# Development of an OCaml API to interact with a Tezos node

Albert-Ludwigs-Universität Freiburg

Tamara Bernhardt, 4743793

Master Project

# Agenda

1

API – specification & signatures

2

Case study: auction

3

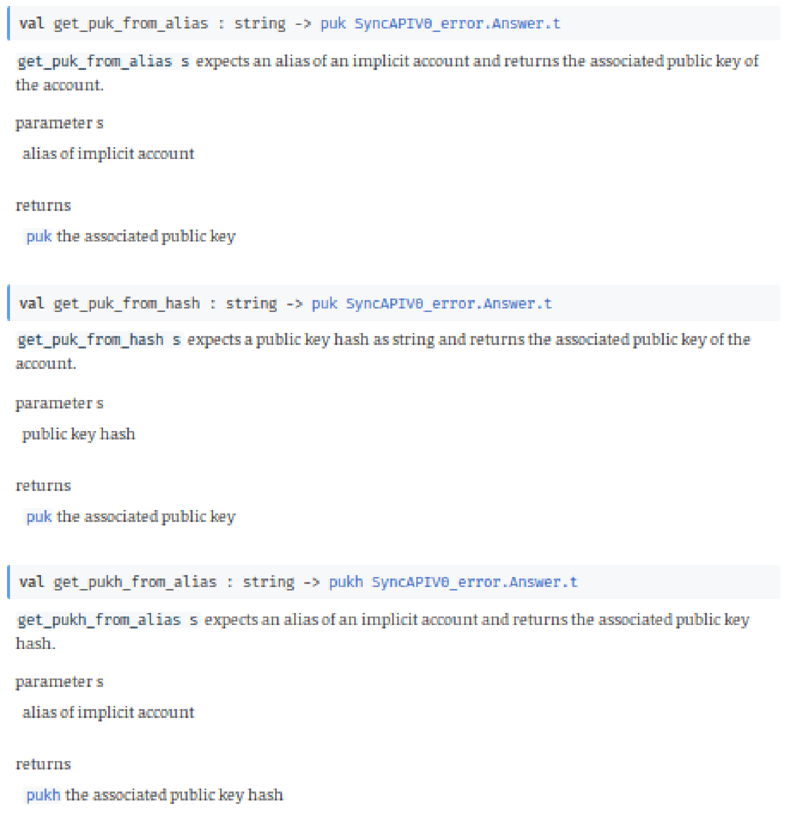
Demo

4

Outlook

# API specification & signatures

# API specification & signatures



```
val get_puk_from_alias : string -> puk SyncAPIV0_error.Answer.t
```

`get_puk_from_alias s` expects an alias of an implicit account and returns the associated public key of the account.

parameter s

alias of implicit account

returns

puk the associated public key

```
val get_puk_from_hash : string -> puk SyncAPIV0_error.Answer.t
```

`get_puk_from_hash s` expects a public key hash as string and returns the associated public key of the account.

parameter s

public key hash

returns

puk the associated public key

```
val get_pukh_from_alias : string -> pukh SyncAPIV0_error.Answer.t
```

`get_pukh_from_alias s` expects an alias of an implicit account and returns the associated public key hash.

