Zappy SERVER

Generated by Doxygen 1.9.6

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 s_app Struct Reference

Collaboration diagram for s_app:

3.2 s_client Struct Reference

Public Attributes

- size_t **fd**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/server/client.h

3.3 s_egg Struct Reference

Collaboration diagram for s_egg:

Public Attributes

- vector2i_t * **pos**
- size_t **id**
- bool **is_laid**
- size_t **id_player_laid**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/types.h

3.4 s_game Struct Reference

Collaboration diagram for s_game:

Public Attributes

- size_t **height**
- size_t **width**
- tile_t ** **map**
- struct timeval **start**
- struct timeval **start_food**
- int **freq**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/game/game.h

3.5 s_gui Struct Reference

Collaboration diagram for s_gui:

Public Attributes

- size_t **fd**
- list_t * **list_messages**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/gui/gui.h

3.6 s_ia Struct Reference

Collaboration diagram for s_ia:

Public Attributes

- size_t **fd**
- list_t * **list_command**
- list_t * **list_messages**
- vector2i_t * **position**
- orientation_t **direction**
- inventory_t * **inventory**
- incantation_info_t * **incantation**
- size_t **level**
- time_info_t * **time**
- char * **team_name**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/ai/ai.h

3.7 s_incantation_info Struct Reference

Public Attributes

- bool **status_incantation**
- size_t **target_level**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/ai/ai.h

3.8 s_inventory Struct Reference

Public Attributes

- size_t **food**
- size_t **linemate**
- size_t **deraumere**
- size_t **sibur**
- size_t **mendiane**
- size_t **phiras**
- size_t **thystame**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/ai/ai.h

3.9 s_list Struct Reference

Collaboration diagram for s_list:

Public Attributes

- list_node_t * **first**
- list_node_t * **last**
- size_t **len**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/list/type.h

3.10 s_list_node Struct Reference

Collaboration diagram for s_list_node:

Public Attributes

- node_data_t **data**
- struct s_list_node * **next**
- struct s_list_node * **prev**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/list/type.h

3.11 s_node_data Union Reference

Collaboration diagram for s_node_data:

Public Attributes

- ia_t * **ai**
- gui_t * **gui**
- client_t * **client**
- team_t * **team**
- char * **message**
- char * **command**
- vector2i_t * **coord**
- egg_t * **egg**

The documentation for this union was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/list/type.h

3.12 s_parsing Struct Reference

Struct for parsing the arguments.

```
#include <types.h>
```

Public Attributes

- int **port**
- int **width**
- int **height**
- int **clientsNb**
- int **freq**
- char ** **names**

3.12.1 Detailed Description

Struct for parsing the arguments.

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/types.h

3.13 s_server Struct Reference

Public Attributes

- fd_set **read_fds**
- fd_set **write_fds**
- int **fd**
- socklen_t **addrlen**
- struct sockaddr_in **addr**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/server/server.h

3.14 s_team Struct Reference

Collaboration diagram for s_team:

Public Attributes

- list_t * **list_ai**
- list_t * **eggs_list**
- char * **name**
- size_t **max_place**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/ai/team.h

3.15 s_time_info Struct Reference

Public Attributes

- bool **stuck**
- struct timeval **start_stuck**
- double **total_stuck**
- struct timeval **start_life**
- double **total_life**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/ai/ai.h

3.16 s_vector2i Struct Reference

Public Attributes

- int **x**
- int **y**

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/types.h

3.17 tile_s Struct Reference

Struct for a map's tile.

`#include <map.h>`

Public Attributes

- size_t **food**
- size_t **linemate**
- size_t **deraumere**
- size_t **sibur**
- size_t **mendiane**
- size_t **phiras**
- size_t **thystame**

3.17.1 Detailed Description

Struct for a map's tile.

