

## Chess Heroes

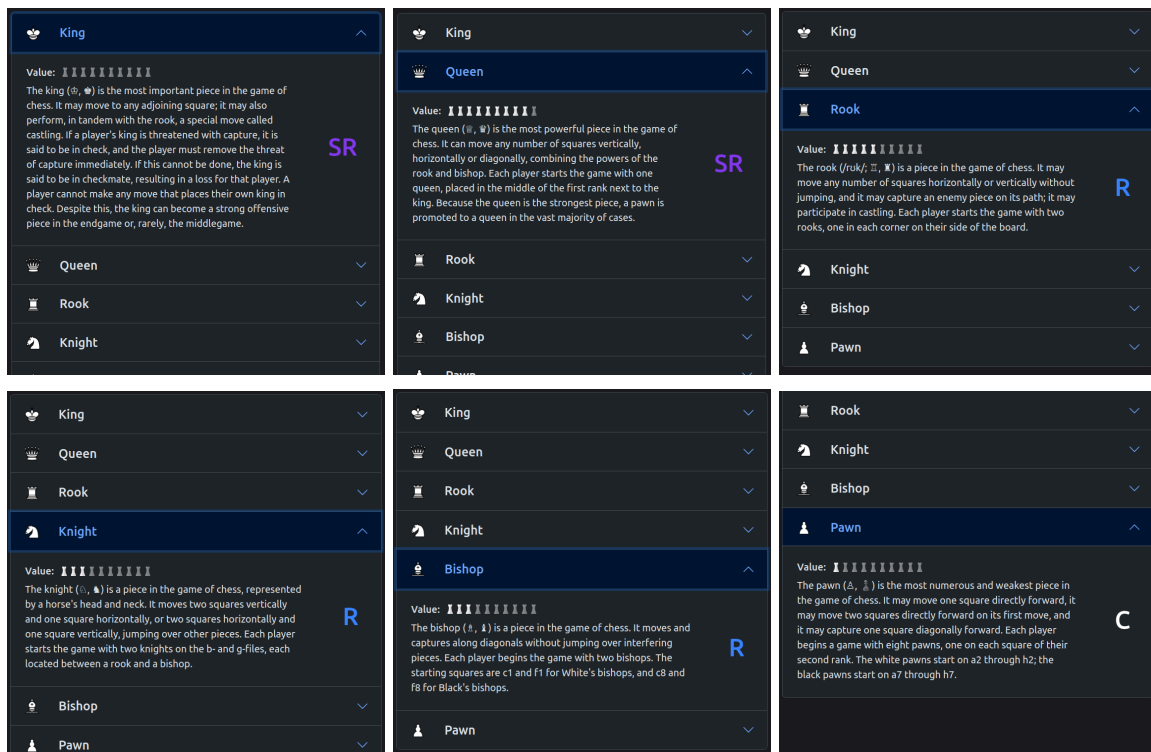
Advanced software engineering course.  
ICT Solutions Architect Master's Degree

This microservice project implements functionalities to enable users to buy gachas, in this case chess pieces. Users can buy single pieces via auctions or pulling from banners. Auctions are time limited and can be created by users.

