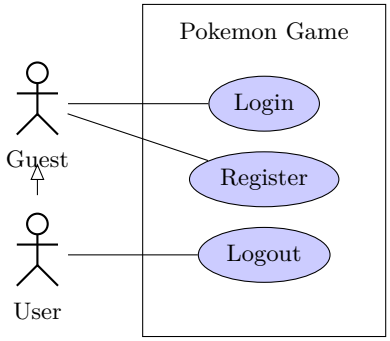
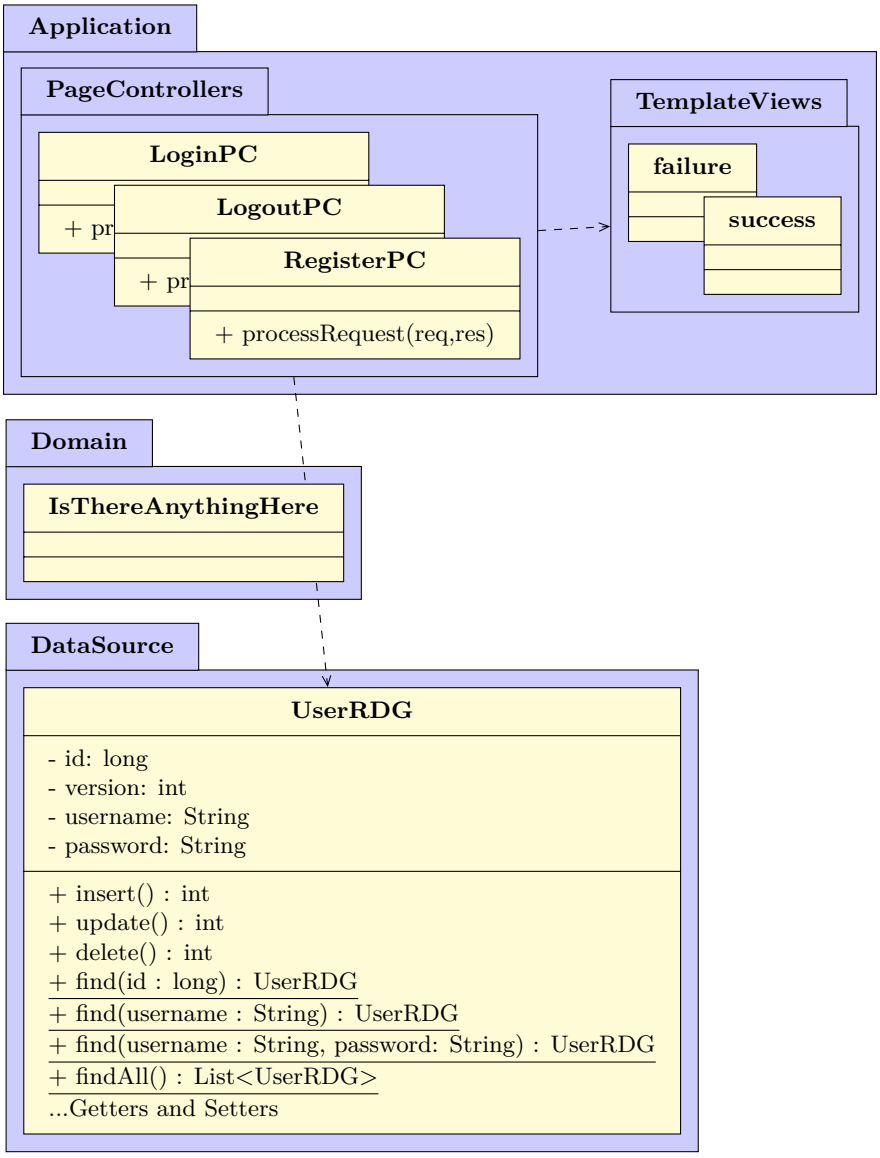
1 User Management

1.1 Use Cases



1.2 Class Diagrams



1.3 Sequence Diagram

This works in `https://www.websequencediagrams.com/`

```
title Register User

participant ":RegisterPC" as PC
participant "req:HttpServletRequest" as req
participant "session:HttpSession" as s
participant "UserRDG" as RDG
participant "u:UserRDG" as u
participant "success.jsp" as jsp

PC->req:getParameter("user"):"fred"
PC->req:getParameter("pass"):"bob"
PC->RDG:find("fred"):null
PC->*u: <<create>>
PC->u:insert()
PC->u:getId():2
PC->req:getSession();
PC->s:setAttribute("userid", 2)
PC->req:setAttribute("message", "...")
PC->jsp:forward(req,res)
```