The documentation for this struct was generated from the following file:

- /home/tjerome-rocher/Desktop/Tek2/Zappy/server/include/map/map.h

Chapter 4

File Documentation

4.1 ai.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** AI
00006 */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdlib.h>
00012 #include <stdbool.h>
00013
00014 #include "list/list.h"
00015
00016 typedef struct s_app app_t;
00017
00018 typedef enum {
00019     NORTH = 1,
00020     EAST,
00021     SOUTH,
00022     WEST,
00023 } orientation_t;
00024
00025 typedef struct s_inventory {
00026     size_t food;
00027     size_t linemate;
00028     size_t deraumere;
00029     size_t sibur;
00030     size_t mendiane;
00031     size_t phiras;
00032     size_t thystame;
00033 } inventory_t;
00034
00035 typedef struct s_time_info {
00036     bool stuck;
00037     struct timeval start_stuck;
00038     double total_stuck;
00039     struct timeval start_life;
00040     double total_life;
00041 } time_info_t;
00042
00043 typedef struct s_incantation_info {
00044     bool status_incantation;
00045     size_t target_level;
00046 } incantation_info_t;
00047
00048 typedef struct s_ia {
00049     size_t fd;
00050     list_t *list_command;
00051     list_t *list_messages;
00052     vector2i_t *position;
00053     orientation_t direction;
00054     inventory_t *inventory;
00055     incantation_info_t *incantation;
00056     size_t level;
00057     time_info_t *time;
00058     char *team_name;
```

```
00059 } ia_t;
00060
00069 ia_t *create_ia(app_t *app, int fd, team_t *team);
00070
00078 void add_ia(app_t *app, size_t fd, char *line);
00079
00087 ia_t *find_ia(app_t *app, size_t fd);
00088
00094 void check_die(app_t *app);
```

4.2 command_ai.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** AI Command
00006 */
00007
00008 #pragma once
00009
00010 #include "ai/ai.h"
00011
00012 #define BEGIN_INCANTATION 1
00013 #define END_INCANTATION 2
00014
00022 void command_ai_handler(app_t *app, ia_t *ai, char *line);
00023
00033 bool move_command(app_t *app, ia_t *ai, char *line);
00034
00044 bool object_info_command(app_t *app, ia_t *ai, char *line);
00045
00055 bool other_command(app_t *app, ia_t *ai, char *line);
00056
00063 void eject_cmd(app_t *app, ia_t *ai);
00064
00071 void eject_player(app_t *app, ia_t *ai);
00072
00079 void eject_egg(app_t *app, ia_t *ai);
00080
00086 void dead_response(app_t *app);
00087
00094 void fork_cmd(app_t *app, ia_t *ai);
00095
00102 void connect_nbr_cmd(app_t *app, ia_t *ai);
00103
00111 void take_cmd(app_t *app, ia_t *ai, char *ressource);
00112
00120 void set_cmd(app_t *app, ia_t *ai, char *ressource);
00121
00131 bool broadcast_command(app_t *app, ia_t *ai, char *line);
00132
00142 bool incantation_command(app_t *app, ia_t *ai, char *line);
00143
00153 list_t *check_incantation(app_t *app, ia_t *ai, int status);
00154
00162 void update_status(app_t *app, ia_t *ai, int update_status);
00163
00170 void level_up(app_t *app, ia_t *ai);
00171
00178 void add_command_to_list(ia_t *ai, char *line);
00179
00185 void treat_command(app_t *app);
00186
00192 void destroy_command_list(list_t *command_list);
00193
00202 size_t calcul_k(app_t *app, ia_t *ai_sender, ia_t *ai_destination);
```

4.3 look.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** AI Command
00006 */
00007
00008 #pragma once
00009
```

