Fejlesztői szolgáltatások

Microsoft Azure

2016.

# Bevezetés

## Célkitűzés

Az labor célja, hogy a hallgatók megismerkedjenek az alkalmazás telemetria gyakorlati alkalmazásaival az Application Insights és a HockeyApp szolgáltatásokon keresztül.

## Előfeltételek

A labor elvégzéséhez szükséges eszközök:

* Visual Studio 2015
  + Web development tools
  + Universal Windows Platform SDK + Emulators
* Azure előfizetés

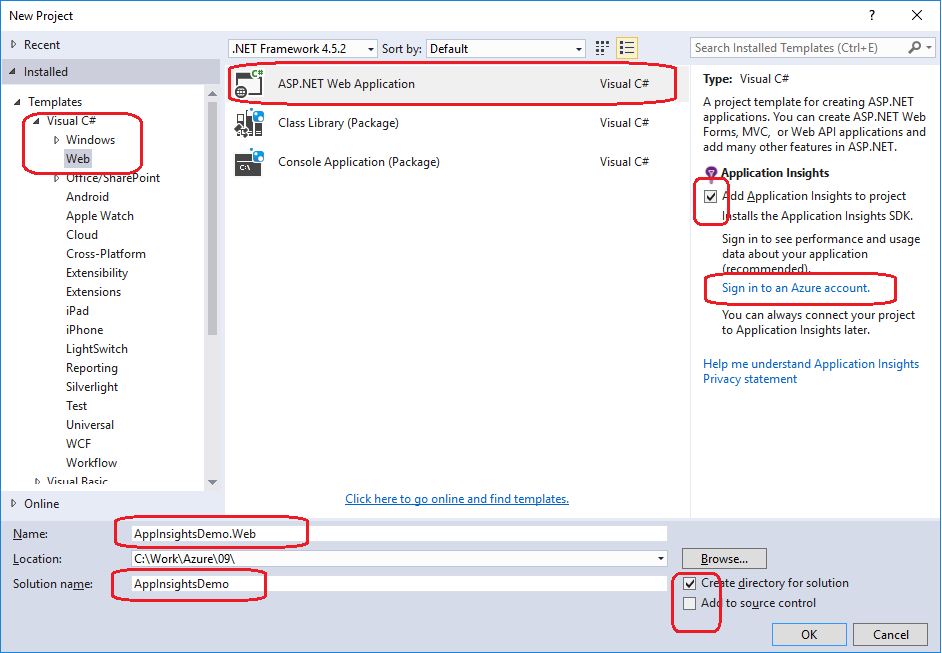
# ASP.NET webalkalmazások telemetriája Application Insights-al

## Webalkalmazás készíte

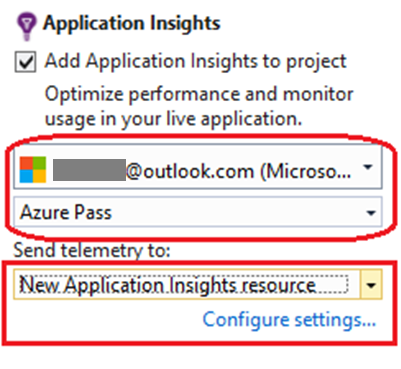
Indítsuk el a Visual Studio 2015 fejlesztőkörnyezetet, majd válasszuk ki a **File / New / Project** menüpontot. A felugró ablakban válasszuk ki a telepített project sablonok közül a **Visual C# / Web / ASP.NET Web Application** opciót.

