

A decorative border surrounds the central text, composed of various colorful geometric shapes and patterns including squares, circles, crosses, wavy lines, and grids of dots in shades of blue, green, yellow, orange, and purple.

# PartyPlaytime

<https://github.com/bogdanputinelu/uni-PartyPlaytime>

# WHOA!

Aplicatia **PartyPlaytime** este o aplicatie destinata distractiei pentru grupuri de prieteni. Utilizatorul poate juca mai multe mini-game-uri cu grupurile de prieteni. In cazul in care nu are un grup de prieteni cu care sa se joace, utilizatorul poate opta pentru jocuri single player, avand posibilitatea de a juca cu un bot.



# JOCURI DISPONIBILE



## Headspin

minim 2 echipe  
de 2 jucatori



## Word Rush

minim 2 echipe  
de 2 jucatori



## TicTacToe

1 – 2 jucatori



## BoardBlitz

2 – 4 jucatori

# Headspin

‘**Headspin**’ este un joc de societate ce necesita cel puțin două echipe a câte doi jucători. Fiecare jucător al echipei se va afla pe rând în spatele calculatorului. Jucătorul are la dispoziție mai multe cuvinte pe care trebuie să ghicească cu ajutorul coechipierului. Castiga echipa care a strans cele mai multe puncte.

+



× ×



# Word Rush

‘**Word Rush**’ este un joc de societate ce necesita cel puțin două echipe a câte doi jucători. Pe ecran va apărea o expresie sau un cuvânt care trebuie mimat, desenat sau descris de un membru al echipei. Ceilalți coechipieri trebuie să ghicească cuvântul. Dacă acestia îl ghicesc, primesc un punct. Prima echipă care ajunge la X puncte castigă.



# Boardblitz

‘**Board Blitz**’ este un joc de societate ce necesita 2, 3, sau 4 persoane. Fiecare jucator are 4 pioni, iar scopul sau este acela de a ii duce in zona de finish. Pionii avanseaza in functie de numarul indicat pe zar. Primul jucator care are toti pionii in zona de finish castiga.

+



× ×

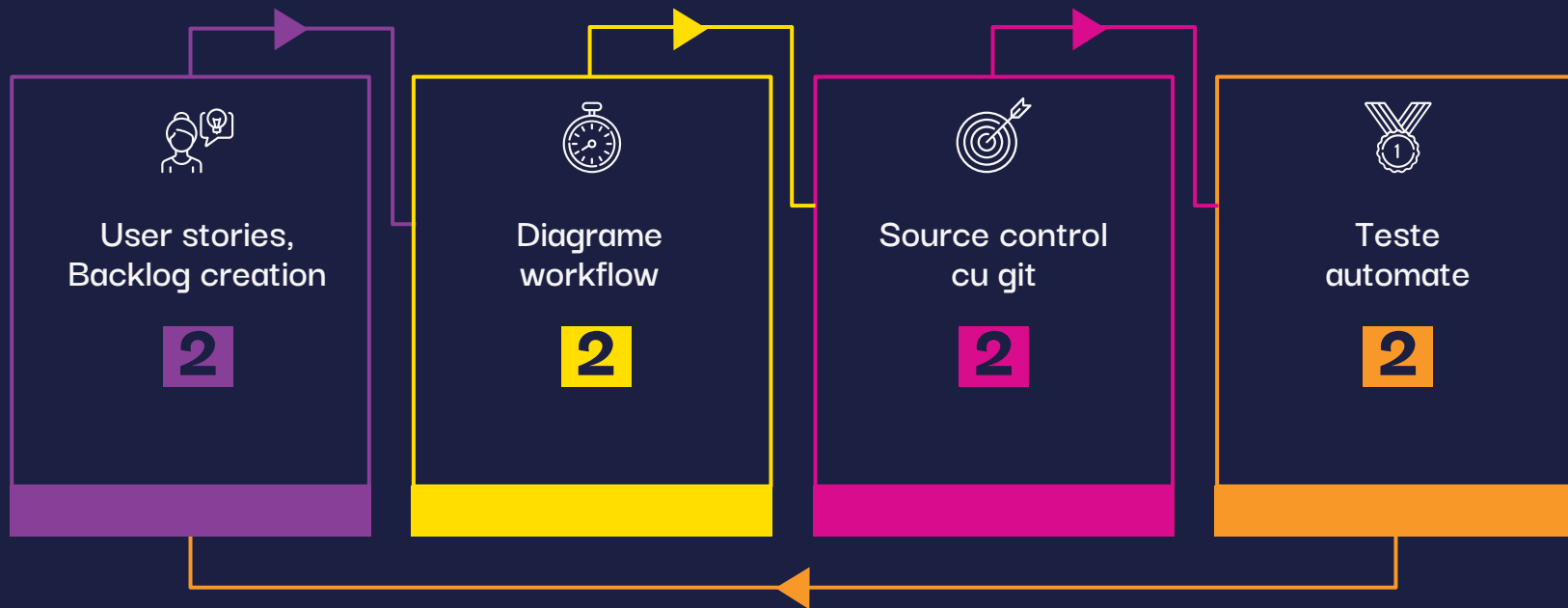


# TicTacToe

‘**Tic Tac Toe**’ este un joc clasic ce necesita doua persoane. Pe ecran va apareea un tabel cu trei linii si trei coloane. Fiecare jucator marcheaza pe rand cate o casuta cu X si O, iar primul jucator care reușește să marcheze 3 căsute adiacente pe orizontală, verticală sau diagonală caștigă jocul. In cazul in care este un singur utilizator, acesta o sa joace cu un bot.

