Distributed Programming

Lecture 05 - REST Web Services and HTTP Communication in C++ on Unreal Engine

Edirlei Soares de Lima
<edirlei.lima@universidadeeuropeia.pt>

What is a REST Web Service?

- A Web Service is a software system identified by a URI/URL, whose public interfaces and bindings are defined and described using XML (or JSON).
  - It is designed to support interoperable machine-to-machine interaction over a network (HTTP or HTTPS).

- REST (Representational State Transfer) is an architectural style for creating web services.
  - The underlying protocol for REST is HTTP, which is the basic web protocol.
  - REST web services are lightweight, maintainable, and scalable.

REST Web Services

- REST Web Services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations.
  - A resource is identified by a URI/URL;
  - A request returns a document with a representation of the requested resource;
  - Each request contains all the information needed to be processed (stateless operations).
  - All resources are accessed by a well-known set of HTTP operations: POST, GET, PUT, DELETE.
  - Information transmitted in the requests and answers is encoded as XML or JSON.

XML vs. JSON

- XML:

```
<players>
  <player>
    <name>John</name>
    <score>100</score>
  </player>
  <player>
    <name>Anna</name>
    <score>200</score>
  </player>
</players>
```

- JSON:

```
{"players":[
    {"name":"John", "score":"100"},
    {"name":"Anna", "score":"200"}
]}
```

REST Web Services – Key Elements

- **Resources:**
  - Considering a REST web service that has records of several players' scores. We can access the score of a player (resource) via a URL: http://example.com/player/1, where 1 would be the ID of a player.

- **Request Verbs:**
  - Describe what the client want to do with the resource. For example, a GET verb is used to get data. Other verbs: POST, PUT, and DELETE.

- **Request Headers:**
  - Additional instructions sent with a request. These might define the type of response required or the authorization details.

REST Web Services – Key Elements

- **Request Body:**
  - Is the data that is sent with the request to the REST web service.
- **Response Body:**
  - Is the data that comes from the REST web service as response of request.
- **Response Status:**
  - Are general codes that are returned along with the response from the web server. For example, 404 indicates that the resource was not found.

REST Architectural Characteristics

- **Distributed resources:**
  - Every resource should be accessible via the normal HTTP commands of GET, POST, PUT, or DELETE.
- **Client/Server:**
  - The client send requests to the web service (server). The server either reject the requests or comply and provide an adequate response to the client.
- **Stateless:**
  - The server should not maintain any sort of information between requests from clients.

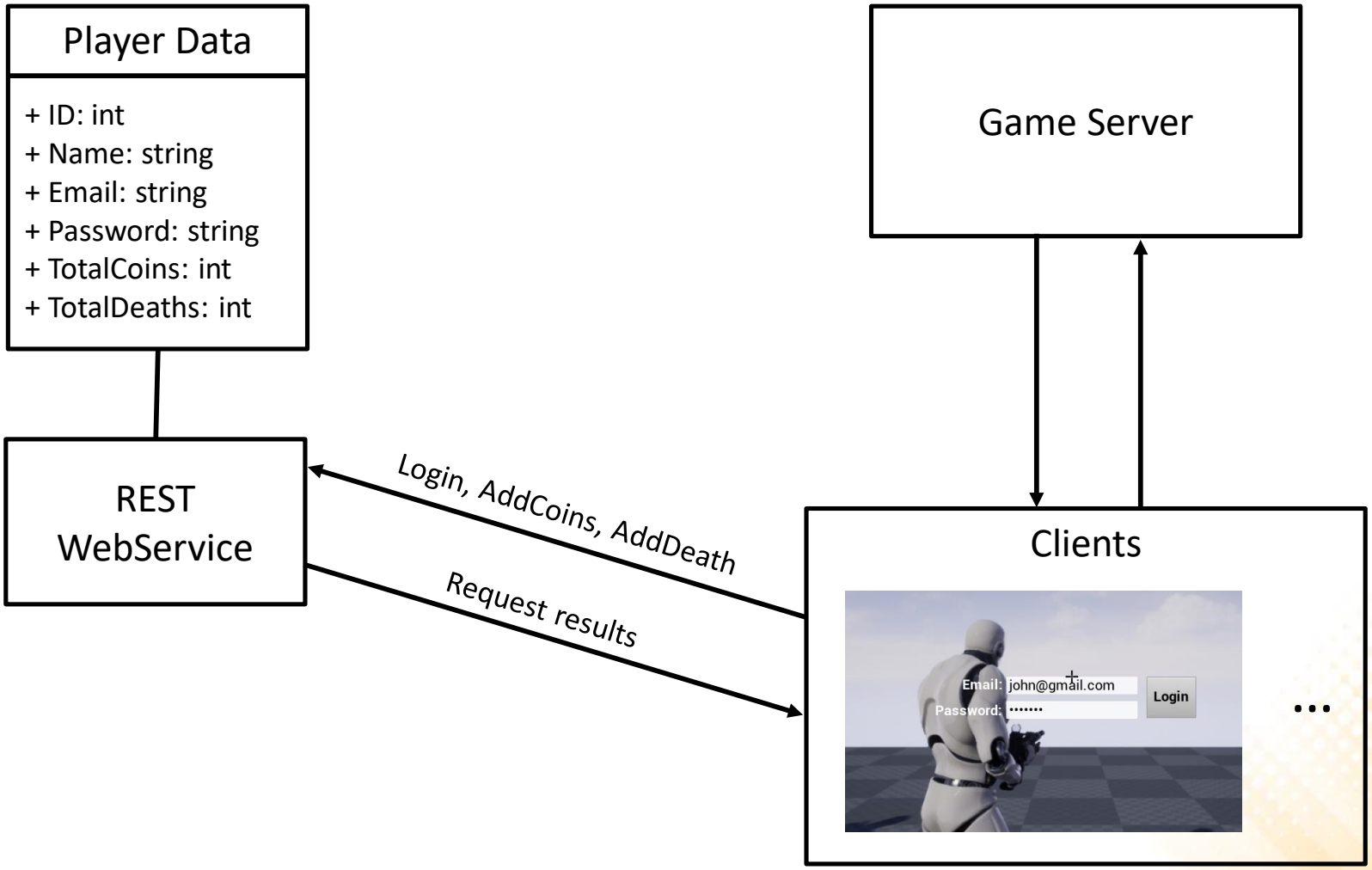
REST Architectural Characteristics

- **Layered:**
  - Any additional layer can be inserted between the client and the actual server hosting the REST web service.

- **Supports Caching:**
  - Since each server client request is independent in nature, sometimes the client might ask the server for the same request again.

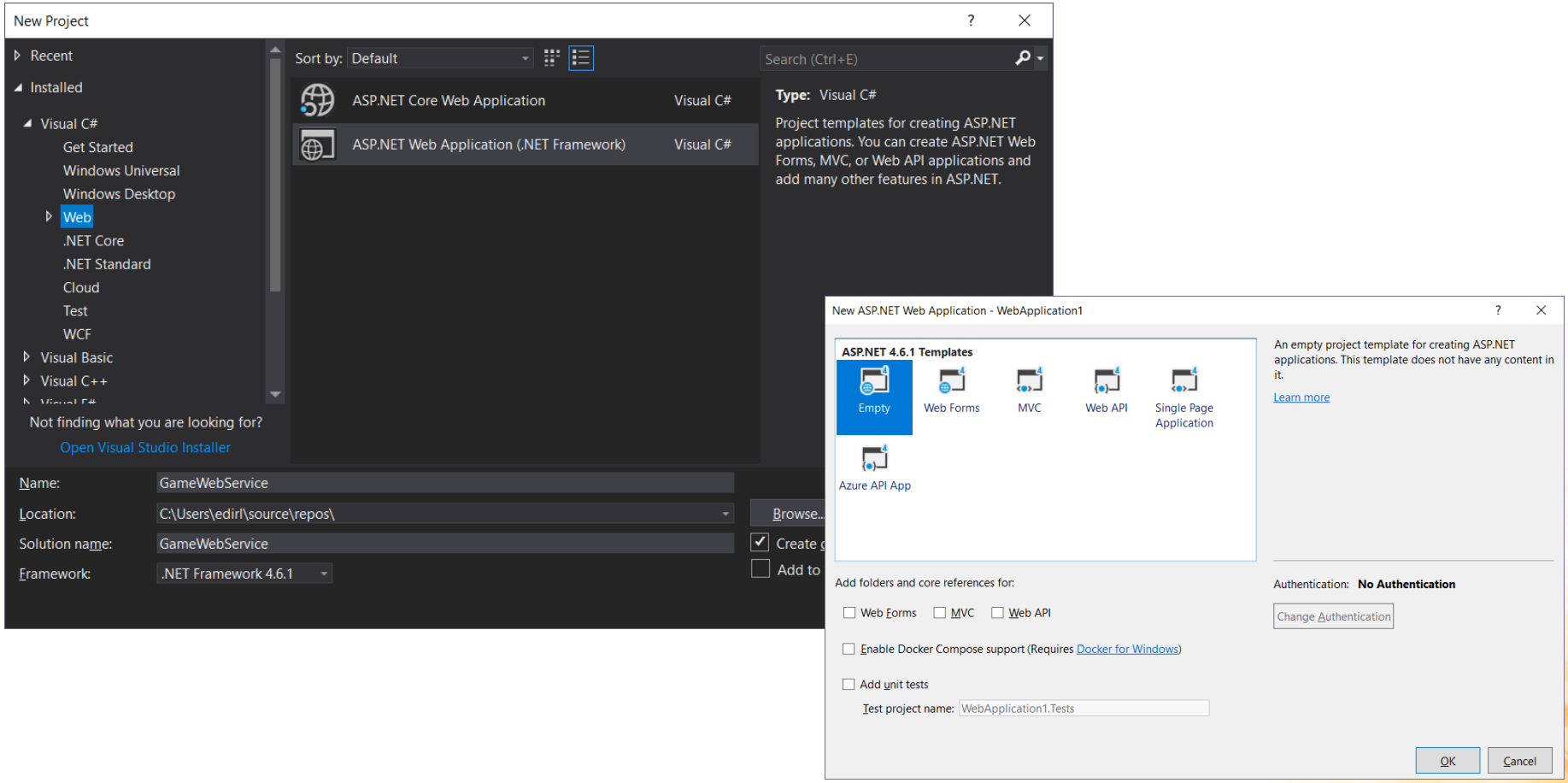
  - The cache is implemented on the client. If the same request is necessary, instead of going to the server, the client go to the cache and get the required information.