parameter s

alias of implicit account

returns

pukh the associated public key hash

HTML Docs

# Case study: auction

1 API – specification & signatures

2 Case study: auction

3 Demo

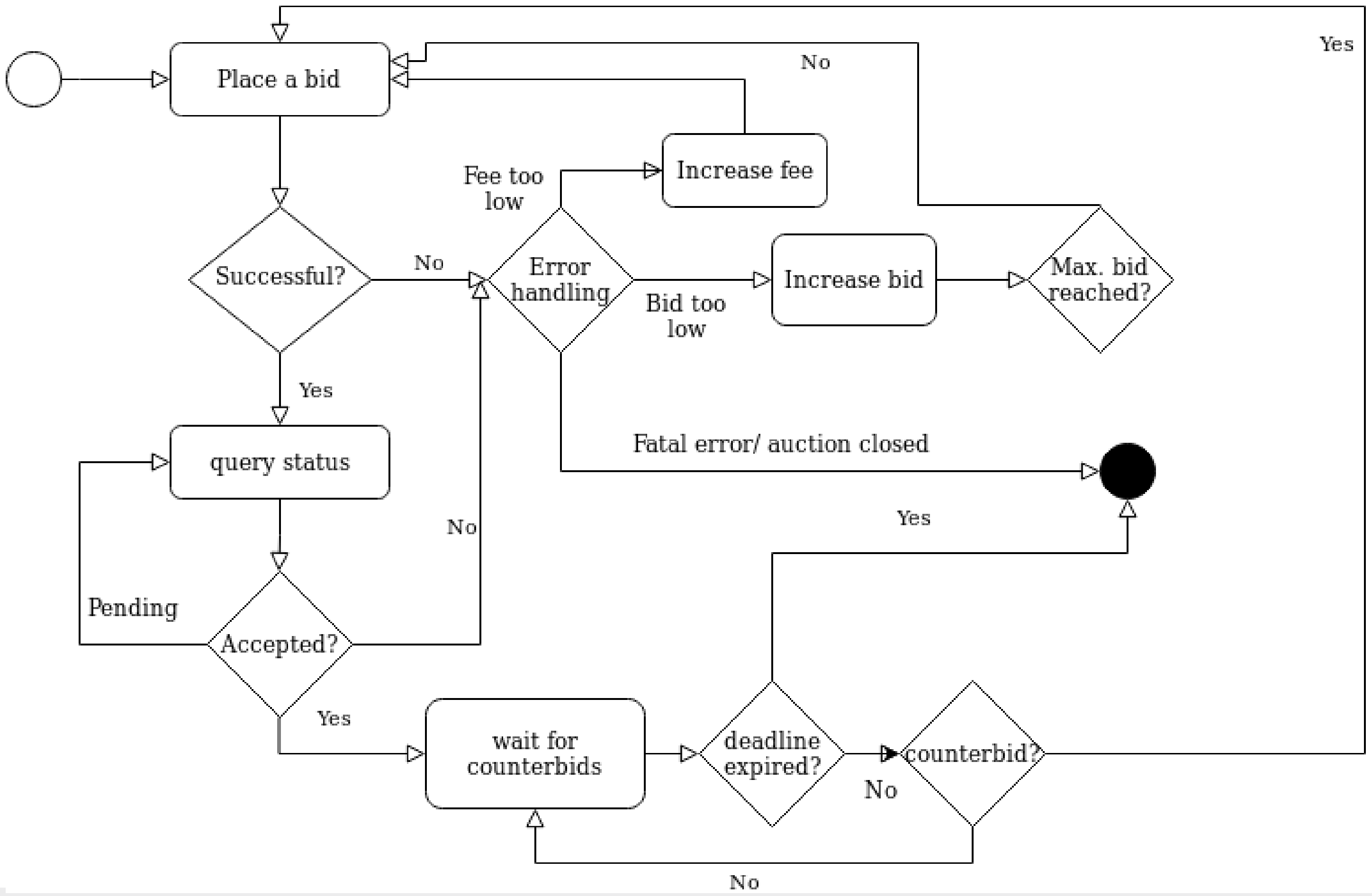
4 Outlook

# Case study: auction

- Uses the auction contract provided
- No auction code amendments yet
- Bidder program:
  - Places bids with increasing stake/fee until succesful (→ call a contract)
  - Stays within stake & fee/burn cap (→error handling)
  - Waits for operation inclusion (→ query)
  - Waits for counterbids (→ balance checking)
  - Terminates due to permanent errors or auction closing (→ error handling)

# Case study: auction

- Program flow:



## Case study: auction

The bidder program

# Demo

1 API – specification & signatures

2 Case study: auction

3 Demo

4 Outlook

## Demo

README

# Outlook

1

API – specification & signatures

2

Case study: auction

3

Demo

4

Outlook

## Outlook

- Versioning/ virtual libraries to handle different protocols
- Observables (time, block level)
- Add more info to the error types
- Make error type extensible? → Add contract specific errors if needed
- Automatic testing (using Tezt?)
- Use the estimated fee from the prevalidator (as option)
- Library packaging/ release management