### Group components:

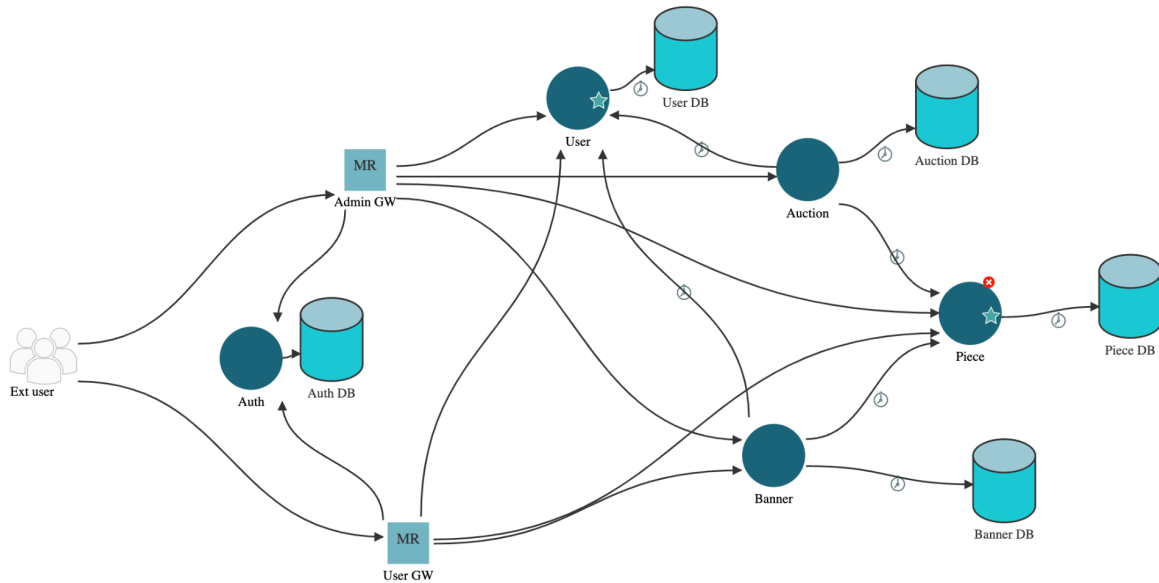
- Deri Gabriele
- Turchetti Gabriele

## Gacha Overview



## Architecture

All of our microservices use Python as a development language and SQLite3 as a persistent database. Below is reported a scheme of our architecture.



### Gateways:

- **Admin**
  - This gateway is used to receive requests for admin actions. Before rerouting the request to the appropriate service, the requestor is validated with the auth service.
- **User**
  - This gateway is used to receive requests for user actions. Before rerouting the request to the appropriate service, the requestor is validated with the auth service.

Gateways are the entry point to our application as microservices are not exposed to external connections to prevent unauthorized access.

## Microservices:

- **Auction**
  - This service is responsible for the creation, modification, and bidding on a specific auction
  - **Connections:**
    - **User:** To update the collection of a user, check if a user has a piece before creating an auction and check if the user has enough tokens to bid on an auction
    - **Auth:** To validate if the user who wants to create a new auction is effectively the one in the presented JWT
- **Auth**
  - This service is responsible for the authentication and authorization of existing and new users on our platform Chess Heroes
  - **Connections:**
    - **User:** To add the new user in the user DB after successful user creation and to delete a user account when requested
- **Banner**
  - This service is responsible for retrieving info, creating, updating, deleting, and performing actions of/on the banners
  - **Connections:**
    - **Piece:** To retrieve the info about the pieces
    - **Auth:** To validate if the user who wants to pull from a banner is effectively the one in the presented JWT
    - **User:** To update the collection of the user
- **Piece**
  - This service is responsible for retrieving info, creating, updating, and deleting pieces
- **User**
  - This service is responsible for the management of user profile, user collection, and token balance management
  - **Connections:**
    - **Auth:** To
      - Update a username when a user modifies it and maintain data consistency between services.
      - Check if the provided user id is the same as the one in the JWT before updating a user collection
      - Modify a user when an admin requires it. This to maintain data consistency