```
00010 #include "ai/ai.h"
00011
00012 #define PLAYER_STRING " player"
00013 #define EGG_STRING " egg"
00014 #define FOOD_STRING " food"
00015 #define LINEMATE_STRING " linemate"
00016 #define DERAUMERE_STRING " deraumere"
00017 #define SIBUR_STRING " sibur"
00018 #define MENDIANE_STRING " mendiane"
00019 #define PHIRAS_STRING " phiras"
00020 #define THYSTAME_STRING " thystame"
00021
00028 void look_cmd(app_t *app, ia_t *ai);
00029
00038 void check_player(vector2i_t *pos, app_t *app, char **reply, ia_t *ai);
00039
00047 void check_egg(vector2i_t *pos, app_t *app, char **reply);
00048
00056 void check_resources(vector2i_t *pos, app_t *app, char **reply);
00057
00066 void look_north(app_t *app, ia_t *ai, int index_line, char **reply);
00067
00076 void look_east(app_t *app, ia_t *ai, int index_line, char **reply);
00077
00086 void look_south(app_t *app, ia_t *ai, int index_line, char **reply);
00087
00096 void look_west(app_t *app, ia_t *ai, int index_line, char **reply);
```

4.4 stuck.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** Game struct
00006 */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdlib.h>
00012 #include <stdbool.h>
00013 #include <sys/time.h>
00014
00015 #include "list/list.h"
00016
00017 typedef struct s_app app_t;
00018
00025 double time_elapsed(struct timeval *time);
00026
00033 void set_time_stuck(ia_t *ai, double total_stuck);
00034
00040 void treat_stuck(app_t *app);
```

4.5 team.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** Team struct
00006 */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdlib.h>
00012 #include <stdbool.h>
00013
00014 #include "list/list.h"
00015
00016 typedef struct s_app app_t;
00017
00018 typedef struct s_team {
00019     list_t *list_ai;
00020     list_t *eggs_list;
00021     char *name;
00022     size_t max_place;
00023 } team_t;
```

```
00024
00033 team_t *create_team(app_t *app, char *name, size_t max_place);
00034
00042 void add_team(app_t *app, char *team_name, size_t max_place);
00043
00051 void add_egg(list_t *eggs, int id_player_laid, app_t *app);
00052
00061 void add_egg_on_player(list_t *eggs, int id_player_laid,
00062     app_t *app, ia_t *ai);
00063
00071 team_t *find_team(app_t *app, size_t fd);
00072
00078 void destroy_team(list_t *teams_list);
```

4.6 app.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** App
00006 */
00007
00008 #pragma once
00009
00010 #include <time.h>
00011 #include <stdio.h>
00012 #include <stdlib.h>
00013 #include <stdarg.h>
00014 #include <string.h>
00015 #include <signal.h>
00016
00017 #include "ai/ai.h"
00018 #include "ai/team.h"
00019 #include "gui/gui.h"
00020 #include "parsing.h"
00021 #include "ai/stuck.h"
00022 #include "list/list.h"
00023 #include "game/game.h"
00024 #include "server/server.h"
00025
00026 typedef struct s_app {
00027     list_t *gui_list;
00028     list_t *teams_list;
00029     list_t *clients_list;
00030     server_t *server;
00031     game_t *game;
00032 } app_t;
00033
00040 app_t *create_app(parsing_t *parsing);
00041
00047 void destroy_app(app_t *app);
```

4.7 game.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** Game struct
00006 */
00007
00008 #pragma once
00009
00010 #include "map/map.h"
00011
00012 typedef struct s_game {
00013     size_t height;
00014     size_t width;
00015     tile_t **map;
00016     struct timeval start;
00017     struct timeval start_food;
00018     int freq;
00019 } game_t;
00020
00029 game_t *create_game(int height, int width, int freq);
00030
00036 void spawn_ressources(app_t *app);
00037
```