Game Project



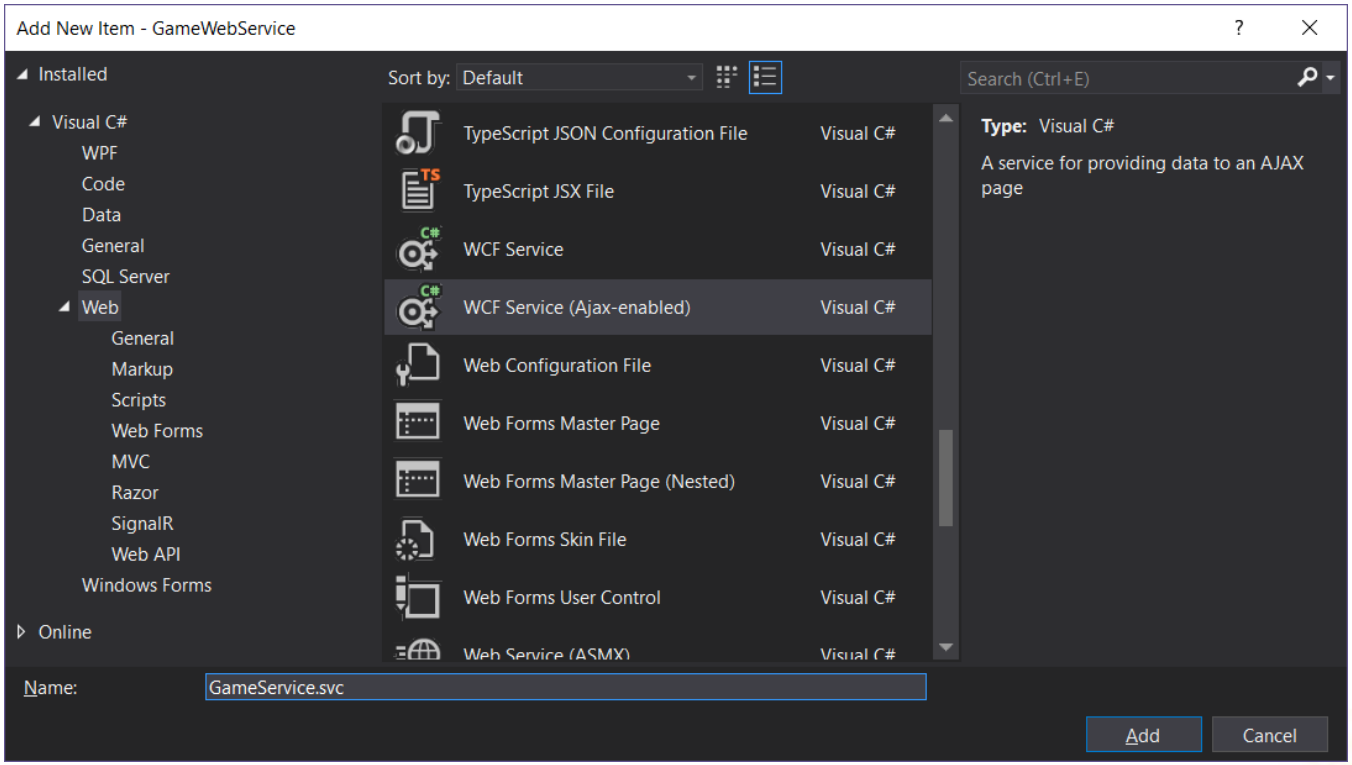
REST WebService

- Create a new **ASP.NET Web Application** (Empty):



REST WebService

- Add a new **WCF Service**:



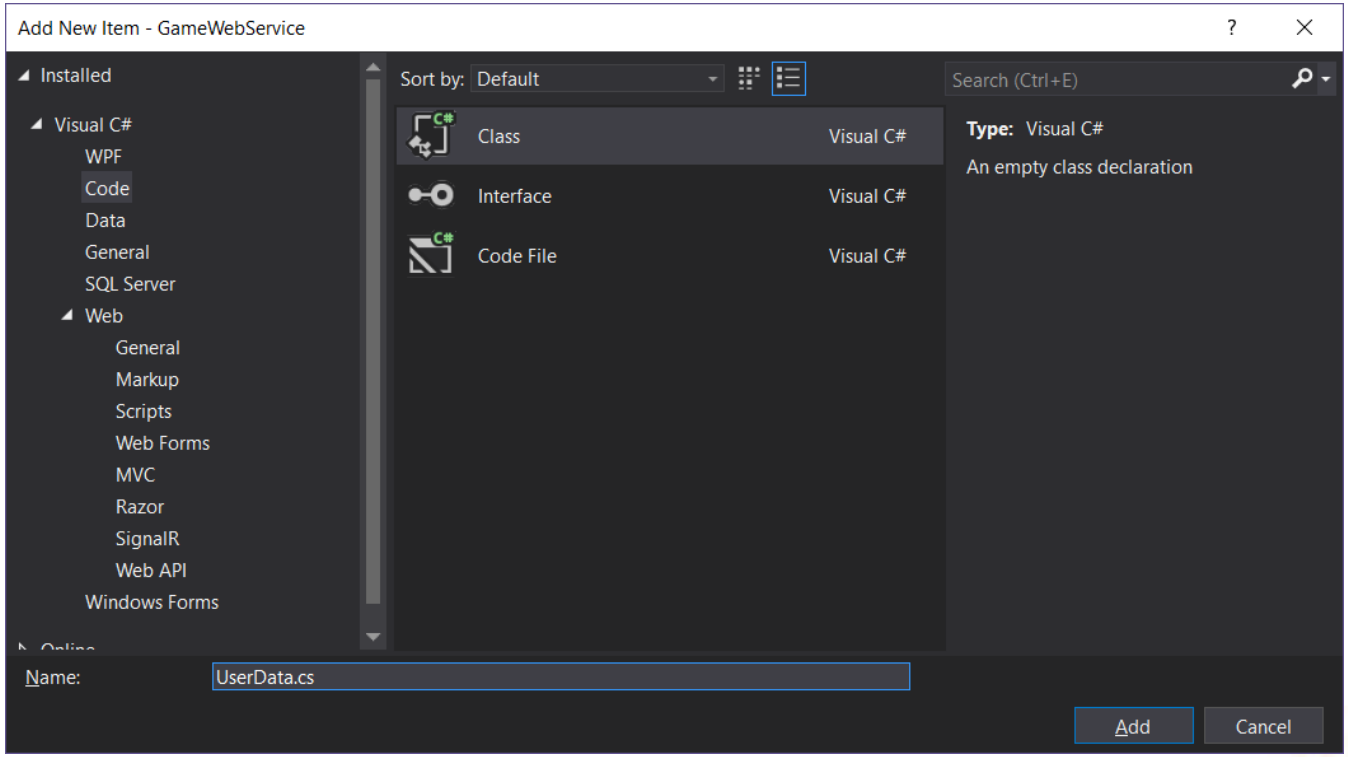
REST WebService

- In **Web.config**, change the tag **<enableWebScript/>** to **<webHttp/>** in order to make the WCF service RESTful:

```
<configuration>

  ...

  <system.serviceModel>
    <behaviors>
      <endpointBehaviors>
        <behavior name="GameWebService.GameServiceAspNetAjaxBehavior">
          <webHttp/>
        </behavior>
      </endpointBehaviors>
    </behaviors>

    ...
```

REST WebService

- Add a new **C# class** to the project:



REST WebService

- Data classes:

```
public class UserData{
  private static int ID_CONT = 0;
  public int id;
  public string name;
  public string email;
  public string password;
  public int totalcoins;
  public int totaldeaths;
  public string hash;

  public UserData(string nname, string nemail, string npass){
    id = ID_CONT++;
    name = nname;
    email = nemail;
    password = npass;
    totalcoins = 0;
    totaldeaths = 0;
    hash = "";
  }
}
```

REST WebService

- Data classes:

```
public enum ResponseStatus { Successed, Failed };

[DataContract]
public class LoginResponse{
  [DataMember] public ResponseStatus status;
  [DataMember] public int userid;
  [DataMember] public string name;
  [DataMember] public string hash;
  [DataMember] public int totalcoins;
  [DataMember] public int totaldeaths;

  public LoginResponse(ResponseStatus nstatus, int nuserid, string
           nname, string nhash, int ntotalcoins, int ntotaldeaths){
    status = nstatus;
    userid = nuserid;
    name = nname;
    hash = nhash;
    totalcoins = ntotalcoins;
    totaldeaths = ntotaldeaths;
  }
}
```

