Blanc manger coco

GROUPE F:

- ALEXANDRE BUSI
- ANTOINE LAUNAY
- ROBIN DEJEAN

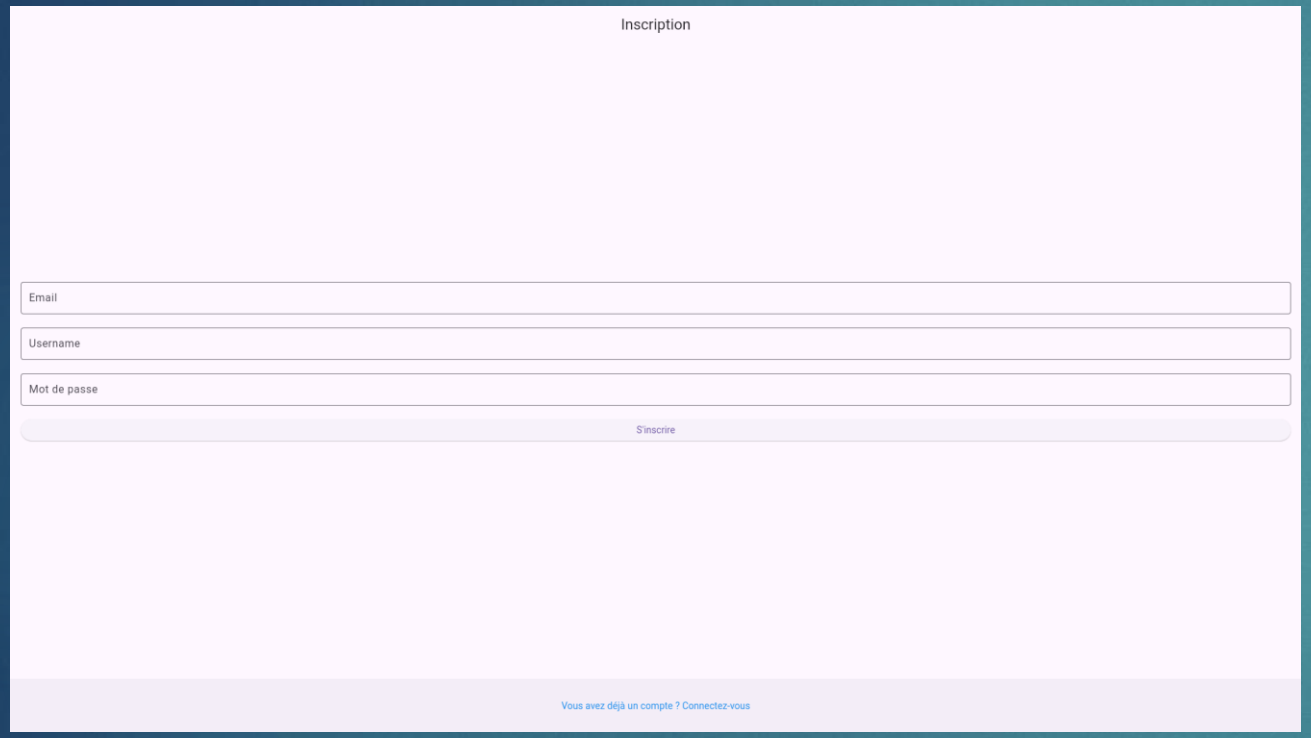
DEROULEMENT

Organisation

Client :
- Robin

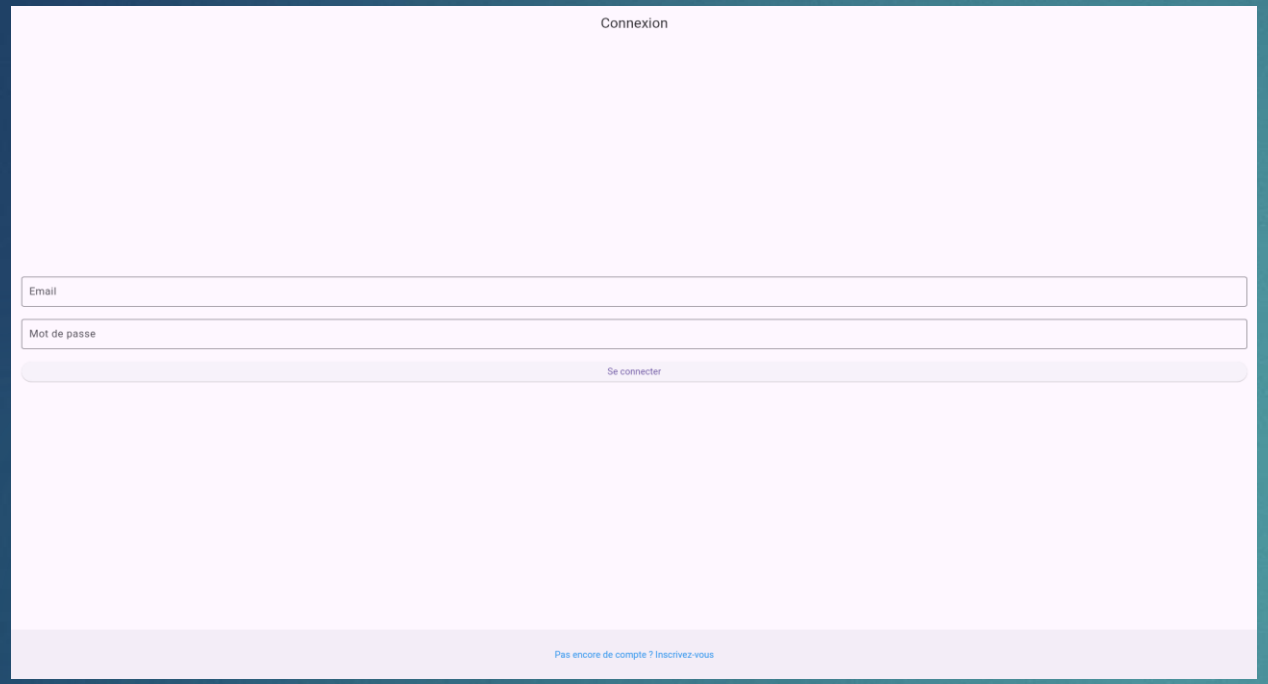
Serveur :
- Antoine
- Alexandre

Inscription / Connexion




```java
@PostMapping("/auth/signup")
public ResponseEntity<String> signup(@RequestBody SignUpRequest request) {
    if(userService.existsByEmail(request.getEmail()))
        return ResponseEntity.ok( body: "Non autorisée");
    return ResponseEntity.ok(authService.signup(request));
}

@PostMapping("/auth/signin")
public ResponseEntity<String> signin(@RequestBody SignInRequest request) {
    return ResponseEntity.ok(authService.signin(request));
}
```

```java
public String signup(SignUpRequest request) {
    request.setPassword(passwordEncoder.encode(request.getPassword()));
    User user = User.builder()
            .email(request.getEmail())
            .username(request.getUsername())
            .password(request.getPassword())
            .role(Role.USER)
            .build();
    userRepository.save(user);
    return jwtService.generateToken(user);
}
```




```java
public class User implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String username;
    private String password;
    private String email;
    @Enumerated(EnumType.STRING)
    private Role role;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return List.of(new SimpleGrantedAuthority(role.name()));
    }
```

```java
public String signin(SignInRequest request) {
    authenticationManager.authenticate(
            new UsernamePasswordAuthenticationToken(request.getEmail(),request.getPassword())
    );
    User user = userRepository.findByEmail(request.getEmail())
            .orElseThrow(() -> new IllegalArgumentException("Invalid email or password"));
    return jwtService.generateToken(user);
}
```