## User Stories

### Player user stories:

- Create my game account/profile SO THAT I can participate in the game
  - **/create\_user (user-user DB / auth-auth DB, gateways)**
- Delete my game account/profile SO THAT I can stop participating to the game
  - **/player/<int:player\_id> (user-user DB / auth-auth DB, gateways)**
- Modify my account/profile SO THAT I can personalize my account/profile
  - **/player/<int:player\_id> (user-user DB, gateways) calls auth for consistency**
  - **user/modify/<int:player\_id> (auth-auth DB, gateways)**
- Login and logout from the system SO THAT can access and leave the game
  - **logout/<int:player\_id> (auth-auth DB, gateways)**
- Be safe about my account/profile data SO THAT nobody can enter in my account and steal/modify my info
  - **/authorize /introspect (auth-auth DB, gateways)**
- See my gacha collection SO THAT i know how many gacha i need to complete the collection
  - **/player/collection/<int:player\_id> (user-user DB, gateways)**
- Buy in-game currency SO THAT I can have more chances to win auctions
  - **/player/gold/<int:player\_id> (user-user DB, gateways)**
- Be safe about the in-game currency transactions SO THAT my in-game currency is not wasted or stolen
  - **/player/gold/<int:player\_id> /authorize (user-user DB, gateways, auth-auth DB)**
- See the auction market SO THAT I can evaluate if buy/sell a gacha
  - **/running/all (auction-auction DB, gateways)**
- Set an auction for one of my gacha SO THAT I can increase in-game currency
  - **/create\_auction (auction-auction DB)**
  - **/introspect (auth-auth DB)**
  - **/user/has\_piece (user-user DB)**

- Bid for a gacha from the market SO THAT I can increase my collection
  - **/bid/<int:auction\_id> (auction-auction DB / auth-auth DB)**
  - **/user/balance (user-user DB)**
  
- View my transaction history SO THAT I can track my market movement
  - **/admin/player/transaction/history/<int:player\_id> (user-user DB)**
  
- Receive a gacha when I win an auction SO THAT only I have the gacha I bid for
  - **/update\_collection (user-user DB)**
  
- Receive in-game currency when someone wins my auction SO THAT the gacha sell works as I expected
  - **/update\_collection (user-user DB)**
  
- Receive my in-game currency back when I lose an auction SO THAT my in-game currency is decreased only when I buy something
  - **/update\_collection (user-user DB)**
  
- That the auctions cannot be tampered with SO THAT my in-game currency and collection are safe
  - **/bid/<int:auction\_id> (auction-auction DB / auth-auth DB)**
  
- AS A player I WANT TO get info about a banner SO THAT i can choose in which pull
  - **/banner/banner/<banner\_id> (GET)**
  
- AS A player I WANT TO use in-game currency to roll a gacha SO THAT i can increase my collection:
  - **/banner/banner/pull/<banner\_ID> (GET)**
  
- AS A player I WANT TO see the info of a gacha of my collection SO THAT i can see all info of one of my gacha
  - **/piece/piece?id=<piece\_id> (GET)**
  
- AS A player I WANT TO see the system gacha collection SO THAT i know what i miss of my collection
  - **/piece/piece/all (GET)**
  
- AS A player I WANT TO see the info of a system gacha SO THAT i can see the info of a gacha i miss
  - **/piece/piece?id=<piece\_id> (GET)**

- AS A player I WANT TO have a beautiful graphic user interface SO THAT i fell motivated to play everyday

#### **Admin user stories:**

- AS AN administrator I WANT TO check all the gacha collection SO THAT i can check all the collection
  - **/piece/piece/all**
- AS AN administration I WANT TO modify the gacha collection SO THAT i can add/remove gachas
- AS AN administrator I WANT TO modify a specific gacha information SO THAT i can modify the status of a gacha
  - **/piece/piece/<piece\_id> (PUT, DELETE)**
- AS AN administrator I WANT TO check a specific gacha SO THAT i can check the status of a gacha
  - **/piece/piece/<piece\_id> (GET)**
- AS AN administrator I WANT TO add, update, delete a banner SO THAT i can modify the system banners
  - **/banner/banner (POST)**
  - **/banner/banner (PUT, DELETE)**

### **Market Rules**

To bid on an auction use the bid endpoint in the auction module. Users can bid if:

- Time of received request is lower than the auction end timestamp
- Bidder is different from the auction creator
- Bidder can only place one bid, two consecutive bids are not permitted
- Bidder can only bid if he has available balance in the account.
- Balance is deducted at auction end only to the winner. This is consistent and safe as the maximum refillable balance is set low to 50 token per refill

### **Testing**

#### **Integration Testing**

For integration testing, as written in the get started in the README, it is necessary to have a clean environment / DB. This is because tests use the same user info when creating a new user, positive tests would return an error given that the username and user info are persisted between different runs. So make sure to clean the DBs as written in the README.