```
00045 bool check_win(app_t *app);
00046
00052 void destroy_game(game_t *game);
```

4.8 communication.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Header file for the communication functions between the GUI and the server.
00006 */
00007
00008 #pragma once
00009
00010 #include "gui.h"
00011 #include "../app/app.h"
00012
00013 #define POS_SPACE 3
00014 #define LEN_COMMAND 3
00015 #define LEN_COMMAND_AND_SPACE LEN_COMMAND + 1
00016
00017 #define FOOD_INDEX 0
00018 #define LINEMATE_INDEX 1
00019 #define DERAUMERE_INDEX 2
00020 #define SIBUR_INDEX 3
00021 #define MENDIANE_INDEX 4
00022 #define PHIRAS_INDEX 5
00023 #define THYSTAME_INDEX 6
00024
00025 enum e_command_label {
00026     CL_MSZ,
00027     CL_BCT,
00028     CL_MCT,
00029     CL_TNA,
00030     CL_PPO,
00031     CL_PLV,
00032     CL_PIN,
00033     CL_SGT,
00034     CL_SST,
00035     CL_LEN,
00036 };
00037
00045 void handle_command_gui(gui_t *gui, app_t *app, char *line);
00046
00054 void msz_response(gui_t *gui, app_t *app, char *line);
00055
00063 void tna_response(gui_t *gui, app_t *app, char *line);
00064
00072 void sgt_response(gui_t *gui, app_t *app, char *line);
00073
00081 void sst_response(gui_t *gui, app_t *app, char *line);
00082
00090 void bct_response(gui_t *gui, app_t *app, char *line);
00091
00099 void mct_response(gui_t *gui, app_t *app, char *line);
00100
00108 void ppo_response(gui_t *gui, app_t *app, char *line);
00109
00117 void plv_response(gui_t *gui, app_t *app, char *line);
00118
00126 void pin_response(gui_t *gui, app_t *app, char *line);
00127
00136 void pnw_command(app_t *app, ia_t *ai, gui_t *gui);
00137
00144 void pex_command(app_t *app, int player_id);
00145
00153 void pbc_command(app_t *app, int player_id, char *message);
00154
00161 void pfk_command(app_t *app, int player_id);
00162
00171 void pdr_command(app_t *app, int player_id, size_t index_ressource);
00172
00181 void pgt_command(app_t *app, int player_id, size_t index_ressource);
00182
00189 void pdi_command(app_t *app, int player_id);
00190
00197 void seg_command(app_t *app, char *team);
00198
00205 void smg_command(app_t *app, char *message);
00206
00212 void suc_command(gui_t *gui);
00213
```

```
00219 void sbp_command(gui_t *gui);
00220
00228 void pic_command(app_t *app, list_t *ai);
00229
00236 void pie_command(app_t *app, list_t *ai);
00237
00246 void enw_command(app_t *app, egg_t *egg, gui_t *gui);
00247
00254 void edi_command(app_t *app, int egg_id);
00255
00262 void ebo_command(app_t *app, int egg_id);
```

4.9 gui.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** Gui struct
00006 */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdbool.h>
00012
00013 #include "app/app.h"
00014 #include "list/list.h"
00015
00016 typedef struct s_gui {
00017     size_t fd;
00018     list_t *list_messages;
00019 } gui_t;
00020
00027 gui_t *create_gui(int fd);
00028
00036 void add_gui(app_t *app, size_t fd, char *line);
00037
00045 gui_t *find_gui(app_t *app, size_t fd);
00046
00052 void destroy_gui(list_t *gui_list);
```

4.10 list.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** Function about list
00006 */
00007
00008 #pragma once
00009
00010 #include <stdbool.h>
00011
00012 #include "type.h"
00013
00019 list_t *list_new(void);
00020
00026 void list_free(list_t *list);
00027
00034 void list_delete(list_t *list, list_node_t *node);
00035
00041 void list_remove_front(list_t *list);
00042
00048 void list_remove_back(list_t *list);
00049
00057 bool list_add_back(list_t *list, node_data_t data);
00058
00066 bool list_add_front(list_t *list, node_data_t data);
```