Project neve legyen **AppInsightsDemo.Web**; Solution neve: **AppInsightsDemo**

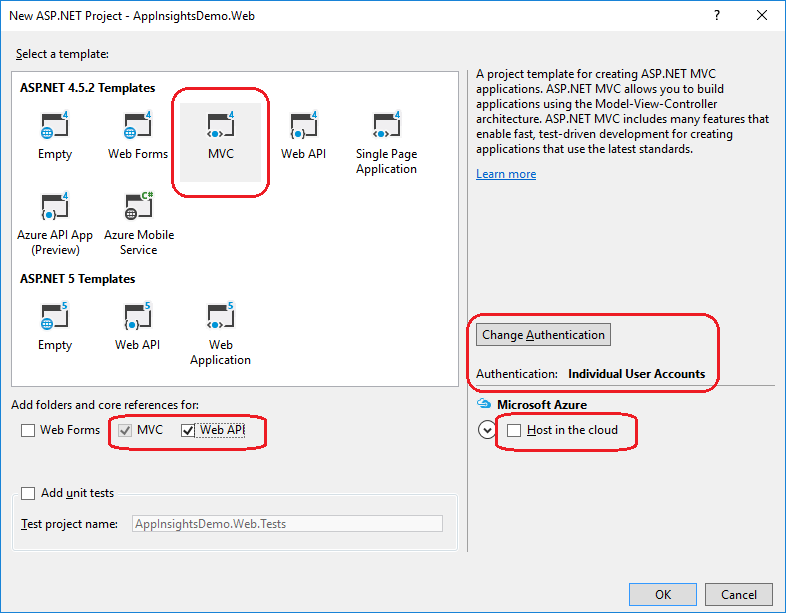
A jobb oldali panelban pipáljuk be **az Add Application Insights to project** opciót majd alatta jelentkezzünk be az Azure teszt fiókunkhoz tartozó LiveID-val, ha még nem lennénk bejelentkezve.



Bejelentkezés után válasszuk ki a megfelelő Azure előfizetést. Lehetőségünk van még az előfizetésen belül resource-hoz rendelni az alkalmazásunkat, mi válasszuk a **New Application Insights resource** opciót. A **Configure settings** menüpontban további csoportosítási beállításokat találunk, melyeket változatlanul hagyhatunk.



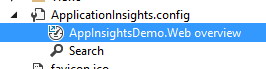
Ezek után a webes project kiválasztásakor válasszuk az ASP.NET 4.x sablonok közül az MVC-t, ha akarjuk bepipálhatjuk a Web Api opciót is. Jobb oldalt az alapértelmezett authentikáció maradjon Individual User Accounts, illetve kapcsoljuk ki a Host in the cloud opciót.



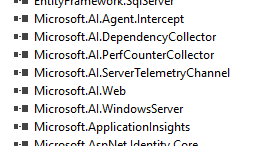
## Application Insights SDK áttekintése

A project legenerálása után vizsgáljuk meg, hogyan telepítette fel Visual Studio a webszolgáltatásunkba a telemetriát. Amit egyből megállapíthatunk:

* Nyissuk meg a Solution Explorer-ben a project gyökerében lévő ApplicationInsights.config fájlt: ez egy xml állomány mely tartalmazza a használni kívánt modulokat, csatornát és egy API kulcsot InstrumentationKey néven.
* A Solution Explorer-ben a konfigurációs fájl lenyitható, ahol gyors linkeket találhatunk az azure portálra és egy a VS-ben beépített egyszerű keresőt.



* A project referenciát megvizsgálva láthatjuk azokat a nuget csomagokat, melyek az ApplicationInsights-hoz szükségesek.



* A web.config fájlba került be egy http module, ami az IIS webszerver szolgáltatásain keresztül fogja naplózni az eseményeinket.

<system.web>

...

<httpModules>

<add name="ApplicationInsightsWebTracking"   
 type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule, Microsoft.AI.Web" />

</httpModules>

</system.web>

<system.webServer>

<modules>

<add name="ApplicationInsightsWebTracking"   
 type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule, Microsoft.AI.Web"  
 preCondition="managedHandler" />

</modules>

...