1.4 Code

1.4.1 Register

```
public class RegisterPC {
    public void processRequest(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException {
        String user = req.getParameter("user");
        String pass = req.getParameter("pass");
        if(user==null || user.isEmpty() || pass==null || pass.isEmpty() ) {
            req.setAttribute("message", "Please enter both a username and a password.");
            req.forward("failure.jsp");
        } else {
            UserRDG u = UserRDG.find(user);
            if(u != null) {
                req.setAttribute("message", "That user has already registered.");
                req.forward("failure.jsp");
            } else {
                u = new UserRDG(getNewUserID(), user, pass);
                u.insert();
                long id = u.getId();
                req.getSession(true).setAttribute("userid", id);
                req.setAttribute("message", "User '" + user + "' has been successfully registered.");
                req.forward("success.jsp");
            }
        }
    }
}
```

1.4.2 Login

```
public class LoginPC {
    public void processRequest(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException {
        String user = req.getParameter("user");
        String pass = req.getParameter("pass");

        UserRDG u = UserRDG.find(user, pass);
        if(u == null) {
            req.setAttribute("message", "I do not recognize that username and password combination.");
            req.forward("failure.jsp");
        } else {
            long id = u.getId();
            req.getSession(true).setAttribute("userid", id);
            req.setAttribute("message", "User '" + u.getUsername() + "' has been successfully logged in.");
            req.forward("success.jsp");
        }
    }

}
```

1.4.3 Logout

```
public class LogoutPC {
    public void processRequest(HttpServletRequest req, HttpServletResponse res)
      throws ServletException, IOException {
        long id = (Long)req.getSession(true).getAttribute("userid", id);
        UserRDG u = UserRDG.find(id);

        req.getSession(true).invalidate();
        req.setAttribute("message", "User '" + u.getUsername() + "' has been successfully logged out.");
        req.forward("success.jsp");

    }

}
```

But what if no user was logged in? What do we check for? What can go wrong?

1.5 Data

We really only have failure and success here:

1.5.1 failure

```
{
        "status":"fail",
         "message":"Something went horribly wrong, but make your message more helpful!"
}
```

1.5.2 success

```
{
        "status":"success",
         "message":"Things went okay! We should probably say specifically what."
}
```