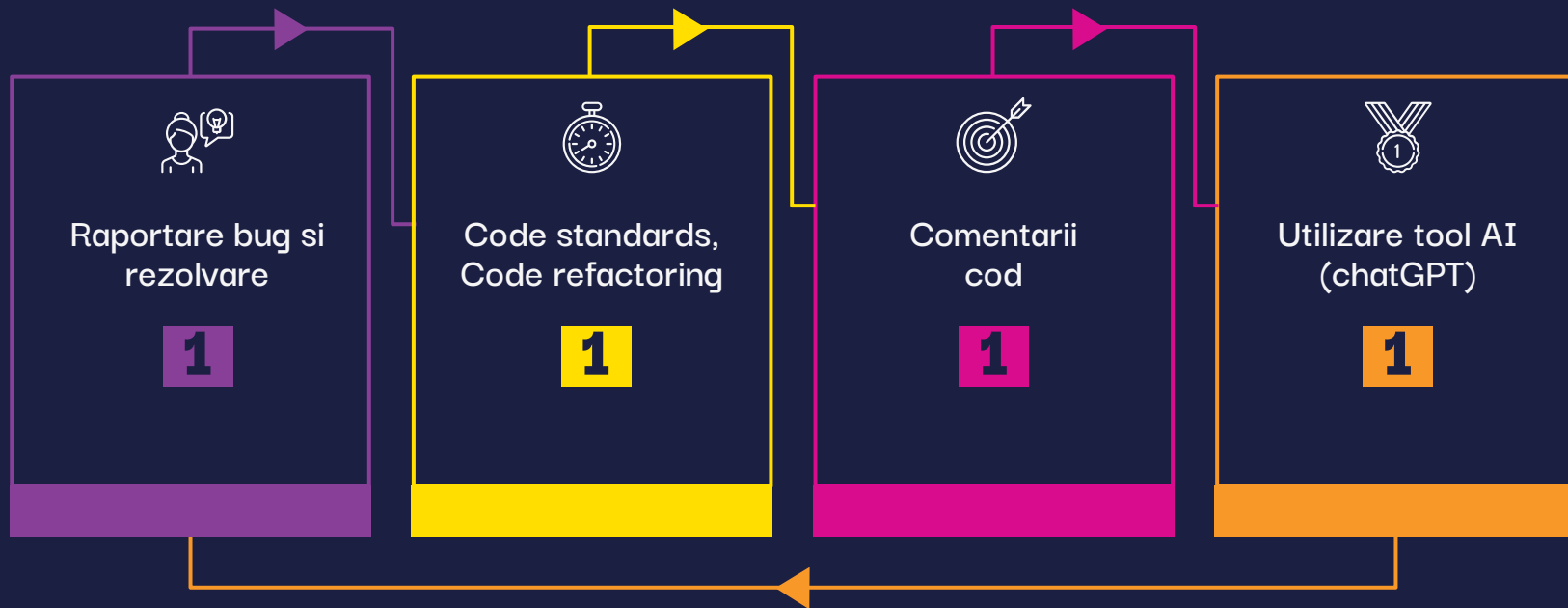


# DEZVOLTAREA SOFTWARE





# DEZVOLTAREA SOFTWARE





# USER STORIES

1. Ca utilizator, as dori o aplicatie cu diverse jocuri de societate.
2. Ca utilizator, as dori sa imi pot crea un cont in aplicatie.
3. Ca utilizator, as dori sa am posibilitatea de a edita contul meu.
4. Ca utilizator, as dori sa imi pot seta un nickname.
5. Ca utilizator, as dori sa pot juca un joc in mod single player cu ajutorul unui bot.
6. Ca utilizator, as dori un set de instructiuni pentru fiecare joc.
7. Ca utilizator, as dori sa pot juca jocuri in echipa.
8. Ca utilizator, as dori sa salvez diverse setari ale jocurilor de grup in contul meu.





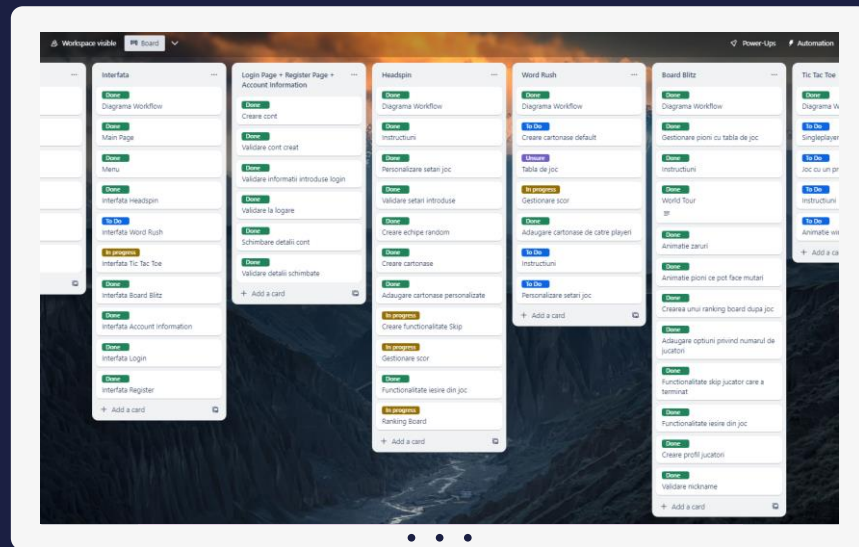
# USER STORIES

9. Ca jucator, as dori ca jocurile din aplicatie sa imi indice de la inceput numarul minim de jucatori.
10. Ca jucator, as dori sa setez numarul de runde in urma carora sa se castige un joc.
11. Ca jucator, as dori sa setez cate puncte trebuie acumulate pentru a castiga un joc.
12. Ca jucator, as dori sa pot adauga optiuni personalizate in jocuri.
13. Ca jucator, as dori sa pot seta numarul de jucatori pentru jocurile de grup.
14. Ca jucator, sa dori sa vad clasamentul la finalul jocului.
15. Ca jucator, as dori sa vad modificari pe tabla de joc in timp real.
16. Ca jucator, as dori sa primesc o atentionare in momentul in care parasesc jocul din greseala.