4.11 type.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
```

```
00003 ** Zappy Server
00004 ** File description:
00005 ** Type of List
00006 */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011
00012 #include "types.h"
00013
00014 typedef struct s_ia ia_t;
00015 typedef struct s_gui gui_t;
00016 typedef struct s_team team_t;
00017 typedef struct s_client client_t;
00018 typedef struct s_vector2i vector2i_t;
00019
00020 typedef union s_node_data {
00021     ia_t *ai;
00022     gui_t *gui;
00023     client_t *client;
00024     team_t *team;
00025     char *message;
00026     char *command;
00027     vector2i_t *coord;
00028     egg_t *egg;
00029 } node_data_t;
00030
00031 typedef struct s_list_node {
00032     node_data_t data;
00033     struct s_list_node *next;
00034     struct s_list_node *prev;
00035 } list_node_t;
00036
00037 typedef struct s_list {
00038     list_node_t *first;
00039     list_node_t *last;
00040     size_t len;
00041 } list_t;
```

4.12 map.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** map
00006 */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdbool.h>
00012
00013 #define FOOD_DENSITY 0.5
00014 #define LINEMATE_DENSITY 0.3
00015 #define DERAUMERE_DENSITY 0.15
00016 #define SIBUR_DENSITY 0.1
00017 #define MENDIANE_DENSITY 0.1
00018 #define PHIRAS_DENSITY 0.08
00019 #define THYSTAME_DENSITY 0.05
00020
00024 enum entity_type_e {
00025     EGG,
00026     FOOD,
00027     LINEMATE,
00028     DERAUMERE,
00029     SIBUR,
00030     MENDIANE,
00031     PHIRAS,
00032     THYSTAME,
00033     NONE
00034 };
00035
00039 typedef struct tile_s {
00040     size_t food;
00041     size_t linemate;
00042     size_t deraumere;
00043     size_t sibur;
00044     size_t mendiane;
00045     size_t phiras;
00046     size_t thystame;
00047 } tile_t;
```

```
00048
00056 tile_t **create_map(int width, int height);
00057
00064 void free_map(tile_t **map, int height);
00065
00073 void display_map(tile_t **map, int height, int width);
```

4.13 parsing.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Header file for parsing the arguments passed to the server.
00006 */
00007
00008 #pragma once
00009
00010 #define HELP_FLAG_LABEL "--help"
00011 #define HELP_FLAG "-h"
00012 #define PORT_FLAG "-p"
00013 #define WIDTH_FLAG "-x"
00014 #define HEIGHT_FLAG "-y"
00015 #define TEAMS_NAMES_FLAG "-n"
00016 #define CLIENTS_FLAG "-c"
00017 #define FREQUENCY_FLAG "-f"
00018
00019 #define NB_ARGS_MIN 13
00020 #define NB_ARGS_HELP 2
00021
00022 #include "types.h"
00023
00024 #include <stdbool.h>
00025
00033 parsing_t *parse_arg(int ac, char **av);
00034
00044 int handle_help(int ac, char **av);
00045
00053 int parse_positive_int_arg(char *arg);
00054
00067 int parse_client(char **arg, int *pos, parsing_t *parsing);
00068
00081 int parse_frequency(char **arg, int *pos, parsing_t *parsing);
00082
00095 int parse_height(char **arg, int *pos, parsing_t *parsing);
00096
00109 int parse_port(char **arg, int *pos, parsing_t *parsing);
00110
00123 int parse_width(char **arg, int *pos, parsing_t *parsing);
00124
00137 int parse_names(char **arg, int *pos, parsing_t *parsing);
00138
00144 void destroy_parsing(parsing_t *parsing);
```

4.14 rules.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Rules for the server
00006 */
00007
00008 #pragma once
00009
00010 enum RETURN_CODES {
00011     CODE_ERROR_MISSING_ARG = -1,
00012     CODE_ERROR_INVALID_ARG = -2,
00013     CODE_ERROR_WRONG_FLAG = -3,
00014     CODE_ERROR_INVALID_NUMBER = -4,
00015     CODE_ERROR_MALLOC_FAILED = -5,
00016     CODE_HELP_SUCCESS = 1,
00017     CODE_SUCCESS = 0,
00018     CODE_FAILLURE = 84
00019 };
00020
00021 #define MAX_ITEMS 7
```