REST WebService

- Data classes:

```
[DataContract]
public class UserInfoResponse{
  [DataMember] public ResponseStatus status;
  [DataMember] public string name;
  [DataMember] public int totalcoins;
  [DataMember] public int totaldeaths;
  public UserInfoResponse(ResponseStatus nstatus, string nname, int
                          ntotalcoins, int ntotaldeaths){
    status = nstatus;
    name = nname;
    totalcoins = ntotalcoins;
    totaldeaths = ntotaldeaths;
  }
}
[DataContract]
public class PostResponse{
  [DataMember] public ResponseStatus status;
  public PostResponse(ResponseStatus nstatus){
    status = nstatus;
  }
}
```

REST WebService

- WebService implementation:

```
private static List<UserData> Users = new List<UserData>(
  new UserData[] {
    new UserData("Peter", "peter@gmail.com", "12345"),
    new UserData("John", "john@gmail.com", "abcd123"),
    new UserData("Mary", "mary@gmail.com", "54321"),
    new UserData("Bob", "bob@gmail.com", "bobbob")
  });

static string GenerateSHA256Hash(string rawdata){
  using (SHA256 sha256Hash = SHA256.Create()){
    byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(
                                          rawdata));
    StringBuilder builder = new StringBuilder();
    for (int i = 0; i < bytes.Length; i++){
      builder.Append(bytes[i].ToString("x2"));
    }
    return builder.ToString();
  }
}
```

REST WebService

- WebService implementation (/login):

```
[WebInvoke(Method = "POST", RequestFormat = WebMessageFormat.Json,
 UriTemplate = "/login", ResponseFormat = WebMessageFormat.Json,
 BodyStyle = WebMessageBodyStyle.Wrapped)]
[OperationContract]
public LoginResponse Login(string email, string password){
  foreach (UserData udata in Users){
    if ((udata.email == email) && (udata.password == password)){
      udata.hash = GenerateSHA256Hash(DateTime.Now.ToString());
      return new LoginResponse(ResponseStatus.Successed, udata.id,
                          udata.name, udata.hash, udata.totalcoins,
                          udata.totaldeaths);
    }
   }
   return new LoginResponse(ResponseStatus.Failed, -1, "", "",
                              -1, -1);
}
```

REST WebService

- WebService implementation (/user/{id}):

```
[WebGet(UriTemplate = "/user/{id}", ResponseFormat =
 WebMessageFormat.Json)]
[OperationContract]
public UserInfoResponse GetUserInfo(string id){
  int uid = int.Parse(id);
  foreach (UserData udata in Users){
    if (udata.id == uid){
      return new UserInfoResponse(ResponseStatus.Successed,
                  udata.name, udata.totalcoins, udata.totaldeaths);
    }
  }
  return new UserInfoResponse(ResponseStatus.Failed, "", -1, -1);
}
```

REST WebService

- WebService implementation (/user/coin):

```
[WebInvoke(Method = "POST", RequestFormat = WebMessageFormat.Json,
 UriTemplate = "/user/coin", ResponseFormat = WebMessageFormat.Json,
 BodyStyle = WebMessageBodyStyle.Wrapped)]
[OperationContract]
public PostResponse AddUserCoin(string userid, string hash){
  int uid = int.Parse(userid);
  foreach (UserData udata in Users){
    if ((udata.id == uid)&&(udata.hash.Equals(hash))){
      udata.totalcoins++;
      return new PostResponse(ResponseStatus.Successed);
    }
  }
  return new PostResponse(ResponseStatus.Failed);
}
```

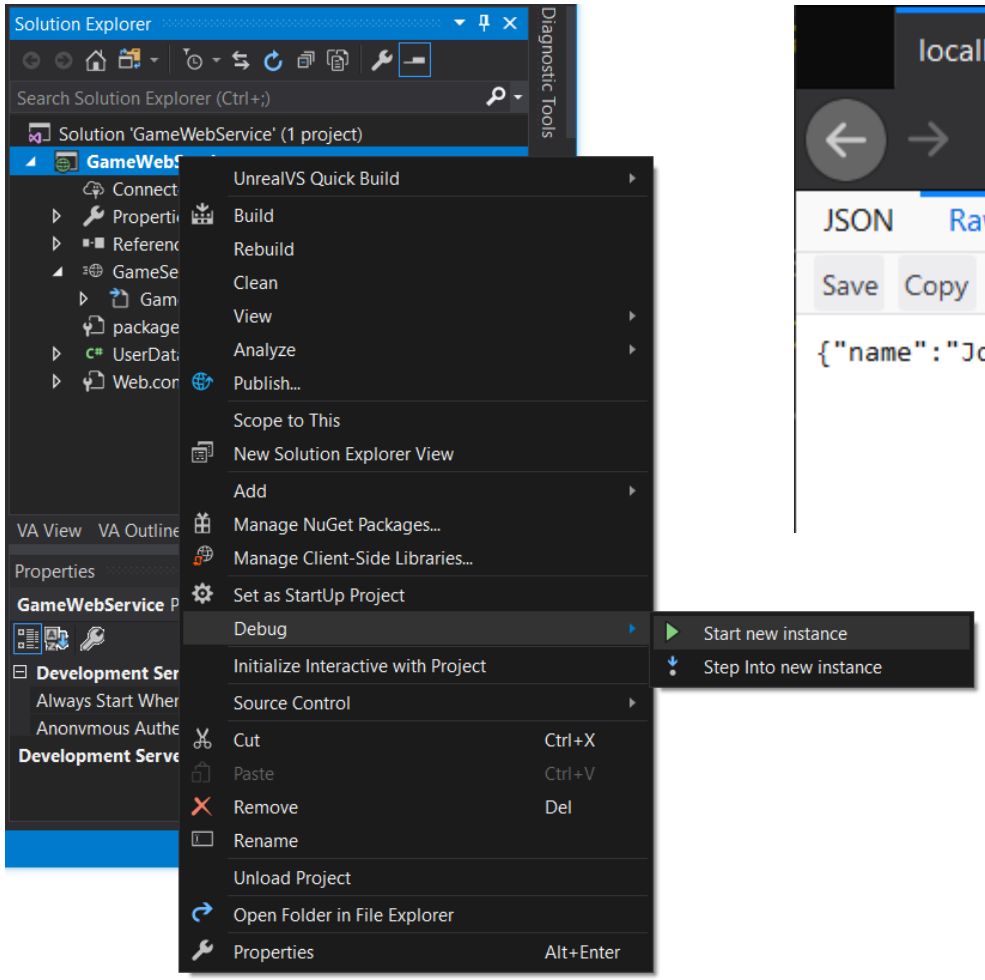
REST WebService

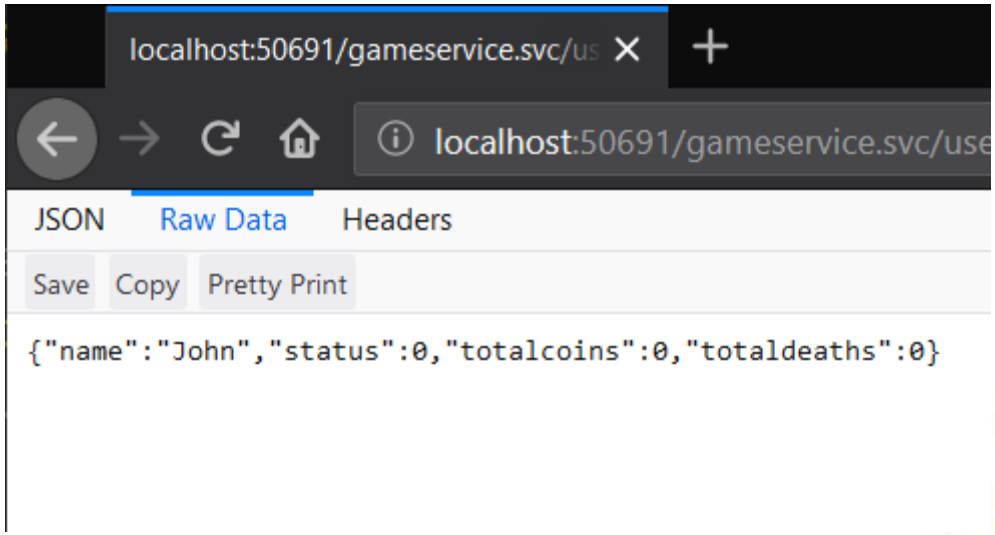
- WebService implementation (/user/death):

```
[WebInvoke(Method = "POST", RequestFormat = WebMessageFormat.Json,
 UriTemplate = "/user/death", ResponseFormat = WebMessageFormat.Json,
 BodyStyle = WebMessageBodyStyle.Wrapped)]
[OperationContract]
public PostResponse AddUserDeath(string userid, string hash){
  int uid = int.Parse(userid);
  foreach (UserData udata in Users){
    if ((udata.id == uid) && (udata.hash.Equals(hash))){
      udata.totaldeaths++;
      return new PostResponse(ResponseStatus.Successed);
    }
  }
  return new PostResponse(ResponseStatus.Failed);
}
```