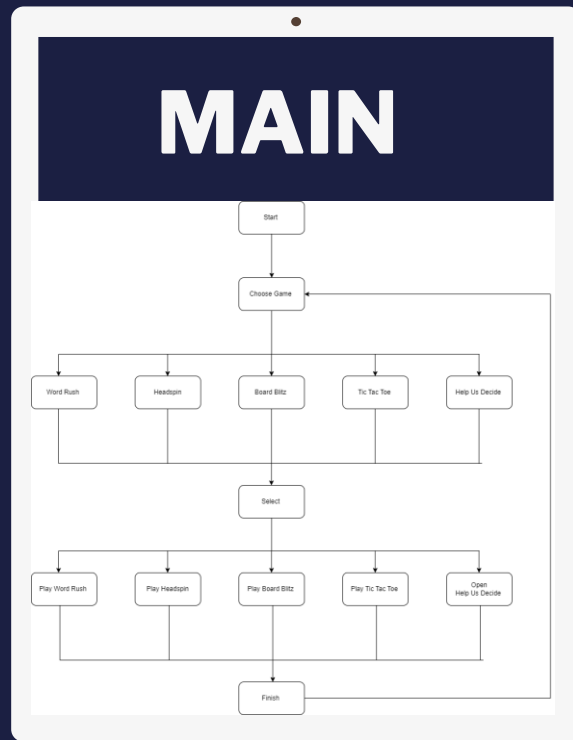
# BACKLOG

Backlog-ul proiectului a fost realizat cu ajutorul aplicatiei Trello. Fiecare joc a avut o lista de functionalitati ce trebuie implementate. Acestea au fost marcate pe parcurs cu etichete de tipul 'Done', 'In progress', 'To do' sau 'Unsure' pentru o gestionare mai buna a activitatii.



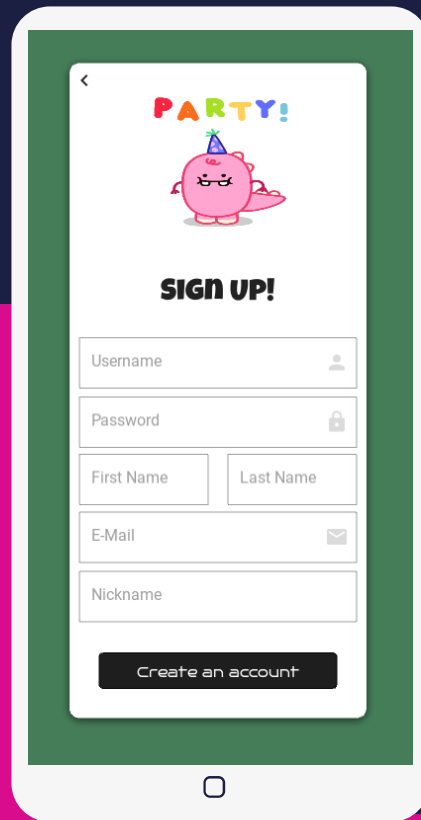
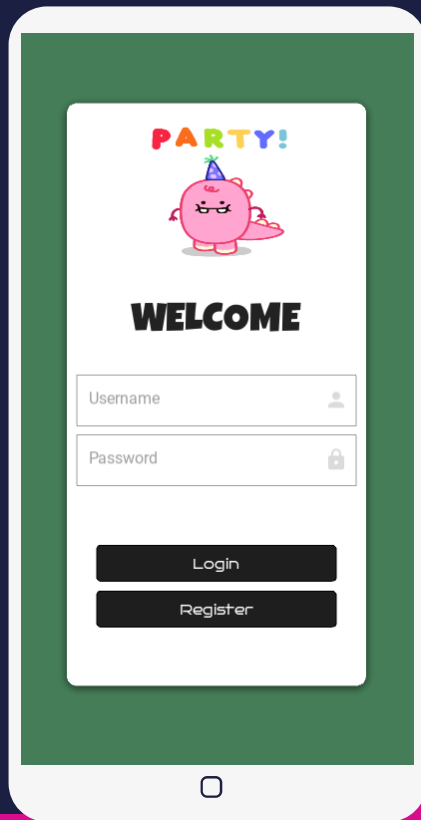
# DIAGRAME WORKFLOW

Diagramele contin cinci componente (**Main**, Word Rush, Headspin, Board Blitz si Tic Tac Toe) impartite conform functionalitatilor aplicatiei. Structurarea lor clara permite o buna intelegere a pasilor de implementare.



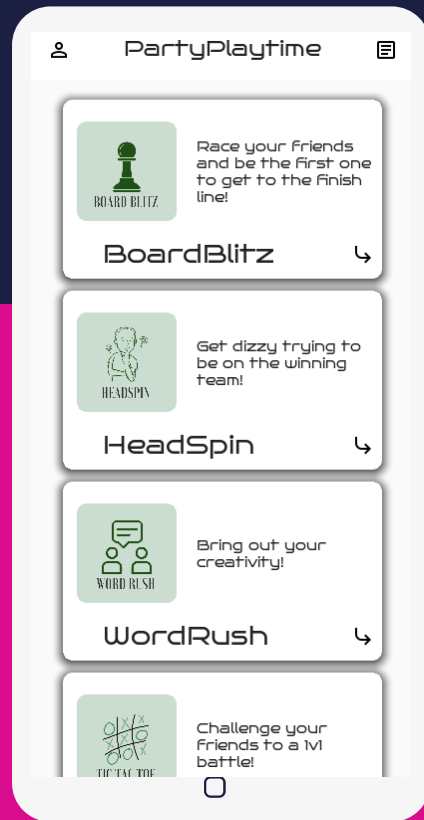
# CREATE ACCOUNT

Utilizatorul isi poate crea un cont pe platforma in care sa isi salveze setarile jocurilor. Username-ul, parola si mail-ul sunt validate atat la crearea unui cont nou, cat si la logare.



# HOME PAGE

Pagina principala contine card-urile care te duc pe fiecare joc. In plus, utilizatorul poate consulta numarul minim de jucatori in sectiunea Game Rules. Acesta isi poate modifica setarile contului in sectiunea User Account.