## Unit Testing

### - Auth, User, Auction

- The following services were copied and calls to external services were mocked. We simply inserted a flag variable **TESTING = True**. We put all the tests in the tests/ folder of each microservice. Used databases are different from the production ones. This is that tests insert data that is not relevant to the real users but it is finalized to testing.
- To mock external services calls we put some values of the received parameters in the request body/URL that simulate an error, and others that return a positive value. This was done to simulate success and error messages.

### - Banner

- The response values of external services are mocked by a function that returns a valid response for every request made.

### - Piece

- This service doesn't call any external service.

## Performance Testing:

The performance tests are made calling the endpoints that don't introduce errors after several calls (e.g. delete requests aren't made due to the impossibility of some requests to restore the initial state of the system). The instructions for those tests are reported in the "docs" folder, along with the information about the distribution test (in the client site are shown the rates for every banner in a clear way).

## Data Security

In the authentication service, using UserRegistrationForm we sanitized username, email and password. Using regex we sanitized username content to prevent SQL injection when inserting data in the database. These values are stored in the authentication service. Username and user id are also sent and persisted in the user database to perform checks, for example when modifying an username, and are kept synchronized between authentication and user to have a consistent state between services.

For data at rest we hashed the user password using a salt. When authenticating an user, this hash is matched with the calculated hash of the received plaintext password.

## Authentication Authorization

For authentication and authorization we used a centralized service: auth. Requests reach the gateway, the gateway checks if the route to which it should redirect the request requires

authentication, if it does and the user provides a bearer token in the header, the request is forwarded, else the gateway raises an error telling the requester that login is required. All routes are protected by authentication.

Additionally, in the auth service, a dictionary is kept to store **route:permissions**. Permissions is an array of int, containing 0,1. These values represent which kind of user is enabled to get rerouted to the appropriate endpoint based on their permissions. This check is performed when the /authorize endpoint is called to check if the requester is enabled to call the requested endpoint.

For the sake of this project, keys are stored in a json configuration file. This is not how keys would be stored in a production environment.

## FLOW

1. User creates an account, username, email, password are stored in the DB
2. User logs in, sends credentials, credentials are validated. If valid a JWT is generated and signed using the secret key of the auth service. In the JWT we store data to identify a user. The response contains the token, refresh token, user\_id and user info
3. The token is used to validate each request, before creating the rerouting path.

We also implement the **/introspect** endpoint to validate the requesting user to see if the user\_id is the same as the one in the JWT to prevent users tampering with other users' accounts without having the correct JWT.

Additionally, the **/authorize** endpoint, instead checks for user authorization using the **route:permissions** mapping to authorize a user before redirecting to the correct service. Below we report the payload of the JWT:

```
def generate_access_token(user_data):
    now = datetime.datetime.now(datetime.UTC)

    payload = {
        "iss": app.config['ISSUER'],
        "sub": str(user_data['user_id']),
        "aud": app.config['AUDIENCE'],
        "exp": now + datetime.timedelta(minutes=app.config['ACCESS_TOKEN_EXPIRE_MINUTES']),
        "iat": now,
        "jti": str(uuid.uuid4()),
        "user_type": int(user_data['user_type']),
        "role": VALID_USER_TYPES[int(user_data['user_type'])],
        "scope": None
    }
    return jwt.encode(payload, app.config['JWT_SECRET_KEY'], algorithm=app.config['ALGORITHM'])
```

Finally we have the **/userinfo** endpoint which enables users (or admins) to get information about their account. All these endpoints enable us to respect the security specs requested in the implementation security requirements.