REST WebService

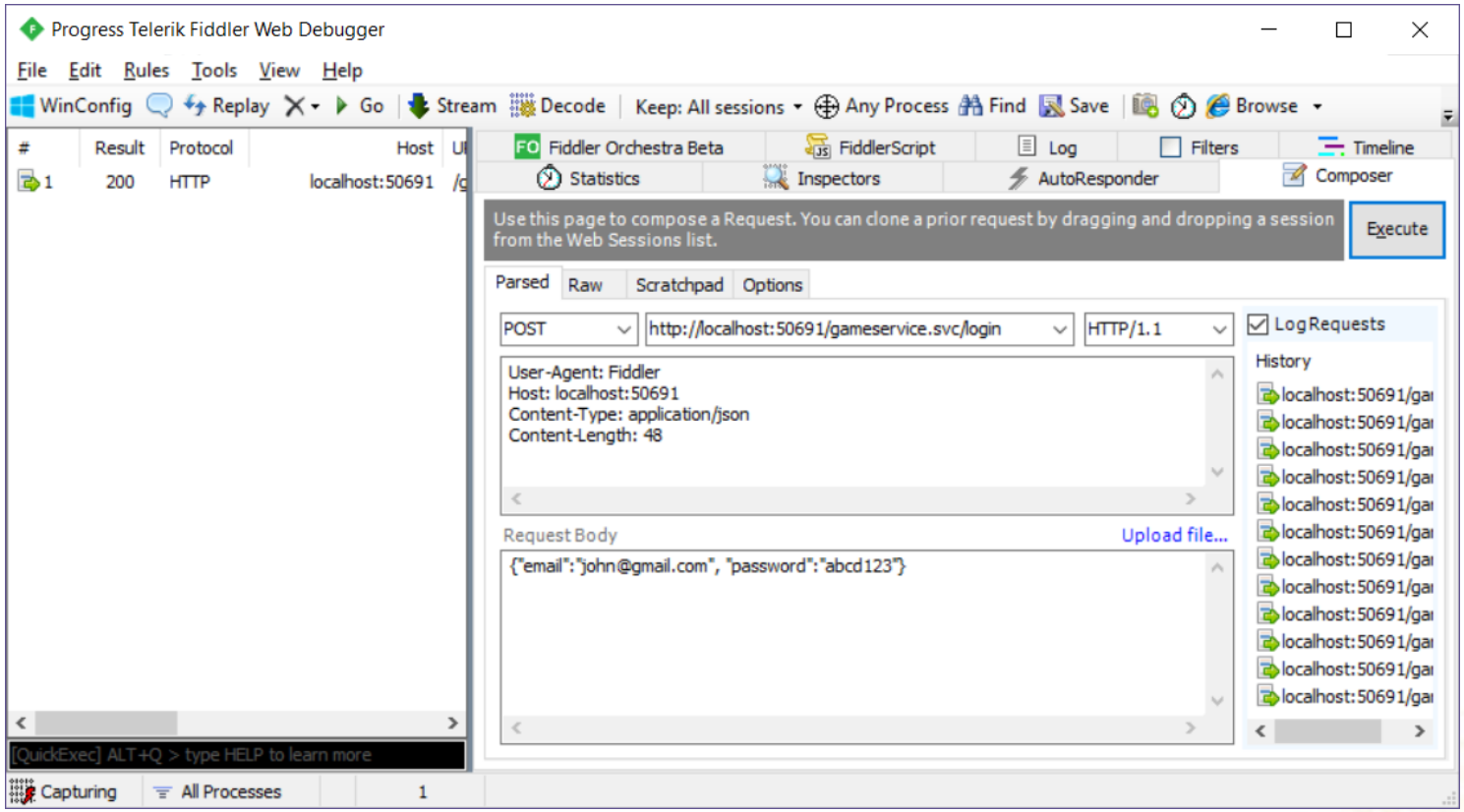
- Test the WebService:





REST WebService

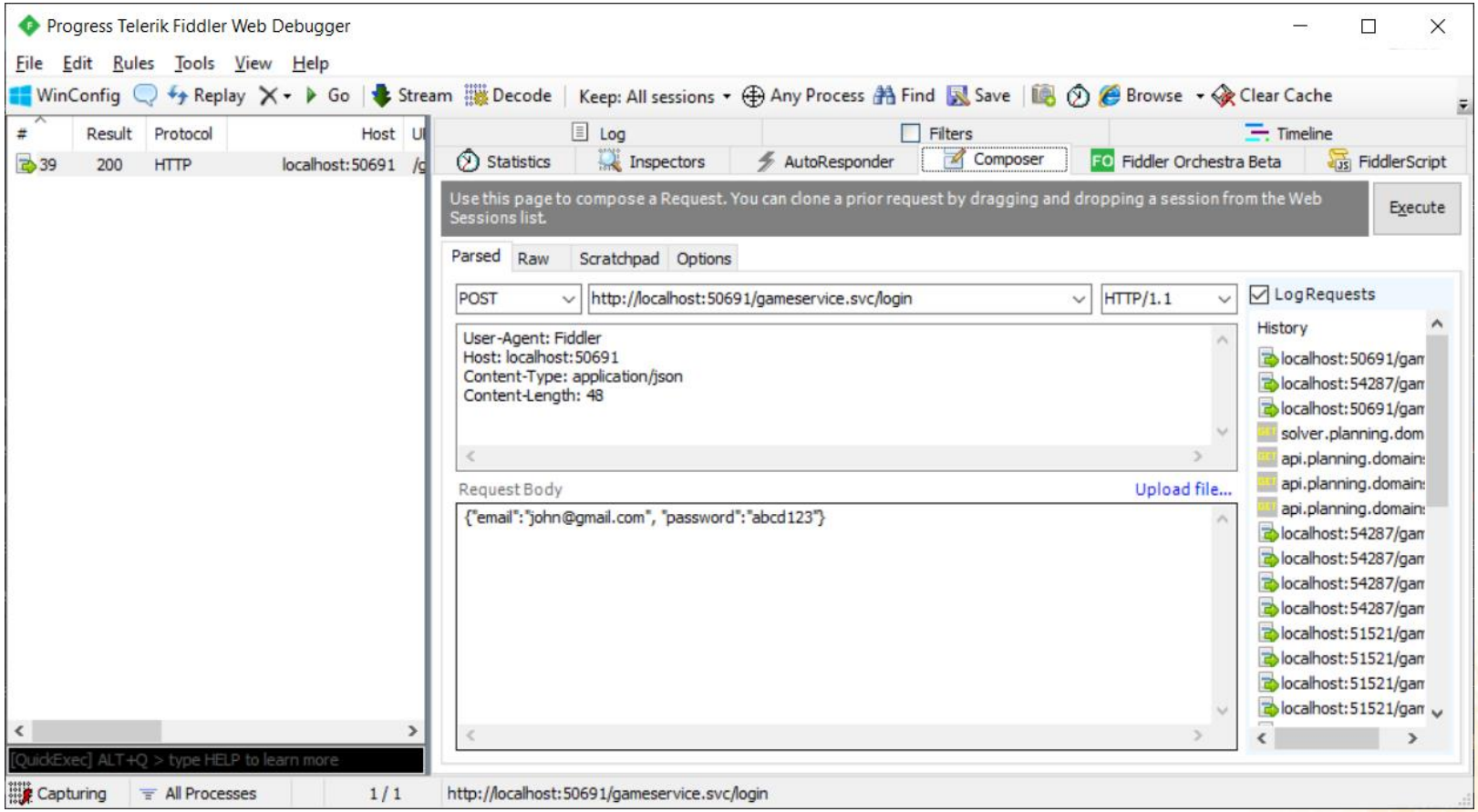
- Test the WebService (**/login**):



Fiddler: https://www.telerik.com/fiddler

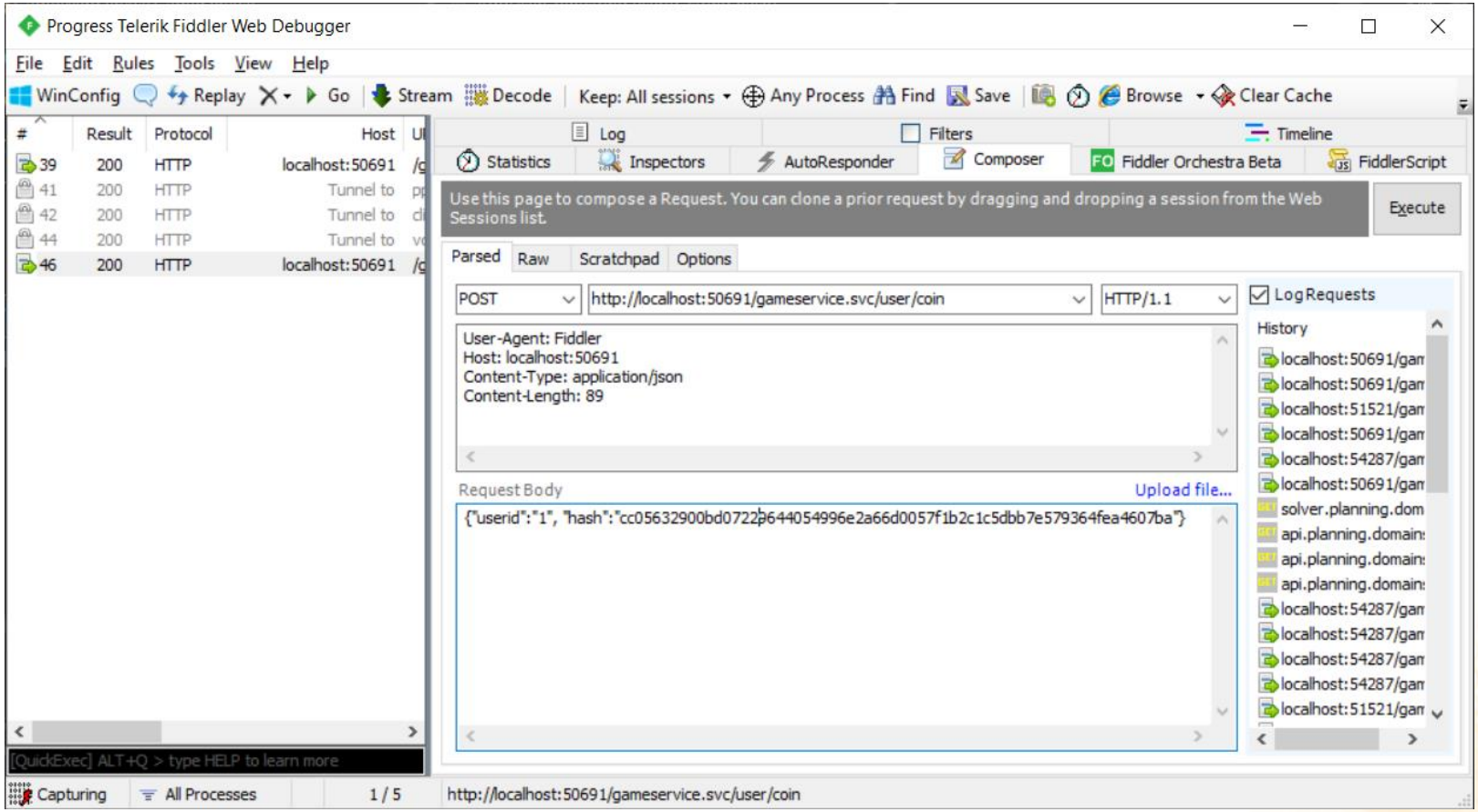
REST WebService

- Test the WebService (**/login**):