</system.webServer>

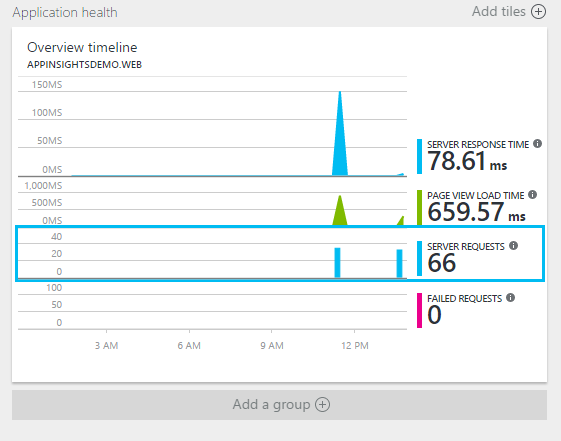
* A Views / Shared / \_Layout.cshtml fájlban a html fejbe bekerült egy javascript kód részlet, aminek a célja a kliens oldali telemetriák gyűjtése, így a javascript kódokban keletkező kivételek is naplózásra kerülnek.

## Alapértelmezett telemetriák kipróbálása

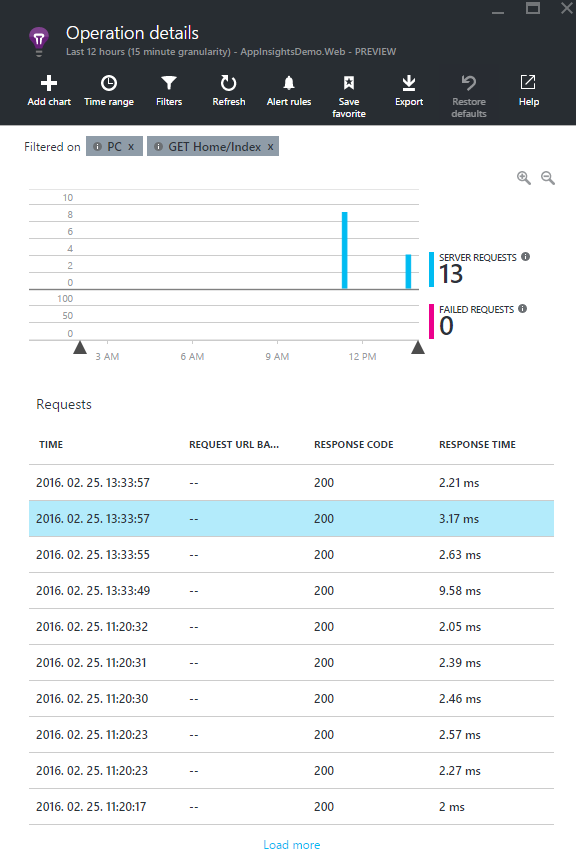
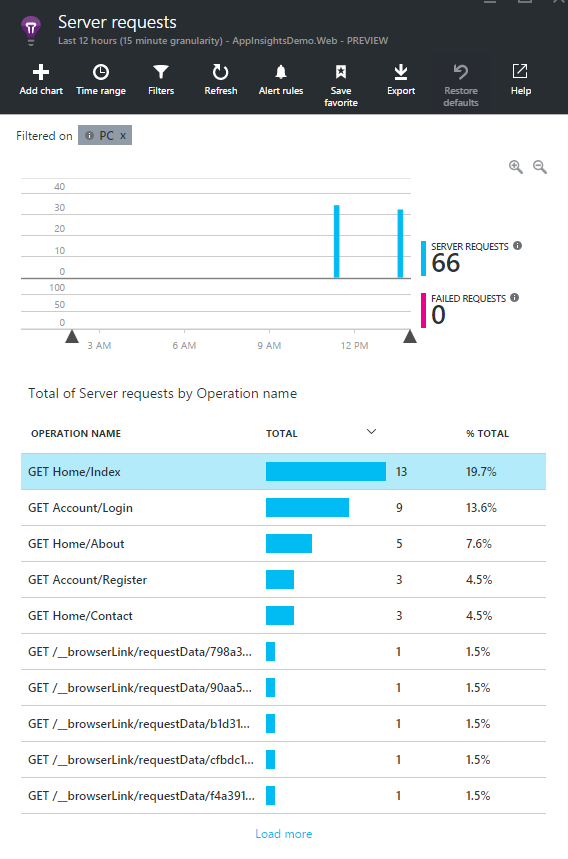
Indítsuk el az alkalmazásunkat az F5 megnyomásával és navigáljunk a menüpontok között ide-oda, hogy generáljunk néhány http kérést. Az Output ablakban (View / Output) láthatjuk, hogy az alkalmazásunk telemetriai adatokat küld JSON üzenetek formájában.