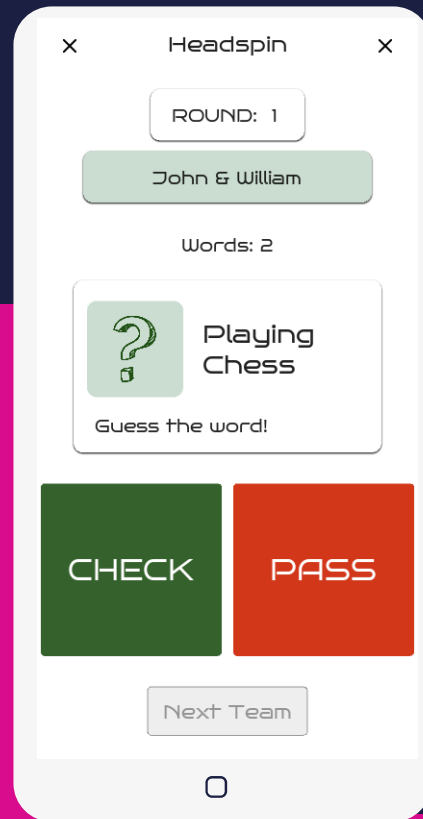
Diagramele contin cinci componente (Main, Word Rush, **Headspin**, Board Blitz si Tic Tac Toe) impartite conform functionalitatilor aplicatiei. Structurarea lor clara permite o buna intelegere a pasilor de implementare.





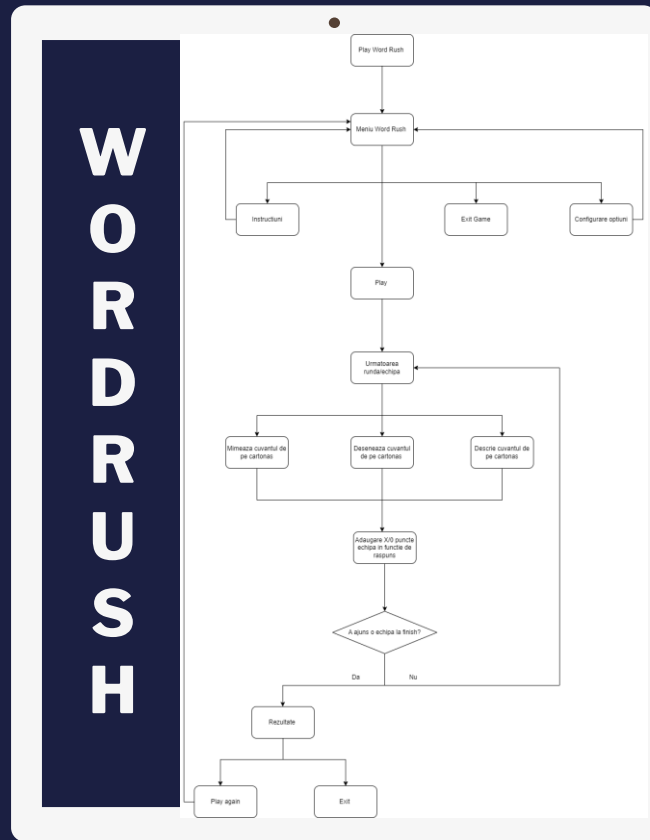
# HEADSPIN GAME

Interfata jocului permite gestionarea punctajului fiecărei echipe, butoanele Check, Pass și Next Team alternându-si starea de disable/enable pentru o bună coordonare a desfășurării jocului.



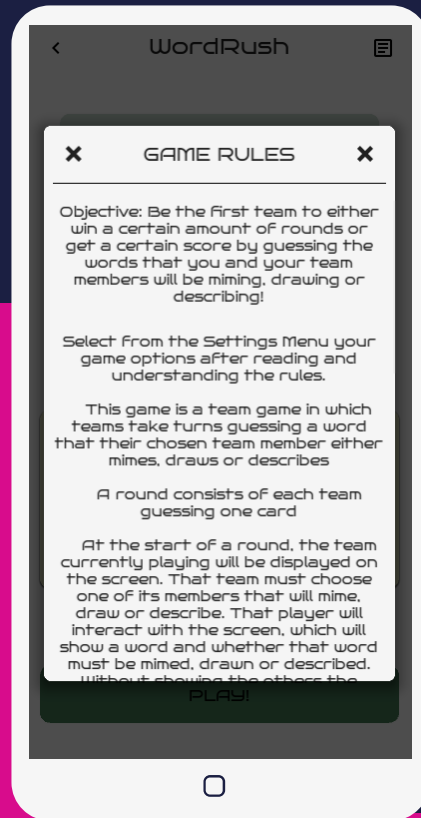
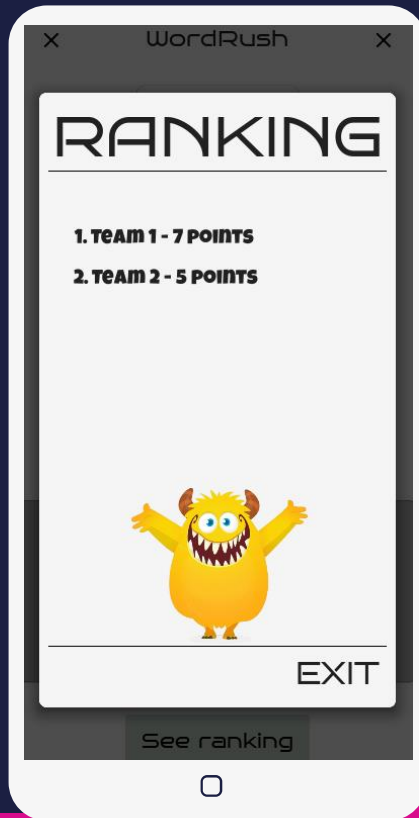
# DIAGRAME WORKFLOW

Diagramele contin cinci componente (Main, **Word Rush**, Headspin, Board Blitz si Tic Tac Toe) impartite conform functionalitatilor aplicatiei. Structurarea lor clara permite o buna intelegere a pasilor de implementare.