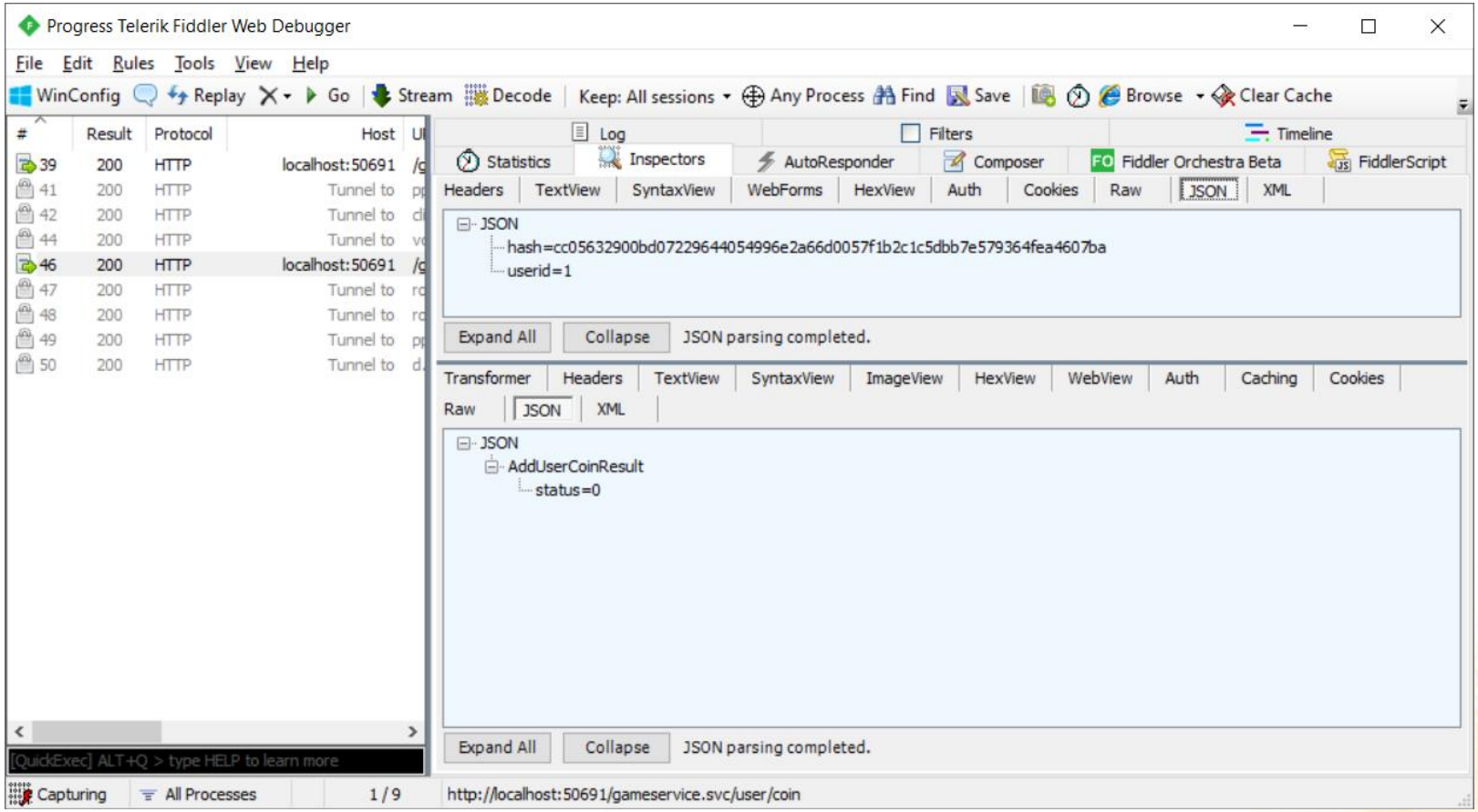
REST WebService

- Test the WebService (**/user/coin**):



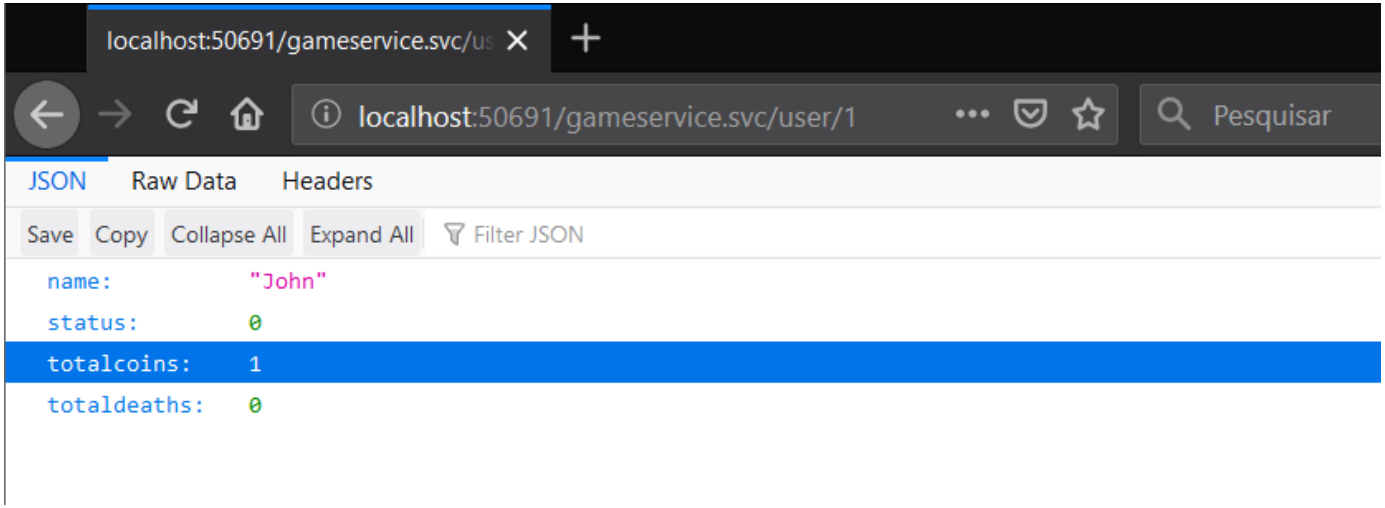
REST WebService

- Test the WebService (**/user/coin**):



REST WebService

- Test the WebService (**/user/coin**):