## Security analysis

**Bandit** did not report errors when building with docker compose up. We registered some security errors when using requests with https and verify set to False. This is needed because we are using self signed certificates. To fix this we simply put **# nsec** as comment on the line of the request, so that bandit does not raise an error and the build correctly functions. There are other lines marked as **# nsec** where bandit found vulnerabilities about SQL Injection, but in those cases SQL Injection is not possible (if u see the code it's clear why).

```
1555 #80 [user_gateway 6/8] RUN bandit -r .
1557 #80 0.530 [main] INFO profile include tests: None
1558 #80 0.520 [main] INFO profile exclude tests: None
1559 #80 0.521 [main] INFO cli include tests: None
1560 #80 0.521 [main] INFO cli exclude tests: None
1561 #80 0.521 [main] INFO running on Python 3.12.8
1562 #80 0.533 Run started:2024-12-05 09:09:45.078181
1563 #80 0.533
1564 #80 0.533 Test results:
1565 #80 0.533 No issues identified.
1566 #80 0.533
1567 #80 0.533 Code scanned:
1568 #80 0.533 Total lines of code: 88
1569 #80 0.533 Total lines skipped (#nsec): 1
1570 #80 0.533 Total potential issues skipped due to specifically being disabled (e.g., #nsec BXXX): 0
1571 #80 0.533
1572 #80 0.533 Run metrics:
1573 #80 0.533 Total issues (by severity):
1574 #80 0.533 Undefined: 0
1575 #80 0.533 Low: 0
1576 #80 0.533 Medium: 0
1577 #80 0.533 High: 0
1578 #80 0.533 Total issues (by confidence):
1579 #80 0.533 Undefined: 0
1580 #80 0.533 Low: 0
1581 #80 0.533 Medium: 0
1582 #80 0.533 High: 0
1583 #80 0.533 Files skipped (0):
1584 #80 DONE 0.6s
```

```
1520 #79 [admin_gateway 6/8] RUN bandit -r .
1522 #79 0.529 [main] INFO profile include tests: None
1523 #79 0.529 [main] INFO profile exclude tests: None
1524 #79 0.529 [main] INFO cli include tests: None
1525 #79 0.529 [main] INFO cli exclude tests: None
1526 #79 0.529 [main] INFO running on Python 3.12.8
1527 #79 0.547 Run started:2024-12-05 09:09:45.140990
1528 #79 0.547
1529 #79 0.547 Test results:
1530 #79 0.547 No issues identified.
1531 #79 0.547
1532 #79 0.547 Code scanned:
1533 #79 0.547 Total lines of code: 111
1534 #79 0.547 Total lines skipped (#nsec): 1
1535 #79 0.547 Total potential issues skipped due to specifically being disabled (e.g., #nsec BXXX): 0
1536 #79 0.547
1537 #79 0.547 Run metrics:
1538 #79 0.547 Total issues (by severity):
1539 #79 0.547 Undefined: 0
1540 #79 0.547 Low: 0
1541 #79 0.547 Medium: 0
1542 #79 0.547 High: 0
1543 #79 0.547 Total issues (by confidence):
1544 #79 0.547 Undefined: 0
1545 #79 0.547 Low: 0
1546 #79 0.547 Medium: 0
1547 #79 0.547 High: 0
1548 #79 0.547 Files skipped (0):
1549 #79 DONE 0.6s
```

```
1164 #63 [lucian 6/6] RUN bandit -r .
1165 #63 0.405 [main] INFO profile include tests: None
1166 #63 0.405 [main] INFO profile exclude tests: None
1167 #63 0.405 [main] INFO cli include tests: None
1168 #63 0.405 [main] INFO cli exclude tests: None
1169 #63 0.406 [main] INFO running on Python 3.12.8
1170 #63 0.519 Run started:2024-12-05 09:09:31.953237
1171 #63 0.519
1172 #63 0.519 Test results:
1173 #63 0.519 No issues identified.
1174 #63 0.519
1175 #63 0.519 Code scanned:
1176 #63 0.519 Total lines of code: 1202
1177 #63 0.519 Total lines skipped (#nsec): 9
1178 #63 0.519 Total potential issues skipped due to specifically being disabled (e.g., #nsec BXXX): 0
1179 #63 0.519
1180 #63 0.519 Run metrics:
1181 #63 0.519 Total issues (by severity):
1182 #63 0.519 Undefined: 0
1183 #63 0.519 Low: 0
1184 #63 0.519 Medium: 0
1185 #63 0.519 High: 0
1186 #63 0.519 Total issues (by confidence):
1187 #63 0.519 Undefined: 0
1188 #63 0.519 Low: 0
1189 #63 0.519 Medium: 0
1190 #63 0.519 High: 0
1191 #63 0.519 Files skipped (0):
1192 #63 DONE 0.6s
```

```
1109 #58 [auth 6/8] RUN bandit -r .
1110 #58 1.076 [main] INFO profile include tests: None
1111 #58 1.076 [main] INFO profile exclude tests: None
1112 #58 1.076 [main] INFO cli include tests: None
1113 #58 1.076 [main] INFO cli exclude tests: None