4.15 client.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** Client
00006 */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdlib.h>
00012 #include <stdbool.h>
00013
00014 #include "app/app.h"
00015
00016 typedef struct s_client {
00017     size_t fd;
00018 } client_t;
00019
00026 client_t *create_client(int fd);
00027
00036 bool its_client(app_t *app, size_t fd);
00037
00045 list_node_t *find_client(list_t *clients_list, size_t fd);
00046
00052 void destroy_client(list_t *client_list);
```

4.16 server.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy Server
00004 ** File description:
00005 ** Server struct
00006 */
00007
00008 #pragma once
00009
00010 #include <stdio.h>
00011 #include <stddef.h>
00012 #include <stdlib.h>
00013 #include <unistd.h>
00014 #include <stdbool.h>
00015 #include <sys/select.h>
00016 #include <netinet/in.h>
00017
00018 #include "list/list.h"
00019
00020 #define LISTEN_NUMBER 100
00021 #define WELCOME_MESSAGE "WELCOME\n"
00022 #define WELCOME_MESSAGE_LEN 8
00023 #define SELECT_TIMEOUT_SECONDS 1
00024
00025 #define END_GAME 1
00026 #define ERROR -1
00027 #define GAME_CONTINUE 0
00028
00029 typedef struct s_app app_t;
00030
00031 typedef struct s_server {
00032     fd_set read_fds;
00033     fd_set write_fds;
00034     int fd;
00035     socklen_t addrlen;
00036     struct sockaddr_in addr;
00037 } server_t;
00038
00044 void destroy_server(server_t *server);
00045
00052 server_t *create_server(size_t port);
00053
00060 bool server_run(app_t *app);
00061
00069 bool server_connection_handler(app_t *app, size_t fd);
00070
00078 bool server_data_handler(app_t *app, size_t fd);
00079
00086 char *read_line(int fd);
00087
00094 void server_quit_handler(app_t *app, size_t fd);
00095
```

```
00104 bool write_message(list_t *list_messages, size_t fd);
00105
00112 void add_message(list_t *list, char *message);
00113
00121 char *format_string(const char *format, ...);
00122
00128 void server_reset_fd(app_t *app);
00129
00137 void handle_request(app_t *app, size_t fd, char *line);
00138
00146 char *append_char(char *line, char current_char);
00147
00154 void handle_client_read(app_t *app, int fd);
00155
00162 void handle_client_write(app_t *app, int fd);
00163
00171 vector2i_t *create_vector2i(int x, int y);
00172
00179 void concatenate_strings(char **str1, char *str2);
00180
00186 void destroy_message_list(list_t *message_list);
```

4.17 types.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Types for the server
00006 */
00007
00008 #pragma once
00009
00010 #include <stddef.h>
00011 #include <stdbool.h>
00012
00017 typedef struct s_parsing {
00018     int port;
00019     int width;
00020     int height;
00021     int clientsNb;
00022     int freq;
00023     char **names;
00024 } parsing_t;
00025
00026 typedef struct s_vector2i {
00027     int x;
00028     int y;
00029 } vector2i_t;
00030
00031 typedef struct s_egg {
00032     vector2i_t *pos;
00033     size_t id;
00034     bool is_laid;
00035     size_t id_player_laid;
00036 } egg_t;
```

4.18 utils.h

```
00001 /*
00002 ** EPITECH PROJECT, 2024
00003 ** Zappy
00004 ** File description:
00005 ** Header file for utils functions.
00006 */
00007
00008 #pragma once
00009
00010 #include <stdbool.h>
00011
00019 int parse_positive_int_arg(char *arg);
```

Index