# WORDRUSH GAME

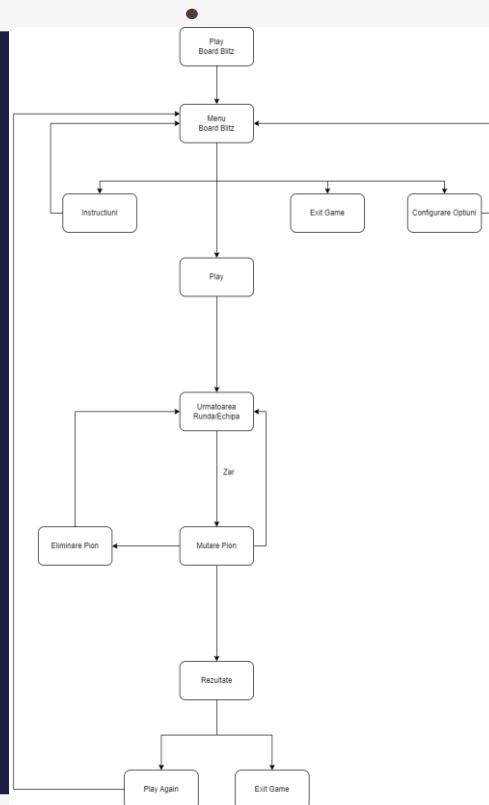
La finalul jocurilor se afiseaza clasamentul conform punctajelor obtinute. De asemenea, jocurile au sectiunea Game Rules ce poate fi consultata inaintea inceperii.



# DIAGRAM WORKFLOW

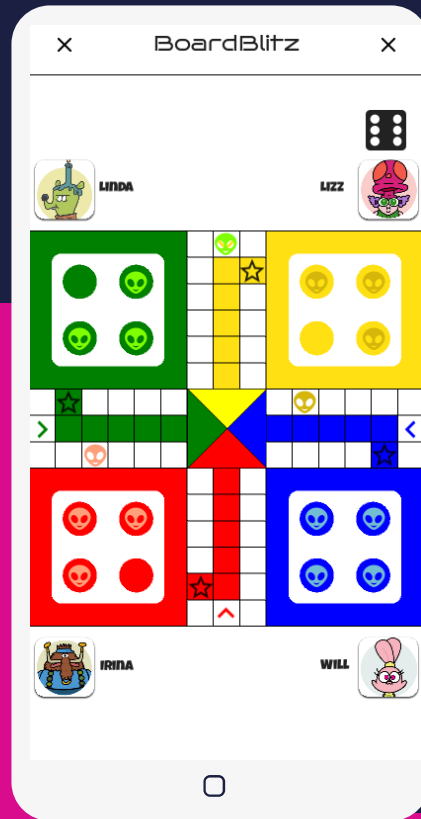
Diagramele contin cinci componente (Main, Word Rush, Headspin, **Board Blitz** si Tic Tac Toe) impartite conform functionalitatilor aplicatiei. Structurarea lor clara permite o buna intelegere a pasilor de implementare.

## B O A R D B L I T Z



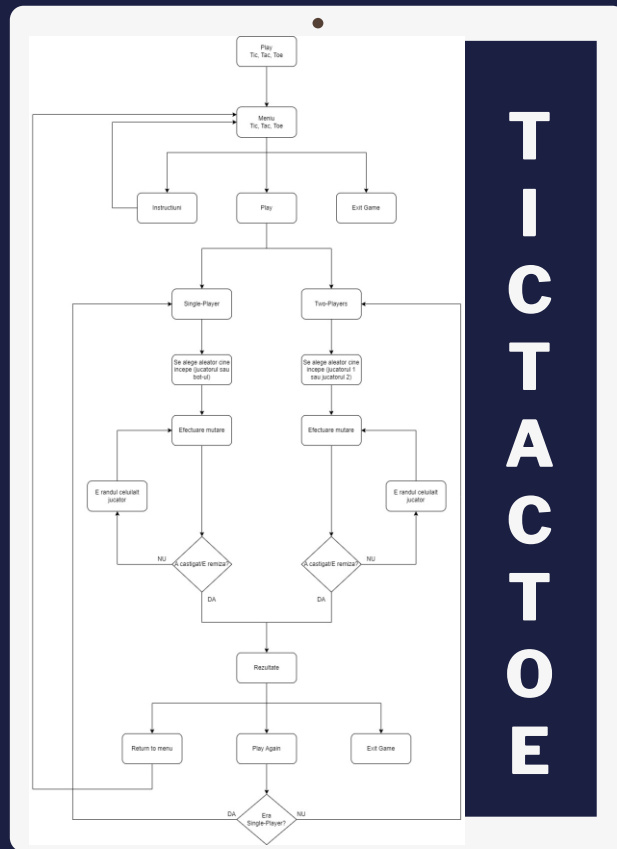
# BOARDBLITZ GAME

La inceputul jocului se selecteaza numarul de jucatori si se introduc nickname-urile participantilor. Fiecare jucator are asociat cate un caracter. Jocul se remarca prin animatiile de mutare a pionilor si animatia zarului.



# DIAGRAME WORKFLOW

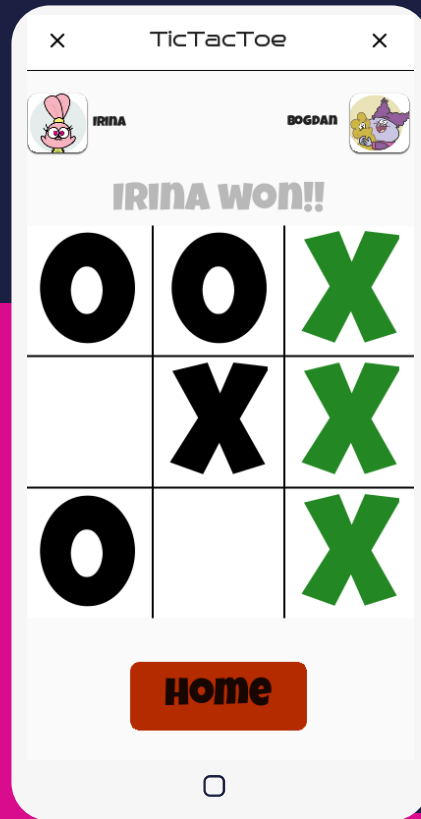
Diagramele contin cinci componente (Main, Word Rush, Headspin, Board Blitz si **Tic Tac Toe**) impartite conform functionalitatilor aplicatiei. Structurarea lor clara permite o buna intelegere a pasilor de implementare.