1114 #58 1.076 [main] INFO running on Python 3.12.8
1115 #58 1.405 Run started:2024-12-05 09:09:29.378306
1116 #58 1.405
1117 #58 1.405 Test results:
1118 #58 1.405 No issues identified.
1119 #58 1.405
1120 #58 1.405 Code scanned:
1121 #58 1.405 Total lines of code: 1605
1122 #58 1.405 Total lines skipped (#nsec): 9
1123 #58 1.405 Total potential issues skipped due to specifically being disabled (e.g., #nsec BXXX): 0
1124 #58 1.405
1125 #58 1.405 Run metrics:
1126 #58 1.405 Total issues (by severity):
1127 #58 1.405 Undefined: 0
1128 #58 1.405 Low: 0
1129 #58 1.405 Medium: 0
1130 #58 1.405 High: 0
1131 #58 1.405 Total issues (by confidence):
1132 #58 1.405 Undefined: 0
1133 #58 1.405 Low: 0
1134 #58 1.405 Medium: 0
1135 #58 1.405 High: 0
1136 #58 1.405 Files skipped (0):
1137 #58 DONE 1.5s
1138
```

```
1034 #51 [user 6/8] RUN bandit -r .
1035 #51 1.127 [main] INFO profile include tests: None
1036 #51 1.127 [main] INFO profile exclude tests: None
1037 #51 1.127 [main] INFO cli include tests: None
1038 #51 1.127 [main] INFO cli exclude tests: None
1039 #51 1.127 [main] INFO running on Python 3.12.8
1040 #51 1.449 Run started:2024-12-05 09:09:24.001115
1041 #51 1.449
1042 #51 1.449 Test results:
1043 #51 1.449 No issues identified.
1044 #51 1.449
1045 #51 1.449 Code scanned:
1046 #51 1.449 Total lines of code: 1518
1047 #51 1.449 Total lines skipped (#nsec): 4
1048 #51 1.449 Total potential issues skipped due to specifically being disabled (e.g., #nsec BXXX): 0
1049 #51 1.449
1050 #51 1.449 Run metrics:
1051 #51 1.449 Total issues (by severity):
1052 #51 1.449 Undefined: 0
1053 #51 1.449 Low: 0
1054 #51 1.449 Medium: 0
1055 #51 1.449 High: 0
1056 #51 1.449 Total issues (by confidence):
1057 #51 1.449 Undefined: 0
1058 #51 1.449 Low: 0
1059 #51 1.449 Medium: 0
1060 #51 1.449 High: 0
1061 #51 1.449 Files skipped (0):
1062 #51 DONE 1.7s
1063
```

```
1000 #49 [piece 14/14] RUN bandit -r .
1001 #49 0.946 [main] INFO profile include tests: None
1002 #49 0.946 [main] INFO profile exclude tests: None
1003 #49 0.946 [main] INFO cli include tests: None
1004 #49 0.946 [main] INFO cli exclude tests: None
1005 #49 0.946 [main] INFO running on Python 3.12.8
1006 #49 1.014 Run started:2024-12-05 09:09:22.064790
1007 #49 1.014
1008 #49 1.014 Test results:
1009 #49 1.014 No issues identified.
1010 #49 1.014
1011 #49 1.014 Code scanned:
1012 #49 1.014 Total lines of code: 223
1013 #49 1.014 Total lines skipped (#nsec): 3
1014 #49 1.014 Total potential issues skipped due to specifically being disabled (e.g., #nsec BXXX): 0
1015 #49 1.014
1016 #49 1.014 Run metrics:
1017 #49 1.014 Total issues (by severity):
1018 #49 1.014 Undefined: 0
1019 #49 1.014 Low: 0
1020 #49 1.014 Medium: 0
1021 #49 1.014 High: 0
1022 #49 1.014 Total issues (by confidence):
1023 #49 1.014 Undefined: 0
1024 #49 1.014 Low: 0
1025 #49 1.014 Medium: 0
1026 #49 1.014 High: 0
1027 #49 1.014 Files skipped (0):
1028 #49 DONE 1.1s
```

```

967 #51 [user 6/6] RUN bandit -r .
968 #51 ...
969
970 #52 [banner 14/14] RUN bandit -r .
971 #52 0.677 [main] INFO profile include tests: None
972 #52 0.677 [main] INFO profile exclude tests: None
973 #52 0.677 [main] INFO cli include tests: None
974 #52 0.677 [main] INFO cli exclude tests: None
975 #52 0.678 [main] INFO running on Python 3.12.8
976 #52 0.811 Run started:2024-12-05 09:09:22.690968
977 #52 0.811
978 #52 0.811 Test results:
979 #52 0.811 No issues identified.
980 #52 0.811
981 #52 0.811 Code scanned:
982 #52 0.811 Total lines of code: 352
983 #52 0.811 Total lines skipped (#nosec): 0
984 #52 0.811 Total potential issues skipped due to specifically being disabled (e.g., #nosec BXXX): 0
985 #52 0.811
986 #52 0.811 Run metrics:
987 #52 0.811 Total issues (by severity):
988 #52 0.811 Undefined: 0
989 #52 0.811 Low: 0
990 #52 0.811 Medium: 0
991 #52 0.811 High: 0
992 #52 0.811 Total issues (by confidence):
993 #52 0.811 Undefined: 0
994 #52 0.811 Low: 0
995 #52 0.811 Medium: 0
996 #52 0.811 High: 0
997 #52 0.811 Files skipped (0):
998 #52 DONE 0.9s

