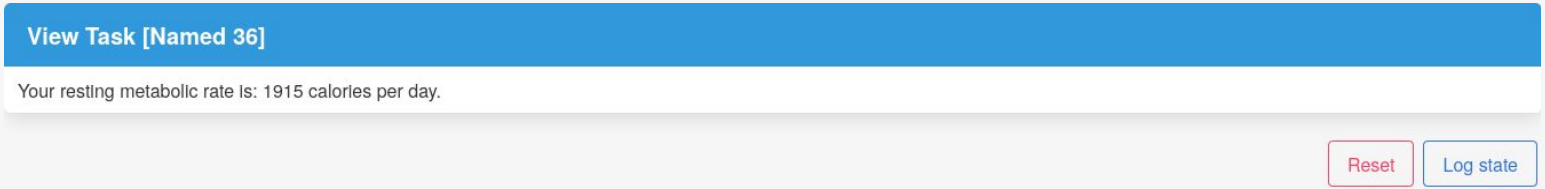

```
data Gender = Male | Female

promptGender :: Task h (Text, Gender)
promptGender =
 view "Select your gender:"
   >< select
   [ "Male" ~> Done Male,
     "Female" ~> Done Female ]
```

Calorie Calculator



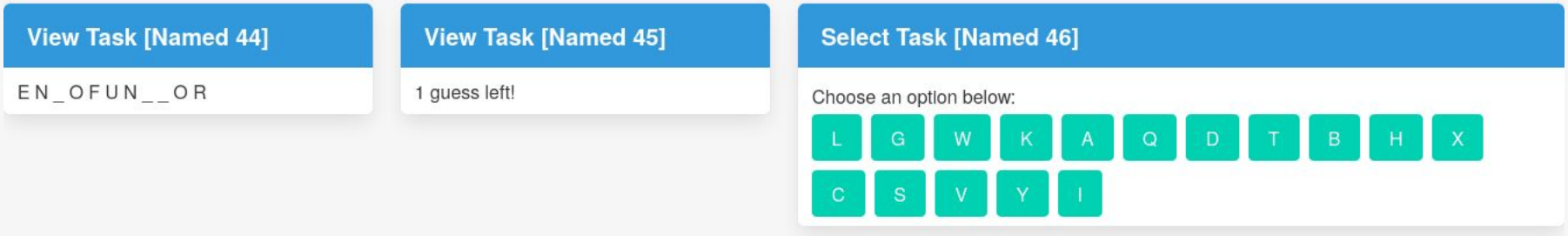

```
promptActivityLevel :: Task h (Text, ActivityLevel)
promptActivityLevel =
 view "What is your activity level?"
   >< select
     [ "Sedentary" ~> Done Sedentary,
       "Low active" ~> Done Low,
       "Active" ~> Done Active,
       "Very Active" ~> Done VeryActive ]
```

Calorie Calculator




```
calculateCaloriesTask :: Task h Text
calculateCaloriesTask = introduction >>? \_ -> do
 (_, height) <- promptHeight
 -- ...
 let calories = calculateCalories gender aLvl height weight age
 view
   ( "Your resting metabolic rate is: "
       <> display calories
       <> " calories per day." )
```

Summary

- It is possible to create an interactive web UI for TopHat programs
- **Limited scope**: data types, TOP framework features
- Future work: more sophisticated data types and more advanced features
- We combine practical applicability with formal reasoning

Hangman in TopHat