T  
I  
C  
T  
A  
C  
T  
O  
E

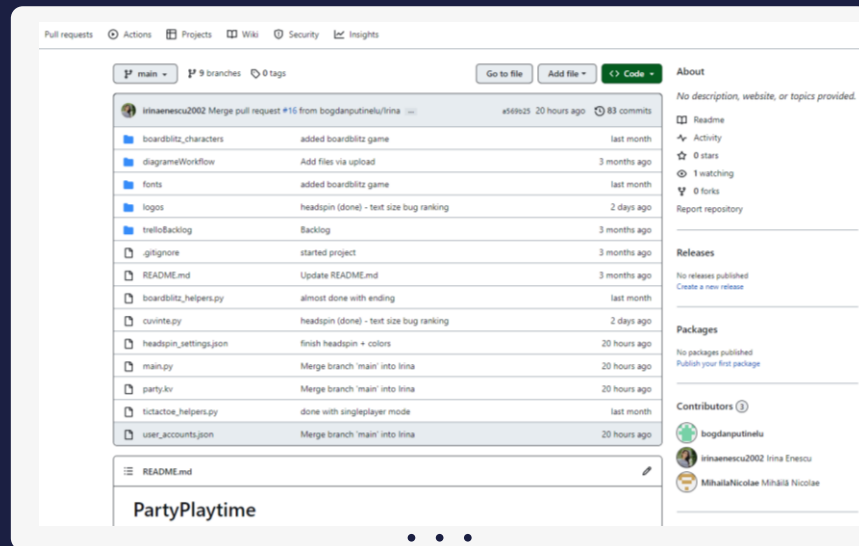
# TICTACTOE GAME

Clasicul joc TicTacToe vine cu o interfata noua si o noua functionalitate: nivele de dificultate pentru jocul Player VS Bot. Bot-ul functioneaza pe baza unei strategii offensive ce creste odata cu dificultatea.



# SOURCE CONTROL

Sincronizarea codului a fost realizata cu ajutorul Git. Am creat mai multe branch-uri (una pentru fiecare membru al echipei). Pe parcurs, am dat pull request-uri pentru a aduce pe main functionalitatile implementate de fiecare si am dat merge la branch-uri. Asa cum se observa in imagine, am avut mai multe commit-uri (83 commit-uri la momentul screenshot-ului).





# CODE STANDARDS

Am respectat un set de reguli și convenții pentru a scrie codul sursă într-un mod coerent și ușor de citit, în conformitate cu standardele industriei. Acest lucru facilitează înțelegerea codului și reduce erorile.

```
1853
1854
1855     def alien_is_pressed(self, alien_id):
1856         screen = self.root.get_screen("boardblitz_game")
1857
1858         alien_moves = PartyPlaytime.boardblitz_miscellaneous[4]
1859         normal_color = PartyPlaytime.boardblitz_miscellaneous[5]
1860         dice_rolled = PartyPlaytime.boardblitz_miscellaneous[6]
1861
1862         color_name = alien_id.split('.')[0]
1863
1864         for alien in alien_moves:
1865             screen.ids[alien].disabled = True
1866             screen.ids[alien].icon_color = normal_color
1867
1868         if aliens.state[alien_id] == 0:
1869             start_box = coordinates[start_boxes[alien_id]]
1870             box_number = start_boxes[alien_id].split('.')[1]
1871             first_corner = "box_" + str(int(box_number) + 4)
1872             second_corner = "box_" + str(int(box_number) + 5)
1873             destination = "box_" + str(int(box_number) + 6)
1874
1875             animation = Animation(
1876                 duration=0.3,
1877                 pos=(dp(start_box["width"]), dp(start_box["height"]))
1878             ) + Animation(
1879                 duration=0.3,
1880                 pos=(dp(coordinates[first_corner]["width"]), dp(coordinates[first_corner]["height"]))
1881             ) + Animation(
1882                 duration=0.2,
1883                 pos=(dp(coordinates[second_corner]["width"]), dp(coordinates[second_corner]["height"]))
1884             ) + Animation(
1885                 duration=0.2,
1886
1887 PartyPlaytime     alien_is_pressed()     else     if (color_name == "red" and cur...     if dice_rolled == 6 and current...     if PartyPlaytime.boardblitz_fin...
```



# CODE STANDARDS

- Am urmat setul de recomandari de stil PEP 8 (Python Enhancement Proposal 8).
- Am folosit denumiri descriptive pentru variabile, funcții și clase. Am folosit snake\_case pentru variabile și funcții (exemplu: alien\_is\_pressed) și PascalCase pentru denumirea claselor (exemplu: GameCardBehaviour).
- Am folosit indentarea consistenta pentru a structura codul (patru spatii pe fiecare nivel).
- Am limitat lungimea liniilor de cod la aproximativ 79-80 de caractere pentru a asigura o vizualizare adecvată pe majoritatea ecranelor.