```

## Pip-audit analysis:

```

948
949 #46 [banner 13/14] RUN pip-audit --fix
950 #46 2.802 No known vulnerabilities found
951 #46 DONE 2.9s
952
953 #47 [auth 5/8] RUN pip-audit --fix
954 #47 ...
955
956 #48 [piece 13/14] RUN pip-audit --fix
957 #48 2.943 No known vulnerabilities found
958 #48 DONE 3.1s
959
960 #49 [piece 14/14] RUN bandit -r .
961 #49 ...
962
963 #50 [user 5/8] RUN pip-audit --fix
964 #50 2.427 No known vulnerabilities found
965 #50 DONE 2.5s
966

```

```

1160 #59 [auction 5/8] RUN pip-audit --fix
1161 #59 2.745 No known vulnerabilities found
1162 #59 DONE 2.8s

```

```

1075
1076 #47 [auth 5/8] RUN pip-audit --fix
1077 #47 6.005 Name      Version ID      Fix Versions Applied Fix
1078 #47 6.005 -----
1079 #47 6.005 requests 2.31.0 GHSA-9wx4-h78v-vm56 2.32.0 Successfully upgraded requests (2.31.0 => 2.32.0)
1080 #47 6.005 werkzeug 3.0.1 GHSA-2g68-c3qc-8985 3.0.3 Successfully upgraded werkzeug (3.0.1 => 3.0.6)
1081 #47 6.005 werkzeug 3.0.1 GHSA-f9vj-2wh5-fj8j 3.0.6 Successfully upgraded werkzeug (3.0.1 => 3.0.6)
1082 #47 6.005 werkzeug 3.0.1 GHSA-q34m-jh98-gwm2 3.0.6 Successfully upgraded werkzeug (3.0.1 => 3.0.6)
1083 #47 6.039 Found 4 known vulnerabilities in 2 packages and fixed 4 vulnerabilities in 2 packages
1084 #47 DONE 6.2s
1085