Nyissuk meg az azure portált (<https://portal.azure.com>), jelentkezzünk be és győződjünk meg arról a profilunkra kattintva, hogy jó előfizetéshez tartozó adatokat böngészünk.

A bal oldali menüből az Application Insights menüpontot kiválasztva látnunk kell a megjelenő listában a webalkalmazásunkat. Ez kiválasztva egy áttekintő nézethez jutunk, melyet tetszés szerint testre is szabhatunk.



A portálon a telemetriát nagy mélységben tudjuk böngészni, amik jobb oldalon megnyíló panelek formájában jelennek meg.

## Saját esemény és metrika készítése

Készítsünk saját eseményt a regisztrációkról, amelyben naplózzuk az esemény sikerességét, és erre vezessünk be egy új metrikát.

Nyissuk meg a Controllers / AccountController osztályt és ebben keressük ki a Register(RegisterViewModel model) metódust. Ebben a felhasználó generálása után a TelemetryClient segítségével küldjünk el egy saját eseményt.

public async Task<ActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

var user = new ApplicationUser { UserName = model.Email, Email = model.Email };

var result = await UserManager.CreateAsync(user, model.Password);

var tc = new TelemetryClient();

var resultText = result.Succeeded ? "Success" : "ERROR: " + result.Errors.First();

var properties = new Dictionary<string, string> { {"Email", model.Email }, {"Result", resultText} };

var metrics = new Dictionary<string, double> { {"Registration Success", result.Succeeded ? 1 : 0} };

tc.TrackEvent("Registration", properties, metrics);

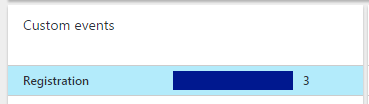
...

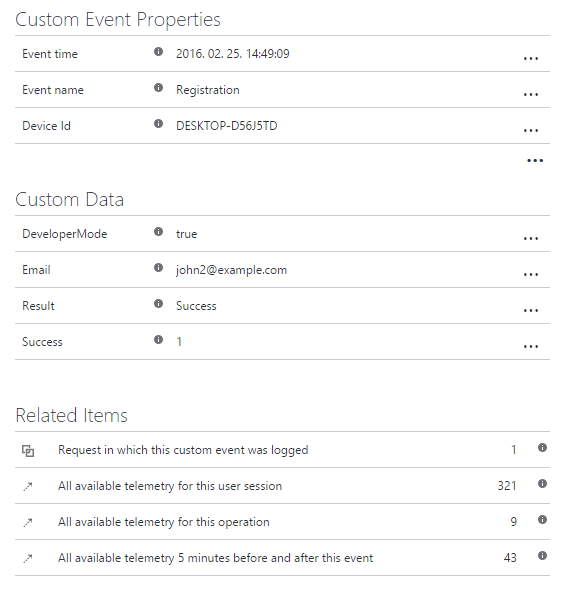
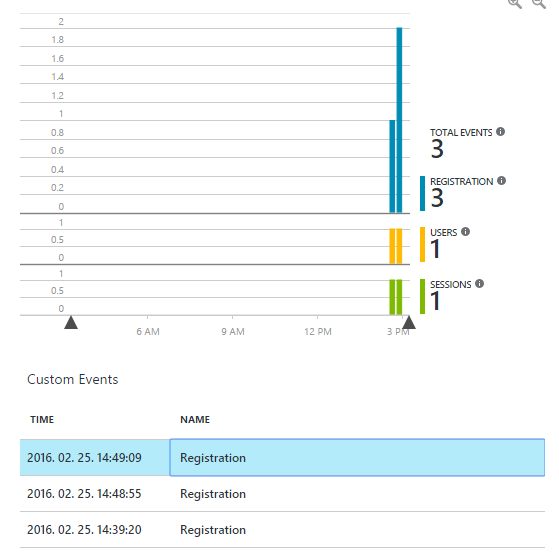
}

