ПРИЛОЖЕНИЕ В

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ПРОГРАММНАЯ БИБЛИОТЕКА ПОДПРОГРАММ ИДЕНТИФИКАЦИИ ПОЛЬЗОВАТЕЛЕЙ

Фрагмент исходного текста

Листов 5

Студент гр. ИУ6-84Б _____ М.А. Гейне
(Подпись, дата) (И.О.Фамилия)

Руководитель _____ П. В. Аргентов
(Подпись, дата) (И.О.Фамилия)

Москва, 2022

Исходный текст модуля Authenticator (`src/core/authenticator.ml`):

```
(**[Authenticator] is module which provides functions both for authentication and logout*)

open Base
open Dream
open Static

(**[Make] creates an instance of {!Auth_sign.AUTHENTICATOR} for a given model and variables*)
module Make (M : Auth_sign.MODEL) (V : Auth_sign.VARIABLES with
   type entity = M.t ) : (Auth_sign.AUTHENTICATOR with type entity
   = M.t) = struct

  type entity = M.t

  (** [strategy] is a function that tries to authenticate an entity*)
  type strategy = (module Auth_sign.STRATEGY with type entity =
   entity)

  module type Strategy = Auth_sign.STRATEGY with type entity =
   entity

  let set_authenticated request =
    set_field request V.authenticated true;
    request

  (**[auth] is a recursive function for running strategies and verifying*)
  let rec auth (lst : strategy list) request ent : AuthResult.t
   promise =
    match lst with
    | [] -> set_field request V.auth_error (Error.of_string "End
   of strategy list");
            Lwt.return AuthResult.Rescue
    | (module S : Strategy)::strats ->
      match%lwt S.call request ent with
      | Next -> auth strats request ent
      | Authenticated ent ->
        let%lwt () =
        request |> set_authenticated |> V.update_current_user ent
   in
        Lwt.return AuthResult.Authenticated
```

```
      | Rescue err -> set_field request V.auth_error err;
                    Lwt.return AuthResult.Rescue
      | Redirect url -> Lwt.return (AuthResult.Redirect url)


  let name_in_list names (module S : Strategy) =
    List.exists names ~f:(String.equal S.name)

  let filter_strategies (strats: strategy list) names =
    List.filter strats ~f:(name_in_list names)

  (** [authenticate] runs all strategies from the list until one of them succeeds.
 Sets session and field variables. Returns a promise. *)
  let authenticate (lst : strategy list) request =
    match%lwt M.identificate request with
    | Error err ->  set_field request V.auth_error err;
    Lwt.return AuthResult.Rescue
    | Ok ent ->
      let filtered_strats = M.applicable_strats ent |>
   filter_strategies lst in
      auth filtered_strats request ent

  (** [logout] clears [auth] session field and sets {V.authenticated} to [false], making session
    unauthenticated.
 Note: the function does NOT modify {!V.current_user}. It will be set to [None] only for the next request.*)
  let logout request =
    set_field request V.authenticated false;
    request |> invalidate_session
end
```

Исходный текст модуля Password (`src/strategies/password.ml`):

```
open Base
open FPauth_core.Static


let name = "password"
module type MODEL = sig
  (** Some representation of an entity to be authenticated *)
  type t

  (** [encrypted_password] retrieves password of an entity in an encrypted form*)
```

```
  val encrypted_password: t -> string option
end

module Make (M : MODEL) : (FPauth_core.Auth_sign.STRATEGY with type
    entity = M.t) = struct

  open StratResult
  open StratResult.Infix

  type entity = M.t
  let get_password request =
    match Params.get_param_req "password" request with
    | None -> Dream.log "\'password\' is needed for Password
   authentication, skipping the strategy..."; Next
    | Some password -> Authenticated password

  let check_password user password =
    match M.encrypted_password user with
    | None -> Rescue (Error.of_string "No encrypted password for
   the user")
    | Some encoded ->
      match Argon2.verify ~encoded ~pwd:password ~kind:Argon2.ID
   with
      | Error Argon2.ErrorCodes.VERIFY_MISMATCH -> Rescue (Error.
   of_string "Incorrect password!")
      | Error err ->  Rescue (Error.of_string (Argon2.ErrorCodes.
   message err))
      | Ok _ -> Authenticated user


  (**[call] is a main function of a strategy, which authenticates the user by "login" and "password" querry
   params*)
  let call request user =
    get_password request >>== check_password user |> Lwt.return

  let routes = Dream.no_route

  (* takes name from outside the functor*)
  let name = name
end
```

Исходный код всего проекта доступен в открытом репозитории кода на GitHub: `https://github.com/mikeGEINE/FPauth`.