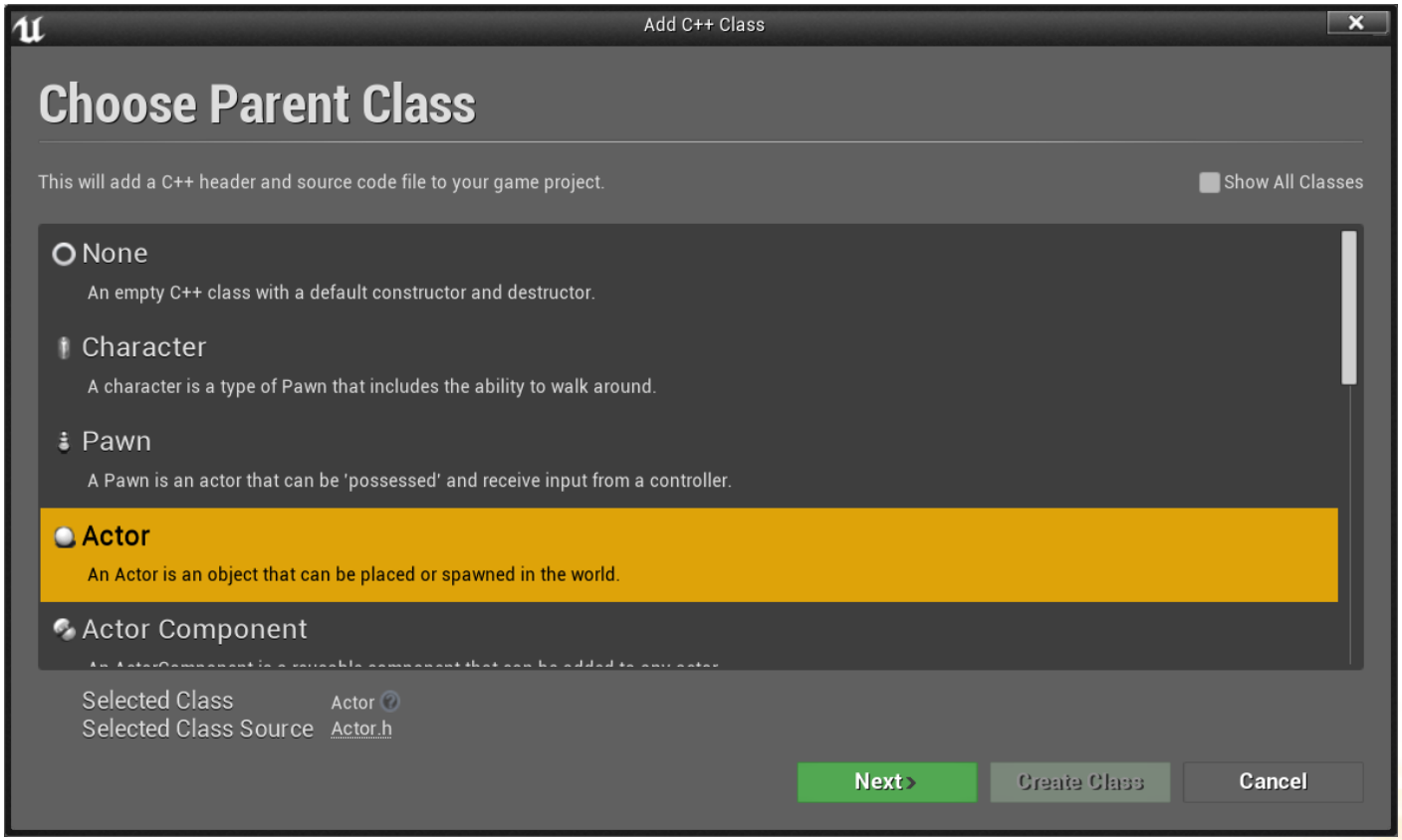
HTTP Communication in Unreal Engine

- Project dependencies to be added to the project build file (YourGameName.Build.cs):
  - Http
  - Json
  - JsonUtilities

```
...

PrivateDependencyModuleNames.AddRange(new string[] {
                            "Http", "Json", "JsonUtilities" });

...
```

HTTP Communication in Unreal Engine

- Create a new C++ class (Actor): HTTPService



HTTP Communication in Unreal Engine

- HTTPService implementation: data structures

HTTPService.h

```
USTRUCT()
struct FLoginRequest {
  GENERATED_BODY()

  UPROPERTY()
  FString email;
  UPROPERTY()
  FString password;
};

USTRUCT()
struct FLoginResponse {
  GENERATED_BODY()

  UPROPERTY()
  FLoginData LoginResult;
};
```

HTTPService.h

```
USTRUCT()
struct FLoginData {
  GENERATED_BODY()

  UPROPERTY()
  int status;
  UPROPERTY()
  int userid;
  UPROPERTY()
  FString name;
  UPROPERTY()
  FString hash;
  UPROPERTY()
  int totalcoins;
  UPROPERTY()
  int totaldeaths;
};
```

HTTP Communication in Unreal Engine

- HTTPService implementation: data structures

HTTPService.h

```
USTRUCT()
struct FAddCoinRequest {
  GENERATED_BODY()

  UPROPERTY()
  FString userid;
  UPROPERTY()
  FString hash;
};

USTRUCT()
struct FResponse_AddCoin {
  GENERATED_BODY()

  UPROPERTY()
  FAddCoinData AddUserCoinResult;
};
```

HTTPService.h

```
USTRUCT()
struct FAddCoinData {
  GENERATED_BODY()

  UPROPERTY()
  int status;
};
```

HTTP Communication in Unreal Engine

- HTTPService implementation: basic http communication

HTTPService.h

```
protected:

  FHttpModule* Http;

  FString APIBaseUrl;

  TSharedRef<IHttpRequest> RequestWithRoute(FString Subroute);

  TSharedRef<IHttpRequest> GetRequest(FString Subroute);

  TSharedRef<IHttpRequest> PostRequest(FString Subroute,
                                       FString ContentJsonString);

  void Send(TSharedRef<IHttpRequest>& Request);

  bool ResponseIsValid(FHttpResponsePtr Response,
                       bool bWasSuccessful);
```

HTTP Communication in Unreal Engine

- HTTPService implementation: basic http communication

HTTPService.cpp

```
AHTTPService::AHTTPService()
{
  APIBaseUrl = TEXT("http://localhost:50691/gameservice.svc");
}


TSharedRef<IHttpRequest> AHTTPService::RequestWithRoute(Fstring
                                                        Subroute)
{
  TSharedRef<IHttpRequest> Request = Http->CreateRequest();
  Request->SetURL(APIBaseUrl + Subroute);
  Request->SetHeader(TEXT("User-Agent"), TEXT("X-UnrealEngine-Agent"));
  Request->SetHeader(TEXT("Content-Type"), TEXT("application/json"));
  Request->SetHeader(TEXT("Accepts"), TEXT("application/json"));
  return Request;
}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: basic http communication

HTTPService.cpp

```
TSharedRef<IHttpRequest> AHTTPService::GetRequest(FString Subroute){
  TSharedRef<IHttpRequest> Request = RequestWithRoute(Subroute);
  Request->SetVerb("GET");
  return Request;
}

TSharedRef<IHttpRequest> AHTTPService::PostRequest(FString Subroute,
                                                   FString ContentJsonString){
  TSharedRef<IHttpRequest> Request = RequestWithRoute(Subroute);
  Request->SetVerb("POST");
  Request->SetContentAsString(ContentJsonString);
  return Request;
}

void AHTTPService::Send(TSharedRef<IHttpRequest>& Request){
  Request->ProcessRequest();
}
```

HTTP Communication in Unreal Engine

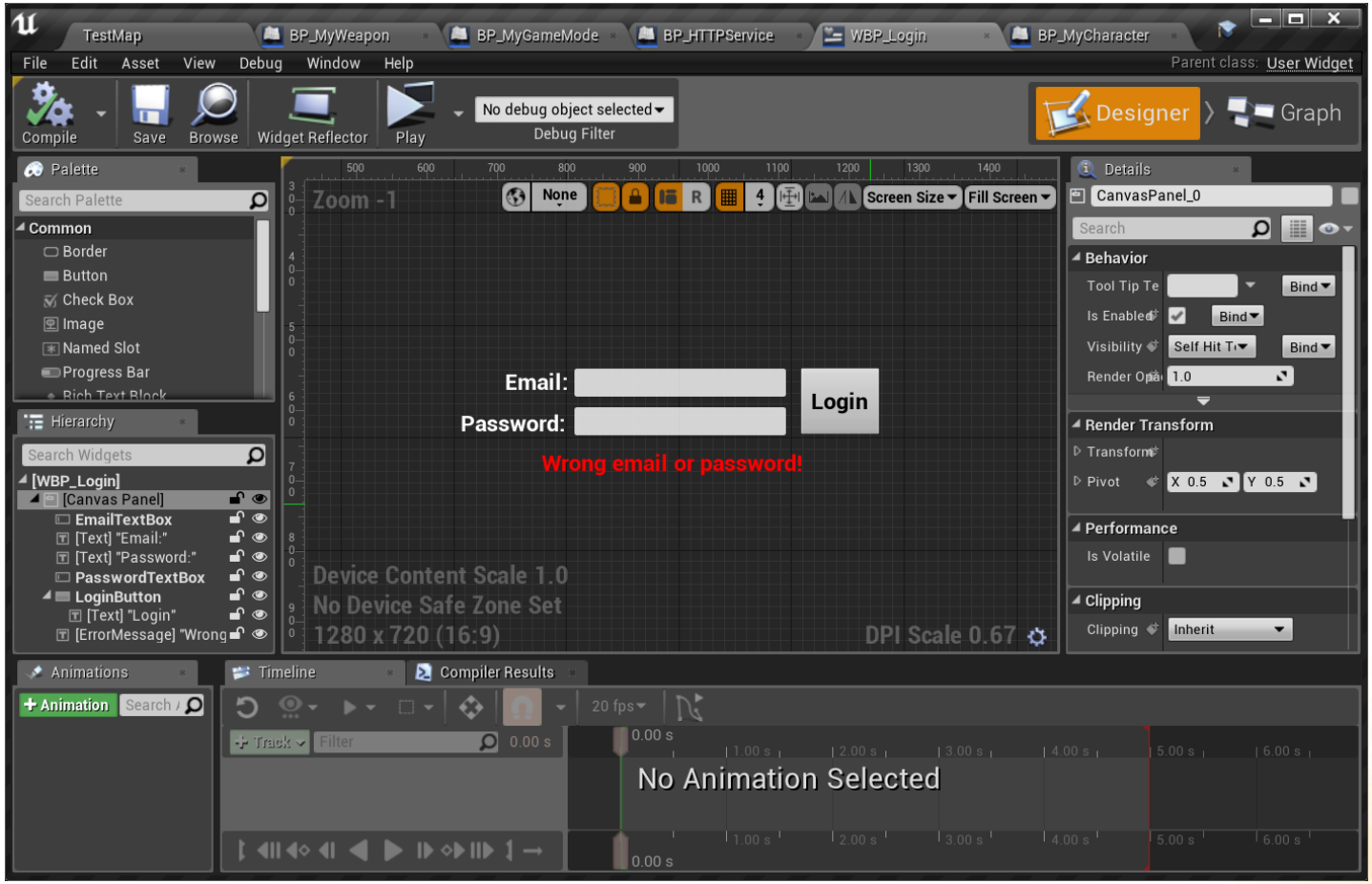
- HTTPService implementation: basic http communication

HTTPService.cpp

```
bool AHTTPService::ResponseIsValid(FHttpResponsePtr Response,
                                   bool bWasSuccessful){
  if ((!bWasSuccessful) || (!Response.IsValid())){
    return false;
  }
  if (EHttpResponseCodes::IsOk(Response->GetResponseCode())){
    return true;
  }
  else{
    UE_LOG(LogTemp, Warning, TEXT("Http Response returned error code:
                                   %d"), Response->GetResponseCode());
    return false;
  }
}
void AHTTPService::BeginPlay(){
  Super::BeginPlay();
  Http = &FHttpModule::Get();
}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login
  - Create a new widget blueprint for the login interface:



HTTP Communication in Unreal Engine

- HTTPService implementation: login

HTTPService.h

```
protected:
  class AMyCharacter* LocalCharacter;

public:

  UPROPERTY(EditDefaultsOnly, Category = "UI")
  TSubclassOf<class UUserWidget> LoginWidgetClass;

  UUserWidget* LoginWidget;

  void Login(FLoginRequest LoginCredentials, class APlayerState*
            playerState);

  void LoginResponse(FHttpRequestPtr Request, FHttpResponsePtr
        Response, bool bWasSuccessful, APlayerState* playerState);

  UFUNCTION()
  void OnLoginClicked();
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login

HTTPService.cpp

```
void AHTTPService::BeginPlay(){

  ...

  if (LoginWidgetClass){
    LoginWidget=CreateWidget<UUserWidget>(GetWorld(),LoginWidgetClass);
    LoginWidget->AddToViewport();

    UButton* LoginButton = Cast<UButton>(LoginWidget->
                                GetWidgetFromName(TEXT("LoginButton")));
    if (LoginButton){
      LoginButton->OnClicked.AddDynamic(this,
                                        &AHTTPService::OnLoginClicked);
    }

    ...
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login

HTTPService.cpp

```
    ...

    for (TActorIterator<AActor> ActorItr(GetWorld()); ActorItr;
++ActorItr){
      AMyCharacter* character = Cast<AMyCharacter>(*ActorItr);
      if (character){
        if (character->IsLocallyControlled()){
          LocalCharacter = character;
          LocalCharacter->DisableInput(nullptr);
          APlayerController* pController = Cast<APlayerController>(
                                        LocalCharacter->GetController());
          pController->bShowMouseCursor = true;
          pController->bEnableClickEvents = true;
          pController->bEnableMouseOverEvents = true;
        }
      }
    }
  }
}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login

HTTPService.cpp

```
void AHTTPService::OnLoginClicked(){
  UEditableTextBox* EmailText = Cast<UEditableTextBox>(LoginWidget->
                                GetWidgetFromName(TEXT("EmailTextBox")));
  UEditableTextBox* PasswordText = Cast<UEditableTextBox>(LoginWidget->
                                GetWidgetFromName(TEXT("PasswordTextBox")));
  UTextBlock* ErrorText = Cast<UTextBlock>(LoginWidget->
                                GetWidgetFromName(TEXT("ErrorMessage")));

  if ((EmailText) && (PasswordText) && (ErrorText)){
    FLoginRequest LoginCredentials;
    LoginCredentials.email = EmailText->GetText().ToString();
    LoginCredentials.password = PasswordText->GetText().ToString();
    ErrorText->SetVisibility(ESlateVisibility::Hidden);
    Login(LoginCredentials, LocalCharacter->PlayerState);
  }
}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login

HTTPService.cpp

```
void AHTTPService::Login(FLoginRequest LoginCredentials,
                         APlayerState* playerState)
{
  FString ContentJsonString;
  FJsonObjectConverter::UStructToJsonObjectString(FLoginRequest::
      StaticStruct(), &LoginCredentials, ContentJsonString, 0, 0);

  TSharedRef<IHttpRequest> Request = PostRequest("/login",
                                                 ContentJsonString);

  Request->OnProcessRequestComplete().BindUObject(this,
                       &AHTTPService::LoginResponse, playerState);
  Send(Request);
}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login

HTTPService.cpp

```
void AHTTPService::LoginResponse(FHttpRequestPtr Request,
                  FHttpResponsePtr Response, bool bWasSuccessful,
                  APlayerState* playerState){
  if (!ResponseIsValid(Response, bWasSuccessful)){
    return;
  }

  FLoginResponse LoginResponse;
  FString JsonString = Response->GetContentAsString();
  FJsonObjectConverter::JsonObjectStringToUStruct<FLoginResponse>(
                              JsonString, &LoginResponse, 0, 0);
  if (LoginResponse.LoginResult.status == 0){
    LoginWidget->RemoveFromViewport();
    LocalCharacter->EnableInput(nullptr);
    APlayerController* pController = Cast<APlayerController>(
                               LocalCharacter->GetController());
    ...
```

HTTP Communication in Unreal Engine

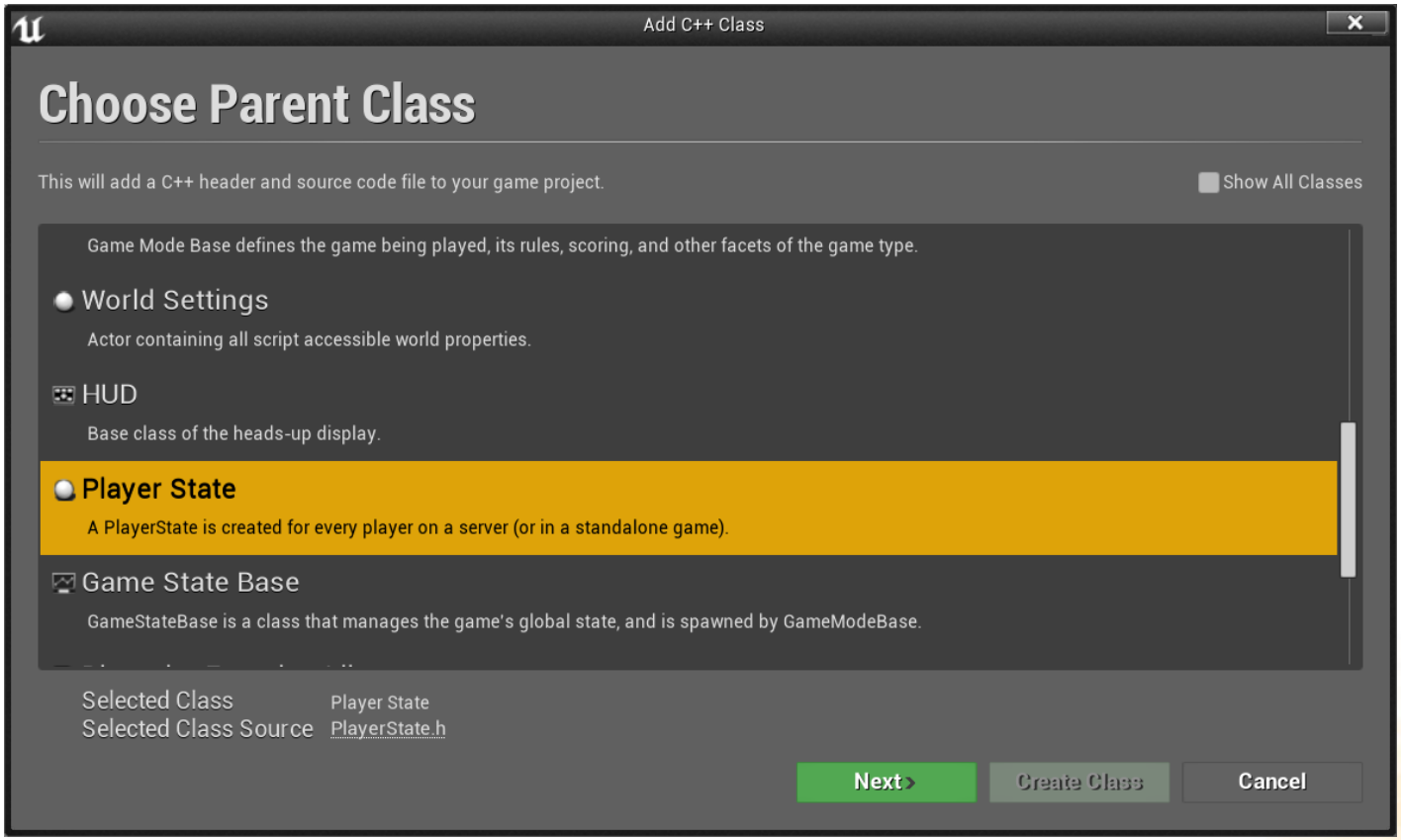
- HTTPService implementation: login

HTTPService.cpp

```
    ...
    pController->bShowMouseCursor = false;
    pController->bEnableClickEvents = false;
    pController->bEnableMouseOverEvents = false;

    AMyPlayerState* pState = Cast<AMyPlayerState>(playerState);
    pState->InitPlayerSession(this, LoginResponse.LoginResult.hash,
                              LoginResponse.LoginResult.userid,
                              LoginResponse.LoginResult.totalcoins,
                              LoginResponse.LoginResult.totaldeaths);
  }
  else{
    UTextBlock* ErrorText = Cast<UTextBlock>(LoginWidget->
                            GetWidgetFromName(TEXT("ErrorMessage")));
    if (ErrorText){
      ErrorText->SetVisibility(ESlateVisibility::Visible);
    }
  }
}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login
  - Create a new C++ class to define a new Player State:



HTTP Communication in Unreal Engine

- HTTPService implementation: login

MyPlayerState.h

```
private:
  class AHTTPService* HTTPServiceRef;

protected:
  UPROPERTY(Replicated)
  FString AuthenticationHash;

  UPROPERTY(Replicated)
  int PID;

  UPROPERTY(Replicated)
  int TotalCoins;

  UPROPERTY(Replicated)
  int TotalDeaths;

public:
  void InitPlayerSession(AHTTPService* HTTPService, FString hash,
                         int id, int coins, int deaths);
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login

MyPlayerState.cpp

```
void AMyPlayerState::InitPlayerSession(AHTTPService* HTTPService,
                    FString hash, int id, int coins, int deaths){
  HTTPServiceRef = HTTPService;
  AuthenticationHash = hash;
  Totalcoins = coins;
  TotalDeaths = deaths;
  PID = id;
}

void AMyPlayerState::GetLifetimeReplicatedProps(TArray<
                    FLifetimeProperty>& OutLifetimeProps) const {
  Super::GetLifetimeReplicatedProps(OutLifetimeProps);
  DOREPLIFETIME(AMyPlayerState, AuthenticationHash);
  DOREPLIFETIME(AMyPlayerState, TotalCoins);
  DOREPLIFETIME(AMyPlayerState, TotalDeaths);
  DOREPLIFETIME(AMyPlayerState, PID);
}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: login
  - Create a blueprint for the HTTPService and select the widget blueprint of the login interface.