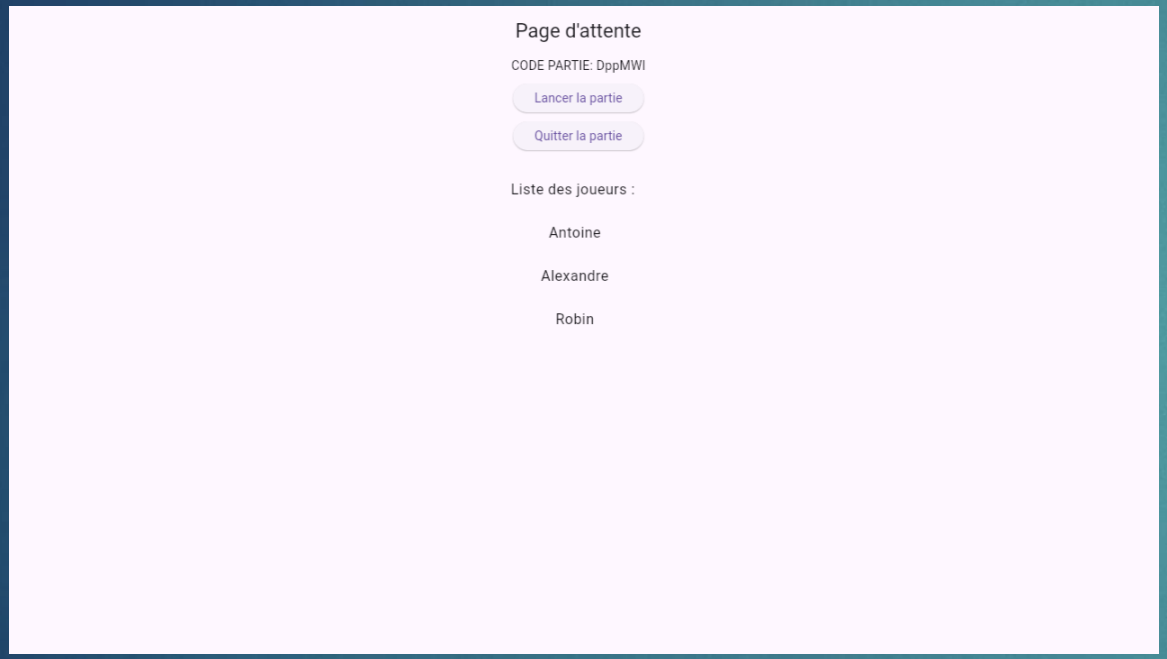
Partie : Connexion / Création


```java
@PostMapping("/join/{gameCode}")
public String joinGame(@PathVariable String gameCode) {
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    Player player = playerService.createPlayer((User) authentication.getPrincipal());
    Game game = (Game) redisTemplate.opsForValue().get("game:"+ gameCode);
    assert game != null;
    assert !game.getPlayers().contains(player.getId());
    if (game.isHasStarted()) return "-1";
    if (game.getPlayers().size()+game.getPlayersIA().size() >= game.getSize()) return "-2";
    game.addPlayer(player.getId());
    redisTemplate.opsForValue().set("game:"+game.getJoinCode(), game);
    messagingTemplate.convertAndSend( destination: "topic/game/"+ gameCode +"/playerlist",gameService.getPlayerDict(game))
    return game.getJoinCode();
}
```


```java
@PostMapping("/create")
public String createGame(@RequestBody createGameRequest request) {
    int taille = Integer.parseInt(request.getTaille());
    if (taille >2 && taille < 6){
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        Player player = playerService.createPlayer((User) authentication.getPrincipal());
        Game game = gameService.createGame(player.getId(),taille);
        game.addPlayer(player.getId());
        redisTemplate.opsForValue().set("game:"+game.getJoinCode(),
                game);
        return game.getJoinCode();
    }
    return "-1";
}
```