}

A hiányzó using-okat vegyük fel úgy, hogy miután begépeljük az osztály teljes nevét a CTRL + . billentyűkombinációval előhívható gyors műveletekkel legeneráltatjuk azt.

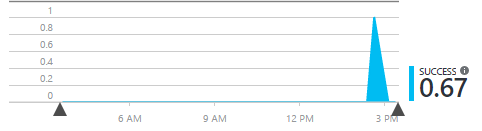
A properties között saját tulajdonságokat csatolhatunk az eseményhez, a metrikákkal pedig grafikusan is megjeleníthető adatokat vehetünk fel. Futtassuk és próbáljuk ki a regisztrációs folyamatot sikeres és sikertelen lefutással is. Sikertelen eseményeket például úgy tudunk generálni, ha nem tartjuk be a jelszó mezőre vonatkozó követelményeket.



Az Azure portálon a Beállítások / Használat panelben fog megjelenni egy új csempe a saját eseményekkel, amelyeket részletesebben is meg lehet tekinteni.

A metrikákat Metrics Explorer-ben tudunk hozzáadni a megnyíló panel üres diagrammjára kattintva, majd a megfelelő metrikát kiválasztva.



## Hibák naplózása

Generáljunk az alkalmazásunkkal hibát (szándékosan): Keressük ki a HomeController Contact metódusát, amiben indítsunk el egy hibás adatbázis beszúrást a User táblába. (Ennek logikailag semmi értelme, csak egy hibát szeretnénk látni a portálon, aminek vannak külső függőségei is például egy SQL kérés.)

public ActionResult Contact()

{

ViewBag.Message = "Your contact page.";

using (var db = new ApplicationDbContext())

{

db.Users.First().Roles.Add(new IdentityUserRole() { RoleId = "valami hibás" });

db.SaveChanges();

}

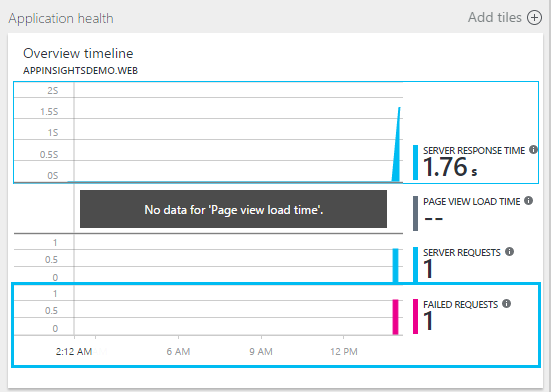
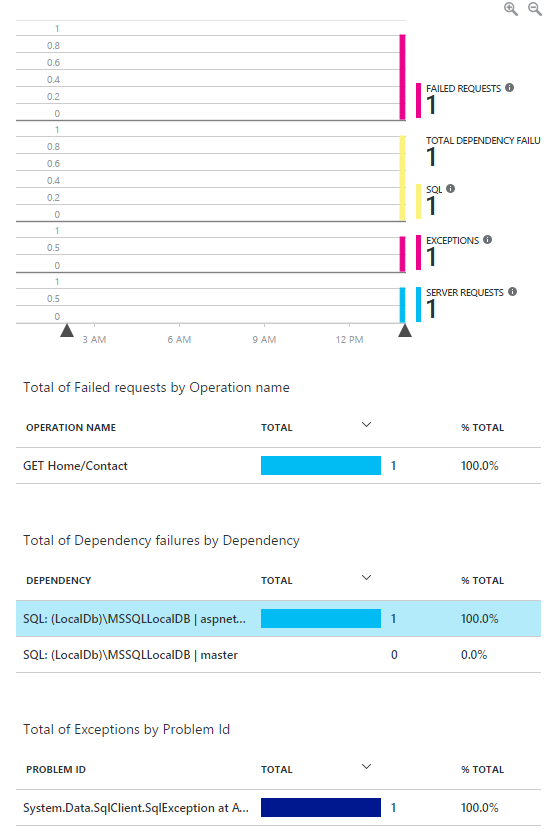
return View();