1.6 Calls

1.6.1 Register

path
/Register

method
POST

params
user
pass

1.6.2 Login

path
/Login

method
POST

params
user
pass

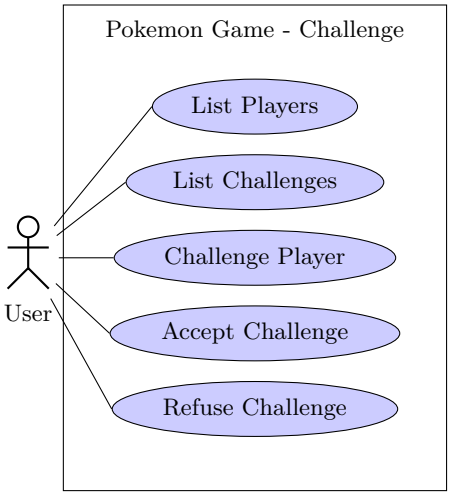
1.6.3 Logout

path
/Register

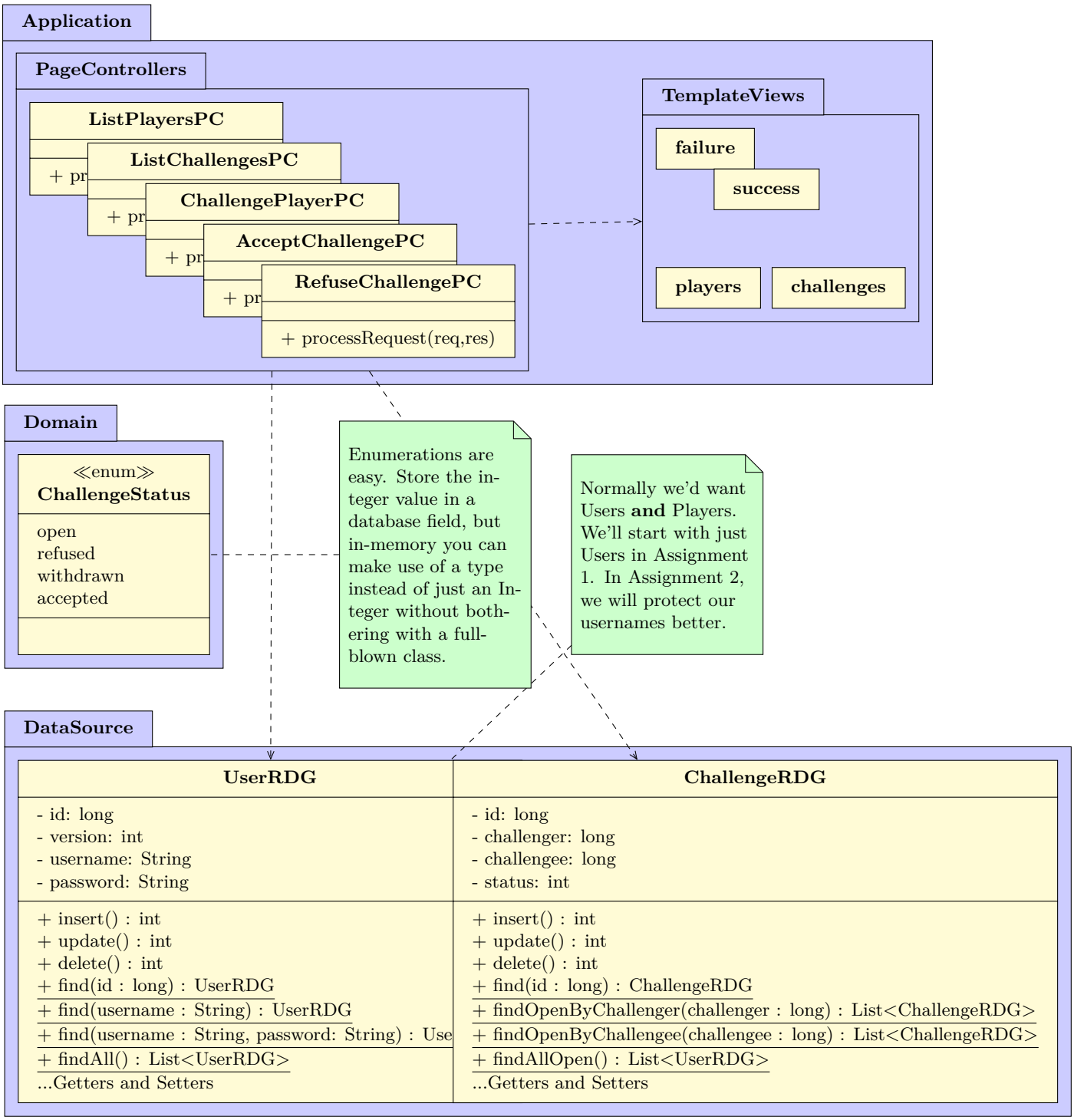
method
POST

2 Challenge Management

2.1 Use Cases



2.2 Class Diagrams



2.3 Sequence Diagram

This works in `https://www.websequencediagrams.com/`

```
title Tentative Solution: Accept Challenge

participant :AcceptChallengePC as PC
participant ChallengeRDG as RDG
participant req:HttpServletRequest as req
participant c:ChallengeRDG as c
participant g:GameRDG as g

PC->req: getParam("challenge"):challenge
PC->RDG: find(challenge):c
PC->c: getChallengee()
PC->req: getSession().getAttribute("userid")
PC->c: setStatus(ChallengeStatus.accepted.ordinal())
PC->c: update()

PC->dRDG: findByPlayer(c.getChellenger): d1
PC->dRDG: findByPlayer(challengee): d2

PC->*g: <<create>>(c.getChallenger, challengee, d1.getId(), d2.getID())
PC->g: insert()
```

2.4 Data

Here we consider the list of Players, in this case Users, and the list of challenges.

2.4.1 players

```
{
        "players": [
                        {"id": 1, "user": "alice"},
                        {"id": 2, "user": "bob"},
                        {"id": 3, "user": "chuck"},
                        {"id": 4, "user": "darcy"}
        ]
}
```

2.4.2 challenges

```
{
        "challenges": [
                        {"id": 1, "challenger": 1, "challengee": 2, "status": 3},
                        {"id": 2, "challenger": 2, "challengee": 1, "status": 2},
                        {"id": 3, "challenger": 3, "challengee": 1, "status": 0},
                        {"id": 4, "challenger": 4, "challengee": 3, "status": 1}
        ]
}
```

2.5 Calls

2.5.1 List Players

path
/ListPlayers

method
GET

2.5.2 List Challenges

path
/ListChallenges

method
GET

2.5.3 Challenge Player

path
/ChallengePlayer

method
POST

params
challengee

2.5.4 Accept Challenge

path
/AcceptChallenge

method
POST

params
challenge

2.5.5 Refuse Challenge

path
/RefuseChallenge

method
POST

params
challenge