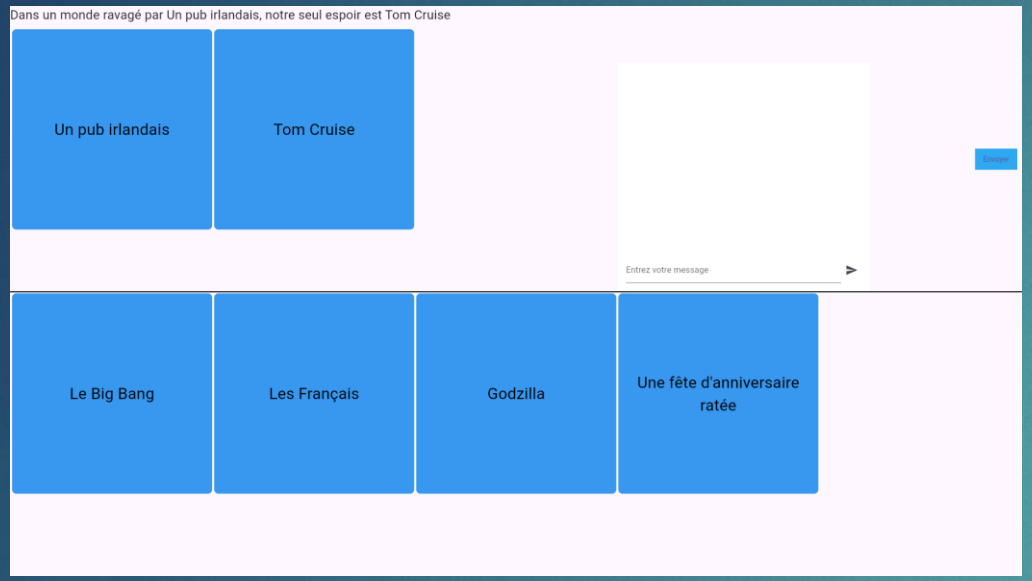
```java
public Game createGame(Long id,Integer size) {
    System.out.println("Creating new game");
    ArrayList<Long> wcards = new ArrayList<>();
    wCardRepository.findAll().forEach(wcard -> wcards.add(wcard.getId()));
    ArrayList<Long> bcards = new ArrayList<>();
    bCardRepository.findAll().forEach(bcard -> bcards.add(bcard.getId()));
    Game game = Game.builder()
            .players(new ArrayList<>())
            .leader(id)
            .joinCode(generateJoinCode())
            .wcards(wcards)
            .bcards(bcards)
            .hasStarted(false)
            .PlayersIA(new ArrayList<>())
            .size(size)
            .build();
    redisTemplate.opsForValue().set("game:"+game.getJoinCode(), game);
    return game;
}
```

Lancement



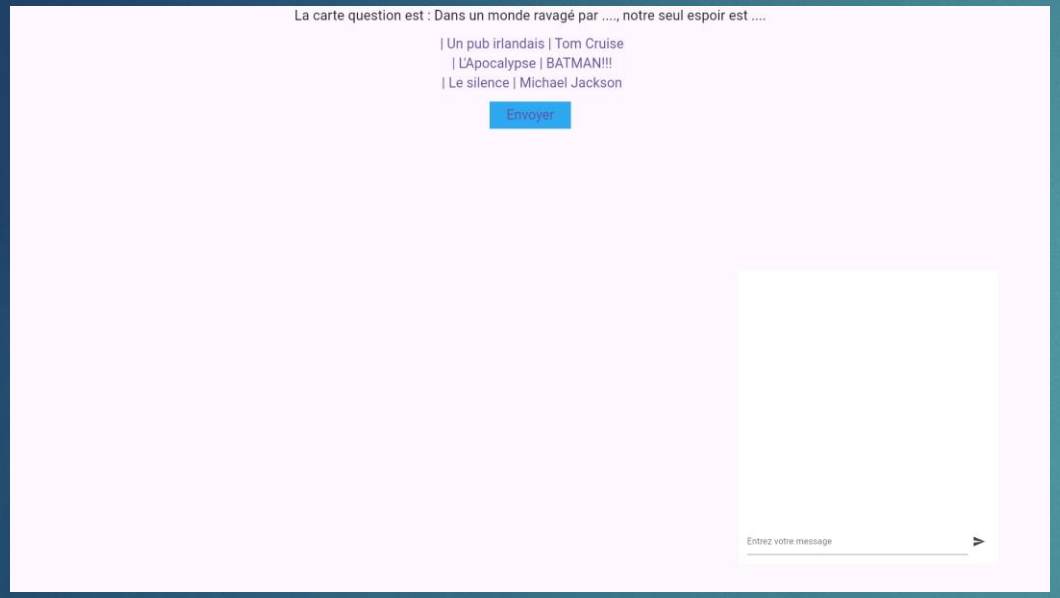

```java
public void startGame(String gameCode) {

    [...]

    for (Long playerId : players) {
        Player player = (Player) redisTemplate.opsForValue().get("player:"+playerId.toString());
        assert player != null;
        ArrayList<Long> playerCards = new ArrayList<>();
        for(int j = 0; j < 7; j++){
            int randomNum = ThreadLocalRandom.current().nextInt( origin: 0, wcards.size());
            playerCards.add(wcards.get(randomNum));
            wcards.remove(randomNum);
        }
        //On initialise les conditions initiales des joueurs
        player.setCards(playerCards);
        CardPlayed cardPlayed = new CardPlayed( id: gameCode +"_"+round.getId().toString()+"_"+player.getId().toString())
        player.setCardPlayed(cardPlayed.getId()); //On initialise le cardplayed pour le round 1


        hmwcards.put(playerId,cardPlayed.getId());

        redisTemplate.opsForValue().set("player:"+ playerId, player);
        redisTemplate.opsForValue().set("cardPlayed:"+cardPlayed.getId(), cardPlayed);

    }

    round.setWCards(hmwcards);
    game.setWcards(wcards);
    game.setRounds(rounds);
    game.setBcards(bcards);

    redisTemplate.opsForValue().set("round:"+round.getId().toString(), round);
    redisTemplate.opsForValue().set("game:"+game.getJoinCode(), game);
}
```