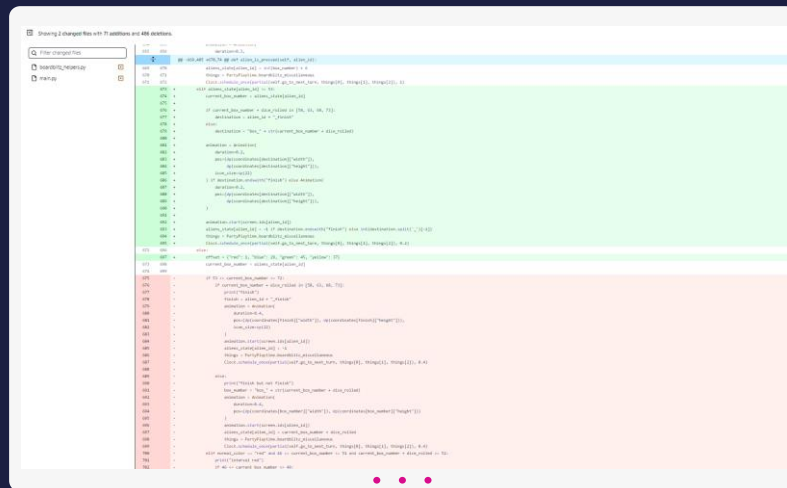


**X +**



# CODE REFACTORING

Acesta a fost refactorizat, fiind eliminate mai multe bucati de cod redundante. Noul cod este unul concis si usor de inteles (<https://github.com/bogdanputinelu/uni-PartyPlaytime/commit/76caf9849976275f1c8eecd845d963031f5356d>)



```
const { Router } = require('express');
const { createParty, updateParty, deleteParty } = require('../controllers/party');
const router = Router();

// Create Party
router.post('/create', createParty);

// Update Party
router.put('/update', updateParty);

// Delete Party
router.delete('/delete', deleteParty);

module.exports = router;
```

# TOOL AI - ChatGPT



## Generare cuvinte

ChatGPT a furnizat  
lista de cuvinte  
pentru jocuri



## Generare cod

ChatGPT a generat  
cateva functii pe care  
le-am integrat in cod



## Bug

ChatGPT ne-a ajutat  
sa rezolvam mici  
erori



# LISTA GENERATA

```
main.py x cuvinte.py x party.kv x
1 words = ["Skydiving",
2         "Tango Dancing",
3         "Baking a Cake",
4         "Playing Tennis",
5         "Painting a Portrait",
6         "Writing a Love Letter",
7         "Doing Yoga",
8         "Building a Sandcastle",
9         "Playing the Guitar",
10        "Making a Collage",
11        "Going Ice Skating",
12        "Playing Charades",
13        "Flying a Kite",
14        "Putting Together a Puzzle",
15        "Playing Chess",
16        "Carving a Pumpkin",
17        "Playing the Piano",
18        "Gardening",
19        "Taking a Photo",
20        "Doing a Scientific Experiment",
21        "Knitting",
22        "Solving a Rubik's Cube",
23        "Bracelet",
24        "Doing Origami",
25        "Playing the Violin",
26        "Writing a Poem",
27        "Making a Smoothie",
28        "Making a Snowman",
29        "Playing Checkers",
```

# REZOLVAREA UNEI ERORI

De ce primesc eroarea 'HeadSpinPlay object has no attribute 'root'?

Eroarea "HeadSpinPlay object has no attribute 'root'" indică faptul că obiectul de tipul 'HeadSpinPlay' nu are atributul 'root'. Acest lucru poate apărea atunci când încercați să accesați atributul 'root' într-un context în care acesta nu este definit.

Atributul 'root' este adesea utilizat în aplicațiile Kivy pentru a accesa obiectul principal al aplicației și pentru a naviga între ecrane. În cazul în care clasa 'HeadSpinPlay' este utilizată ca un ecran într-o aplicație Kivy și doriți să accesați obiectul 'root', asigurați-vă că aveți o referință corectă către instanța aplicației în interiorul clasei.

Pentru a remedia această problemă, puteți adăuga un atribut 'app' în clasa 'HeadSpinPlay' și să îl inițializați cu instanța aplicației înainte de a utiliza metoda 'go\_home\_headspin'. Iată un exemplu:

```
python Copy code

class HeadSpinPlay(MDScreen):
    def __init__(self, **kwargs):
        super(HeadSpinPlay, self).__init__(**kwargs)
        self.app = App.get_running_app()

    def end_game_headspin(self):
        if not self.headspin_ranking_dialog:
            self.headspin_ranking_dialog = MDDialog(
                type="custom",
                content_cls=RankingInformationHeadspin(),
                auto_dismiss=False,
                buttons=[
                    MDFlatButton(
                        text="Regenerate response"
                    ),
                ],
            )
```

Send a message.

# GENERARE COD



Generează o funcție care verifică dacă tabla are un câștigător în cadrul jocului X și O. Informațiile jocului sunt reținute în `PartyPlaytime.tictactoe_board`. Aceasta este o listă cu 9 elemente care au valoarea 0 dacă casuta nu este completată, valoarea 1 dacă casuta conține X și valoarea 2 dacă casuta conține O.



Iată funcția cerută, care verifică dacă tabla are un câștigător în jocul X și O, utilizând informațiile din `PartyPlaytime.tictactoe_board`:



```
python Copy code

def check_winner_tictactoe():
    state = PartyPlaytime.tictactoe_board

    # Verificăm liniile
    for i in range(3):
        if state[3*i + 0] == state[3*i + 1] == state[3*i + 2] != 0:
            return True

    # Verificăm coloanele
    for i in range(3):
        if state[i + 0] == state[i + 3] == state[i + 6] != 0:
            return True

    # Verificăm diagonalele
    if state[0] == state[4] == state[8] != 0:
        return True
    if state[2] == state[4] == state[6] != 0:
        return True

    return False
```

Regenerate response

Send a message.

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT 4o, 04 March 2024.



# GENERARE COD

Pornind de la codul generat de ChatGPT am implementat functia ce verifica daca avem un castigator. Am adaugat animatia aferenta.

```
def check_win_tictactoe(self):
    state = PartyPlaytime.tictactoe_board

    for i in range(3):
        if state[3*i + 0] == state[3*i + 1] and state[3*i + 1] == state[3*i + 2] and state[3*i + 0] != 0:
            self.animate_win(3*i + 0, 3*i + 1, 3*i + 2)
            return True
        if state[i + 0] == state[i + 3] and state[i + 3] == state[i + 6] and state[i + 0] != 0:
            self.animate_win(i + 0, i + 3, i + 6)
            return True

    if state[0] == state[4] and state[4] == state[8] and state[0] != 0:
        self.animate_win(0, 4, 8)
        return True
    if state[2] == state[4] and state[4] == state[6] and state[2] != 0:
        self.animate_win(2, 4, 6)
        return True

    return False
```