  - In the Game Mode blueprint, select the new player state class:

HTTP Communication in Unreal Engine

- Test the login process in the game:

HTTP Communication in Unreal Engine

- HTTPService implementation: adding player coins (webservice)

HTTPService.h

```
...

public:

  void AddCoin(FAddCoinRequest AddCoinInfo);

  void AddCoinResponse(FHttpRequestPtr Request, FHttpResponsePtr
                       Response, bool bWasSuccessful);
```

HTTP Communication in Unreal Engine

- HTTPService implementation: adding player coins (webservice)

HTTPService.cpp

```
void AHTTPService::AddCoin(FAddCoinRequest AddCoinInfo)
{
  FString ContentJsonString;
  FJsonObjectConverter::UStructToJsonObjectString(FAddCoinRequest::
              StaticStruct(), &AddCoinInfo, ContentJsonString, 0, 0);

  TSharedRef<IHttpRequest> Request = PostRequest("/user/coin",
                                                 ContentJsonString);

  Request->OnProcessRequestComplete().BindUObject(this,
                                    &AHTTPService::AddCoinResponse);
  Send(Request);
}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: adding player coins (webservice)

HTTPService.cpp

```
void AHTTPService::AddCoinResponse(FHttpRequestPtr Request,
                        FHttpResponsePtr Response, bool bWasSuccessful){
  if (!ResponseIsValid(Response, bWasSuccessful)){
    return;
  }

  FResponse_AddCoin AddCoinResponse;
  FString JsonString = Response->GetContentAsString();
  FJsonObjectConverter::JsonObjectStringToUStruct<FResponse_AddCoin>(
                                    JsonString, &AddCoinResponse, 0, 0);

  if (AddCoinResponse.AddUserCoinResult.status == 0){
    UE_LOG(LogTemp, Log, TEXT("AddCoin Succeeded!"));
  }

}
```

HTTP Communication in Unreal Engine

- HTTPService implementation: adding player coins (webservice)

MyPlayerState.h

```
protected:
  ...

  UPROPERTY(Replicated)
  int TotalCoins;

  ...



public:
  ...

  void AddPlayerCoin();
```

HTTP Communication in Unreal Engine

- HTTPService implementation: adding player coins (webservice)

MyPlayerState.cpp

```
void AMyPlayerState::AddPlayerCoin()
{
  TotalCoins++;

  if (HTTPServiceRef)
  {
    FAddCoinRequest AddCoinInfo;
    AddCoinInfo.hash = AuthenticationHash;
    AddCoinInfo.userid = FString::FromInt(PID);
    HTTPServiceRef->AddCoin(AddCoinInfo);
  }

}
```

HTTP Communication in Unreal Engine

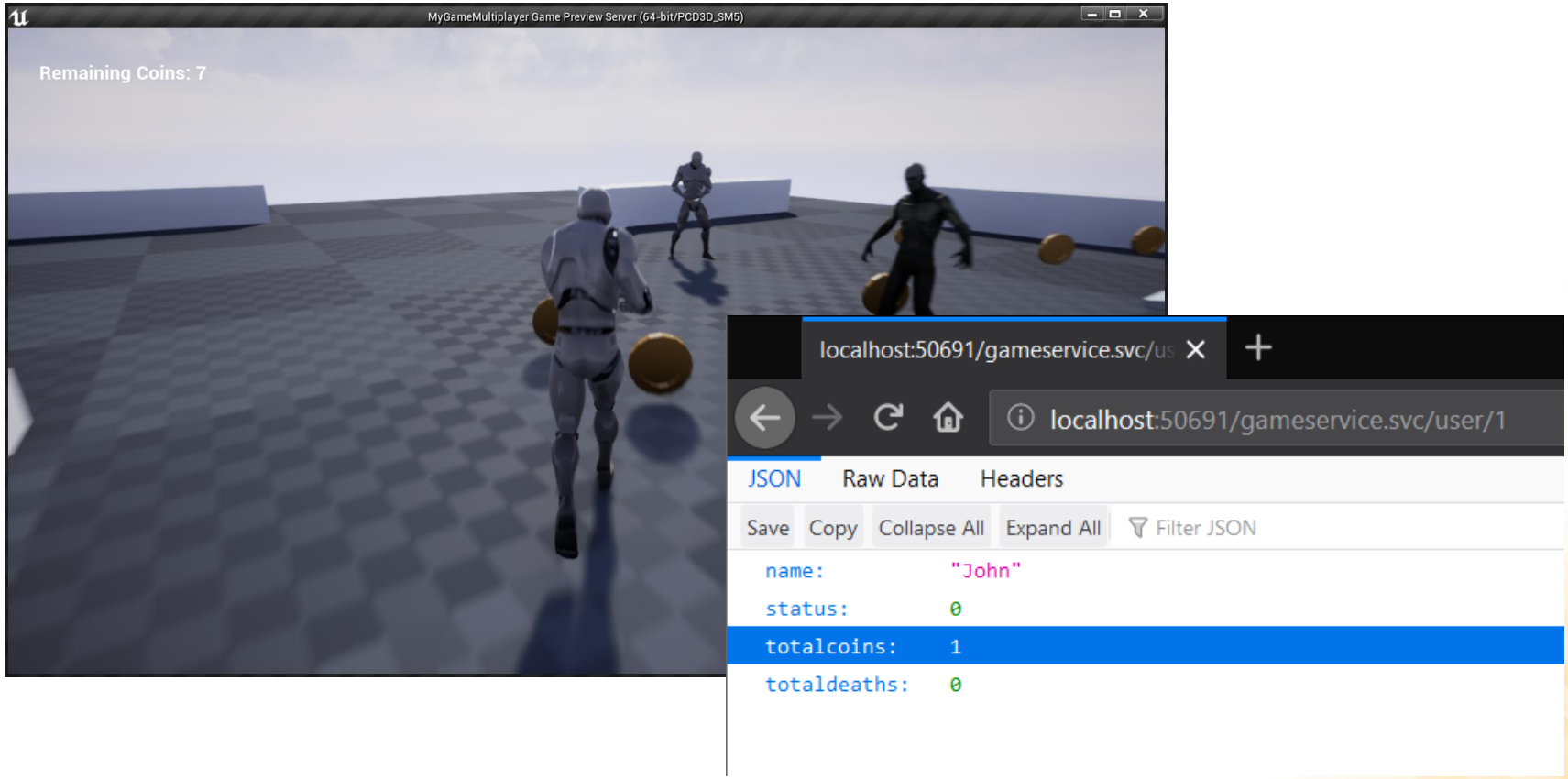
- HTTPService implementation: adding player coins (webservice)

CollectibleCoin.cpp

```
...

void ACollectibleCoin::NotifyActorBeginOverlap(AActor* OtherActor)
{
  Super::NotifyActorBeginOverlap(OtherActor);
  AMyCharacter *mycharacter = dynamic_cast<AMyCharacter*>
                                                    (OtherActor);
  if (mycharacter != nullptr){
    AMyPlayerState* pState = Cast<AMyPlayerState>(
                                    mycharacter->GetPlayerState());
    if (pState){
      pState->AddPlayerCoin();
    }
    Destroy(this);
    PlayEffects();
  }
}
...
```

HTTP Communication in Unreal Engine

- Test the game and check if the total of coins is updated in the server:



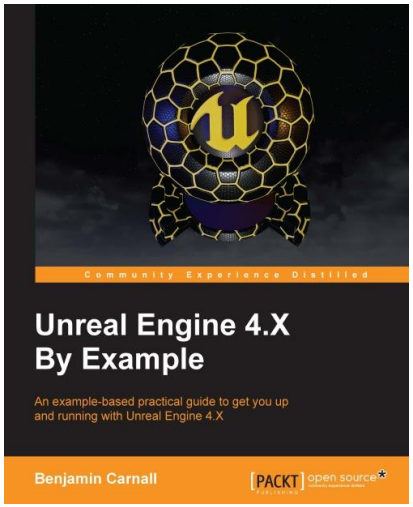
Exercise 1

1) Continue the development of the game and implement the following features:

   a) Implement the process to add player deaths (using the webservice function `AddUserDeath`).
      - The total of deaths must be incremented every time the player dies.
   b) Disable the movement of the enemies while the player is at the login screen.
   c) (Optional - OK+ Challenge) Implement a ranking to be displayed on the game over screen (when the player dies).
      - The ranking must include the players' names, total of coins and total of deaths.
      - The ranking must be ordered by: $\frac{total\ of\ coins}{total\ of\ deaths}$
      - Only the information of the first 10 players must be shown on screen.

Further Reading

- Carnall, B. (2016). **Unreal Engine 4.X By Example**. Packt Publishing. ISBN: 978-1785885532.



- **Web Resources:**
  - HTTP - Unreal Engine - https://api.unrealengine.com/INT/API/Runtime/HTTP/index.html
  - Http-Requests - https://wiki.unrealengine.com/Http-requests