Joueur



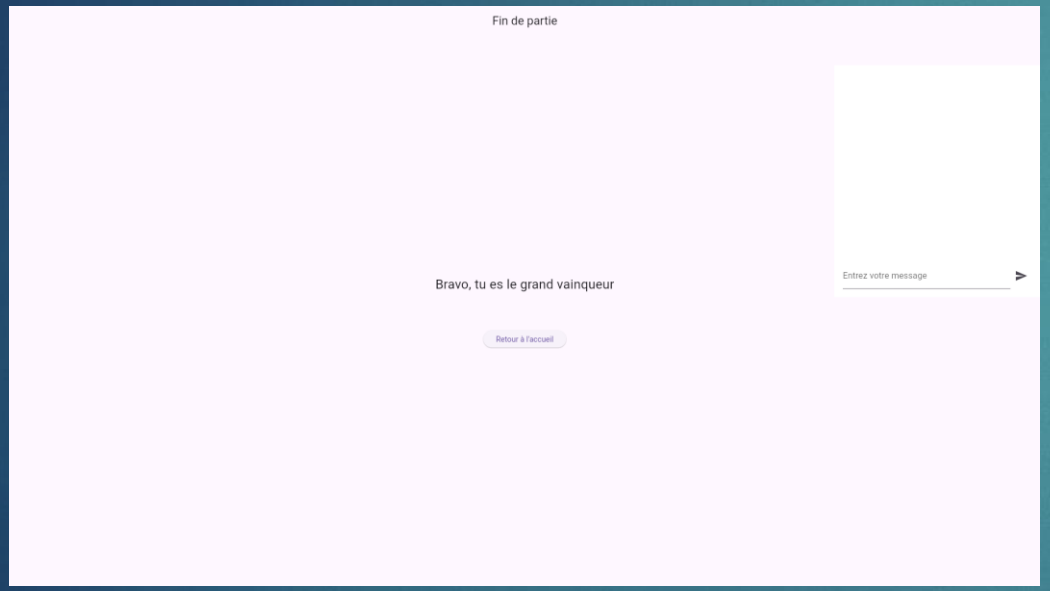

```java
@MessageMapping("/{gameCode}/chooseCard")
public void ChooseCard(@RequestBody Map<Integer,Long> cardsMap, @PathVariable String gameCode) {
    gameService.chooseCard(gameCode,cardsMap, SecurityContextHolder.getContext().getAuthentication().getPrincipal());
    Game game = (Game) redisTemplate.opsForValue().get("game:"+gameCode);
    assert game != null;
    Round round = (Round) redisTemplate.opsForValue().get("round:"+game.getRounds().get(game.getRounds().size() - 1).toString());
    assert round != null;
    if(round.getWCards().size() == game.getPlayers().size()+game.getPlayersIA().size()-1){ //Si tous les joueurs ont envoyé leurs ca
        HashMap<Long,HashMap<Integer, HashMap<String,Object>>> cards = gameService.getALlCardPlayedAsDict(round.getId());
        Long masterid = round.getMasterid();
        if (round.isIsmasterAI()) {
            if (aiServices.ActionIAMaster(masterid, round.getId(), gameCode)) utilsCards.endgame(gameCode,round.getWinnerRoundId());
            else utilsCards.newround(gameCode);
        } else {
            messagingTemplate.convertAndSend( destination: "topic/game/" + gameCode + "needchoose/" + masterid, cards); //envoie au joueu
        }
    }

}
```

