

Contents

1	Module Server : Server-side from the iSketch game	1
---	---	---

1 Module Server : Server-side from the iSketch game

```
val max_players : int Pervasives.ref
    Maximum players capacity

val timeout : int Pervasives.ref
    Timeout when a word is found

val port : int Pervasives.ref
    Server port number

val cheat_parameter : int Pervasives.ref
    Number of players needed to report a cheating behavior

val verbose_mode : bool Pervasives.ref
    Verbose mode enabled or not

val dictionary_filename : string Pervasives.ref
    Dictionary filename

val dictionary_words : string list Pervasives.ref
    List of words from the dictionary

val logfile : string
    Log file

val registered_players : string
    Single file which stands for database

val word : string Pervasives.ref
    Word to find at each round

val word_found : bool Pervasives.ref
    Says whether or not the word has been found

val word_finders : int Pervasives.ref
    Backs up the number of people who have found the word at each round

type role =
    | DRAWER
```

The drawer needs to draw the word that is chosen at each round

| FINDER

A finder has to find the word the drawer is trying to draw

At each round, each player is either a drawer or a finder

val players :

```
< get_name : unit -> string; get_role : unit -> role;  
  get_score_game : unit -> int; get_score_round : unit -> int;  
  get_status : unit -> bool; send_command : string -> unit;  
  set_role : role -> unit; set_score_round : int -> unit;  
  start_game : unit -> unit; update_score_game : unit ->
```

list Pervasives.ref

List of all the players

val mutex_players : Mutex.t

Mutex to prevent multiple players to connect/register/login simultaneously

val mutex_maximum_players : Mutex.t

val condition_players : Condition.t

val condition_end_round : Condition.t

val mutex_end_round : Mutex.t

val mutex_guessed_word : Mutex.t

Mutex to prevent multiple players to guess a word simultaneously

val players_connected : int Pervasives.ref

Gives the number of connected players

val round : int Pervasives.ref

Number of the current round

val rgb : string Pervasives.ref

The string that symbolizes the current color used by the drawer

val size : string Pervasives.ref

The string that symbolizes the current size used by the drawer

val line : string Pervasives.ref

The string that symbolizes the current line drawn by the drawer

val score_round_finder : int Pervasives.ref

val score_round_drawer : int Pervasives.ref

val timeout_on : bool Pervasives.ref

Says whether or not the timeout is on

```

val round_canceled : bool Pervasives.ref
    A round is canceled if the drawer decides to pass or if a cheating behavior is reported

val cheat_counter : int Pervasives.ref
    Counts how many players have reported cheating behavior

val thread_timeout : Thread.t Pervasives.ref
val trace : string -> unit
    trace message writes the message in the log file

val notify_timeout : unit -> unit
    Sends the WORD_FOUND_TIMEOUT command to all players

val update_variables : unit -> unit
val init_variables : unit -> unit
val remove : 'a -> 'a list -> 'a list
    remove e l removes the element e from the list l

val read_file : Pervasives.in_channel -> string list
val init_dict : unit -> string list
val unescaped : string -> string
val escaped : string -> string
val exists_in_db : string -> bool
    exists_in_db user returns true if user is already registered in the database

val is_ok : string -> string -> bool
    is_ok user password returns true if user and password is a correct couple registered in the
    database

val gen_salt : int -> string
    gen_salt n generates a salt of size n

val exists : string -> bool
    exists user returns true if user is already connected that is to say is in the players list

val generate_name : string -> string
    generate_name name returns a name of the form (name ^ number) different from those that
    already exist

val register_in_db : string -> string -> unit
    register_in_db name password registers the couple in the database

val update_statistics : unit -> unit
val choose_word : unit -> unit

```

Sets the word variable by choosing a word in the dictionary

```
val my_input_line : Unix.file_descr -> string
    my_input_line file_descr returns the string read on file_descr

val my_nth : string -> int -> string
    my_nth s n returns the n-th element in the string s of the form
    element_0/element_1/.../element_n

val notify_players : string -> string -> unit
    notify_players keyword name sends the keyword command (such as EXITED or
    WORD_FOUND) to all players

val notify_guess : string -> string -> unit
val notify_line : string -> unit
val notify_cheat : string -> unit
val notify_talk : string -> string -> unit
val send_connected_command : unit -> unit
val send_new_round_command : string -> unit
val choose_drawer : unit -> int
val send_score_round_command : unit -> unit
val send_end_round_command : unit -> unit
val reset_score_players : unit -> unit
class player : string -> Unix.file_descr ->
    object
        val mutable connected : bool
        val mutable name : string
        val mutable role : Server.role
        val s_descr : Unix.file_descr
        val mutable score_game : int
        val mutable score_round : int
        method get_name : unit -> string
        method get_role : unit -> Server.role
        method get_score_game : unit -> int
        method get_score_round : unit -> int
        method get_status : unit -> bool
        method send_command : string -> unit
        method set_role : Server.role -> unit
        method set_score_round : int -> unit
        method start_game : unit -> unit
        method update_score_game : unit
```

```

end

val welcome_player : string -> Unix.file_descr -> unit
val connection_player : Unix.file_descr * 'a -> unit
class server : int -> int ->
  object
    val nb_pending : int
    val port_num : int
    val s_descr : Unix.file_descr
    method start_game : unit -> unit

    Method to start one game

    method wait_connections : unit -> unit

    Accept connections on the socket

  end

  Class for iSketch server

val read_db : Pervasives.in_channel -> string
  Read in the database the number of wins and defeats of every player

val generate_response : string -> string
  Generate the webpage to display with the current values of the database

val response : Unix.file_descr * 'a -> unit
  Calls the generate_response function if a GET request is sent

class serverHTTP : int -> int ->
  object
    val nb_pending : int
    val port_num : int
    val s_descr : Unix.file_descr
    method start : unit -> unit
  end

  Class for HTTP server

val main : unit -> unit

```