```

```

1512 #77 [user_gateway 5/8] RUN pip-audit --fix
1513 #77 2.445 Found 1 known vulnerability in 1 package and fixed 1 vulnerability in 1 package
1514 #77 2.446 Name      Version ID      Fix Versions Applied Fix
1515 #77 2.446 -----
1516 #77 2.446 requests 2.31.0 GHSA-9wx4-h78v-vm56 2.32.0 Successfully upgraded requests (2.31.0 => 2.32.0)
1517 #77 DONE 2.6s
1518
1519 #78 [admin_gateway 5/8] RUN pip-audit --fix
1520 #78 2.461 Found 1 known vulnerability in 1 package and fixed 1 vulnerability in 1 package
1521 #78 2.463 Name      Version ID      Fix Versions Applied Fix
1522 #78 2.463 -----
1523 #78 2.463 requests 2.31.0 GHSA-9wx4-h78v-vm56 2.32.0 Successfully upgraded requests (2.31.0 => 2.32.0)
1524 #78 DONE 2.6s
1525

```

Regarding **docker scout**, the only vulnerability, which is not yet fixable regards flask cors. We need this module to make the frontend interact with the backend. The other services do not have vulnerabilities critical or higher so we did not include repeated pictures.


Analized by 

Images (3) Vulnerabilities (29) Packages (197)

☐ Fixable packages [Reset filters](#)

Package	Vulnerabilities
> flask-cors 5.0.0	0 1 0 0 0
> debian/glibc 2.36-9+deb12u9	0 0 0 7 0
> debian/systemd 252.31-1~deb12u1	0 0 0 4 0
> debian/krb5 1.20.1-2+deb12u2	0 0 0 3 0
> debian/perl 5.36.0-7+deb12u1	0 0 0 2 0
> debian/gcc-12 12.2.0-14	0 0 0 2 0
> debian/util-linux 2.38.1-5+deb12u2	0 0 0 1 0
> debian/tar 1.34+dfsg-1.2+deb12u1	0 0 0 1 0
> debian/sqlite3 3.40.1-2+deb12u1	0 0 0 1 0
> debian/shadow 1:4.13+dfsg1-1	0 0 0 1 0

1-10 of 16 < >

Analized by 

Images (3) Vulnerabilities (29) Packages (197)

☒ Fixable packages [Reset filters](#)

Package	Vulnerabilities
No results found.	

### Additional Features

- **Logs:** user actions are logged to the users DB and the admin can query logs with: **/admin/logs**
  - This feature is useful to see what users are doing, to log errors and success events. It is implemented by keeping a table “logs” in the users database. Each time an endpoint is queried a log is inserted with the result of the intermediate steps. This was useful during debugging and development.
- **Client:** the application provides an intuitive GUI to interact with the system and perform a part of the total possible requests to the backend **(not all the user stories are implemented on the client side, only the most effective)**
  - The navigation in the client side application doesn't follow a precise flow of action. The only thing needed is to login first (by the credential already filled in the login form) or by other credentials used for the registration of another player. The client side provides only an interface for the players (no logs section for the admin).

If in the process of login the client shows an alert like ‘axios error network’, you must tell the browser to not apply specific verification to the page caused by the HTTPS requests made. For this the solution is to contact the backend directly

(<https://localhost:3000/auth/login>), then click on any part of the screen and type “thisisinsecure”. Go back to the original page and everything should work.

Reference (secondo comment):

<https://stackoverflow.com/questions/55381447/getting-err-cert-authority-invalid-with-axios>