Master




```java
@MessageMapping("/{gameCode}/choosewinner")
public void ChooseWinnerRound(@RequestBody Long playerId, @PathVariable String gameCode) {
    Game game = (Game) redisTemplate.opsForValue().get("game:"+gameCode);
    assert game != null;
    Round round = (Round) redisTemplate.opsForValue().get("round:"+game.getRounds().get(game.getRounds().size() - 1).toString());
    assert round != null;
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    Player player1 = playerService.createPlayer((User) authentication.getPrincipal());
    assert player1 != null;
    assert Objects.equals(player1.getId(), round.getMasterid());
    if ((game.getPlayers().contains(playerId) || game.getPlayersIA().contains(playerId) )&& !Objects.equals(playerId, round.getMasterid())){
        Player player = playerService.getPlayer(playerId.toString());
        assert player != null;
        player.setScore(player.getScore()+1);
        redisTemplate.opsForValue().set("player:"+player.getId().toString(), player);
        round.setCardWinnerId(player.getCardPlayed());
        redisTemplate.opsForValue().set("round:"+round.getId().toString(), round);
        if (player.getScore()==game.getEndScore()){ //On verifie si la partie se finit
            gameService.endGame(game.getJoinCode(),player.getId()); //On appelle la fonction qui va set le winner et fermer la partie
        } else {
            gameService.newRound(gameCode); //On appelle la méthode qui va gérer le lancement du nouveau round
        }
    }
}
```

Chat / Fin




```java
@MessageMapping("/{gameCode}/chat")
public String chatroom(@RequestBody String text, @PathVariable String gameCode){
    Game game = (Game) redisTemplate.opsForValue().get("game:"+gameCode);
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    Player player = playerService.createPlayer((User) authentication.getPrincipal());
    assert game != null;
    ArrayList<String> chat = game.getChat();
    game.setChat(chat);
    redisTemplate.opsForValue().set("game:"+gameCode, game);
    return "["+player.getUsername()+"] : " + text;
}
```

Intelligence Artificielle

```java
public class Player extends User implements Serializable {

    private int score;

    @ElementCollection
    private ArrayList<Long> cards;
    private String cardPlayed;

    public Player(User user) {
        super(user.getId(), user.getEmail(), user.getPasswor
    }

}
```

```java
@Setter
@Getter
@Entity
@RequiredArgsConstructor
public class AIPlayer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;
    private Long copy=-1L;
    private int score=0;
    @ElementCollection
    private ArrayList<Long> cards;
    private String cardPlayed;
}
```

```java
public void ActionIAJoueur(Long iaId, Long roundid){ //Fonction qui génère l'action de l'IA

    [...]

    wcards.forEach(wcardid->{
        CaseMatrice caseM = caseMatriceRepository.getReferenceById(bcardid+"_"+wcardid);
        cardValues.putIfAbsent(wcardid, caseM.getValue());
    });
    ArrayList<Long> wcardsUsed = new ArrayList<>();
    Long value;
    for (int i=0;i<bCard.getNbBlank();i++){ //On récupère la/les meilleures cartes de la
        value = getMaxofDict(cardValues);
        wcardsUsed.add(value);
        cardValues.remove(value);
    }



    HashMap<Integer,Long> cardsp = cardPlayed.getCards(); //On crée la hashmap pour les
    for (int i = 0;i<wcardsUsed.size();i++){
        cardsp.put(i,wcardsUsed.get(i));
    }
    cardPlayed.setCards(cardsp);
    iaPlayer.setCards(new ArrayList<>(cardValues.keySet()));
    redisTemplate.opsForValue().set("cardplayed:"+cardPlayed.getId(),cardPlayed);
```

```java
public void calculIa(Game game){ //Fonction qui calcule la nouvelle valeur des caseMatrice

    [...]

    for (Long wcardid : cards.values()) {
        caseMatriceRepository.findById(wcardid + "_" + bCardid).ifPresent(caseMatrice -> {
            if (Objects.equals(pid, round.getWinnerRoundId())) { //Si les cartes jouées sor
                caseMatrice.setValue(min( a: caseMatrice.getValue() * 1.1, b: 1.0)); //On au
            } else {
                caseMatrice.setValue(max( a: caseMatrice.getValue() * 0.9, b: 0.0)); //On ba
            }
            caseMatriceRepository.save(caseMatrice);
        });
    }
```

Conclusion

Améliorations :

- Meilleure Organisation
- Proposition de cartes par les Joueurs
- Liste de Parties Publiques
- Système de Tags
- IA plus précise