# COMENTARII

Comentariile din cod sunt utilizate pentru a adăuga explicații și clarificări cu privire la modul în care funcționează sau ar trebui să funcționeze anumite funcții.

```
# Functie ce se apeleaza automat in ultima runda in momentul in care ultima echipa a ghicit sau nu a ghicit toate  
# cuvintele alocate. Pe baza clasamentului generat de get_ranking() se editeaza dialogul deschis.  
  
def end_game_headspin(self):  
    self.get_ranking()  
    if not self.headspin_ranking_dialog:  
        self.headspin_ranking_dialog = MDDialog(  
            type="custom",  
            content_cls=RankingInformationHeadspin(),  
            auto_dismiss=False,  
            buttons=[  
                MDFlatButton(  
                    text="EXIT",  
                    font_name="Fonts/BrunoAce-Regular.ttf",  
                    font_size="30sp",  
                    on_release=self.go_home_headspin,  
                    pos_hint={"center_x": .5, "center_y": .5}  
                )  
            ]  
        )  
    )  
  
# Se adauga dinamic intr-un container cate o eticheta cu fiecare echipa si scorul corespunzator.  
  
container = MDBoxLayout(orientation='vertical',  
                        spacing='5dp',  
                        padding='10dp',  
                        pos_hint={"center_x": .5, "center_y": .5},  
                        size_hint=0.5  
                    )
```



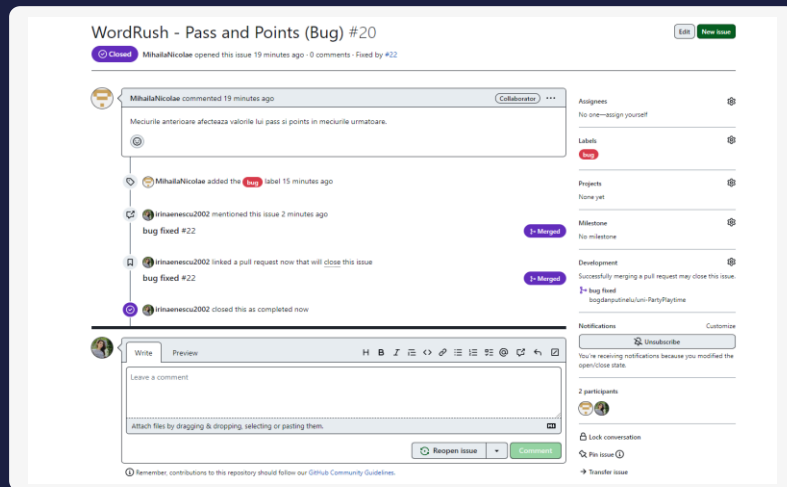
# COMENTARII

- Am adaugat comentarii in cod doar acolo unde sunt necesare si relevante, evitand redundanta si aglomerarea codului. Nu a fost nevoie de foarte multe comentarii deoarece denumirile variabilelor si functiilor sunt sugestive.
- Am respectat stilul, formatarea si identarea corespunzatoare a comentariilor adaugate pentru a le face usor de citit si de urmarit.
- Am folosit comentariile pentru a comunica intre membrii echipei, punand in evidenta observatii sau sugestii utile pentru bucatile de cod ce se aseamana foarte mult.



# RAPORTARE SI REZOLVARE BUG

Am raportat un bug in cadrul functionalitatii unui joc (jocurile anterioare afectau valorile pass & points in jocurile urmatoare) si l-am rezolvat cu pull request.



# TESTE AUTOMATE

- Functia care verifica daca avem un castigator
- Functia care prezice urmatoarea mutare a bot-ului pe dificultatea hard
- Functia care testeaza daca un pion are in fata un blocaj
- Functia care verifica ce pioni pot fi mutati

```
35 @pytest.mark.parametrize(
36     "board_state, expected",
37     [
38         ((0, 0, 0, 0, 0, 0, 0, 0, 0), [-1]),
39         ((1, 1, 0, 0, 2, 1, 0, 0, 0), [0]),
40         ((2, 1, 0, 0, 2, 1, 2, 0, 0), [2, 3, 0]),
41         ((2, 0, 0, 1, 1, 2, 2, 0, 0), [-1]),
42         ((1, 2, 0, 0, 1, 0, 1, 0, 2), [2, 3]),
43         ((1, 0, 2, 2, 1, 1, 0, 0, 2), [-1]),
44         ((1, 0, 0, 1, 2, 2, 0, 0, 1), [0]),
45         ((0, 2, 1, 0, 1, 0, 0, 0, 0), [0]),
46     ]
47 )
48
49 def test_predict_next_move(board_state, expected):
50     app = main.PartyPlayline()
51
52     main.PartyPlayline.tictactoe.board = board_state
53     main.PartyPlayline.tictactoe.difficulty = "hard"
54     assert app.predict_next_move() in expected
55
56 @pytest.mark.parametrize(
57     "test_cases",
58     [
59         # ...
60     ]
61 )
62
63 def test_session_start():
64     # ...
65
66 # Test session summary
67
68 test session starts
69 collecting ... collected 33 items
70
71 automated_tests.py::test.tictactoe_check_win[board_state0-False] PASSED [ 3%]
72 automated_tests.py::test.tictactoe_check_win[board_state1-True] PASSED [ 6%]
73 automated_tests.py::test.tictactoe_check_win[board_state2-False] PASSED [ 9%]
74 automated_tests.py::test.tictactoe_check_win[board_state3-False] PASSED [ 12%]
75 automated_tests.py::test.tictactoe_check_win[board_state4-True] PASSED [ 15%]
76 automated_tests.py::test.tictactoe_check_win[board_state5-True] PASSED [ 18%]
77 automated_tests.py::test.tictactoe_check_win[board_state6-False] PASSED [ 21%]
78 automated_tests.py::test.tictactoe_check_win[board_state7-False] PASSED [ 24%]
79 automated_tests.py::test.tictactoe_check_win[board_state8-True] PASSED [ 27%]
```

# MULTUMIM!



**Bogdan**

Putinelu Bogdan  
Andrei



**Irina**

Enescu Irina  
Stefania



**Nicolae**

Mihaila  
Nicolae