}

Előkövetelmény, hogy fenti regisztrációs feladatokat már leteszteltük, így létezik az adatbázis. Ebben a példában az egyik felhasználóhoz akarunk egy nem létező szerepkört csatolni. De mivel a megadott szerepkör nem létezik ilyen azonosítóval, a rendszer elszáll SqlException-el, mert az adatbázis nem tudja a beszúrás során feloldani a külső kulcs kényszert.

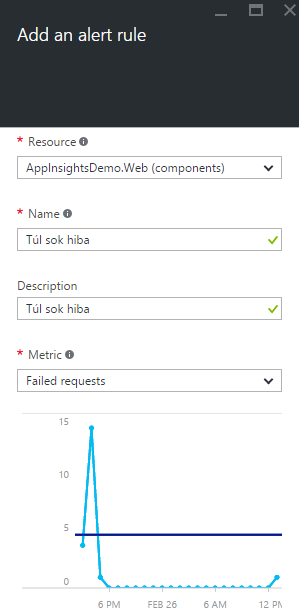
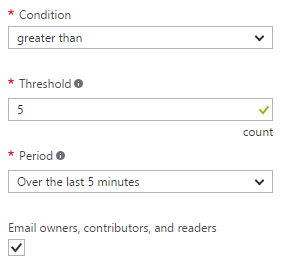
Próbáljuk ki, futtatás (F5) után kattintsunk a Contact menüpontra a weblapon! Yellow screen of death-t kell kapjunk a fenti hibával.

Tekintsük meg a portálon a hibát. A dependency szekcióba azokat a külső eseményeket láthatjuk, amelyek a kérés során egy külső rendszer felé mutatnak, pl.: SQL utasítások, HTTP kérések.

## Alert létrehozása

Az eseményekről értesítéseket készíthetünk, amelyek megadott események során emailt küldenek vagy elsütnek egy webhook feliratkozást. Készítsünk a portálon egy új alertet, ami az 5 percen belüli 5-nél több hiba esetén jelez.

Süssük el többször a Contact menüben lévő hibánkat, és nézzük meg kapunk-e emailt.

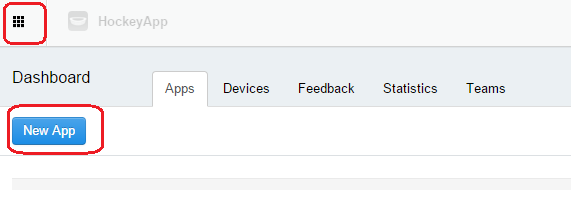
# Mobil alkalmazások telemetriája

A mobil alkalmazások esetében a Microsoft ajánlása a HockeyApp szolgáltatás, mely alapvető hibajelentési, monitorozási, és béta-disztribúciós szolgáltatásokat nyújt. Az Application Insights SDK egy régebbi változata is elérhető mobil kliensekre, de annak támogatása ki fog futni. Talán az egyik kivétel a Windows 10 UWP alkalmazások, melyek egy friss Application Insights SDK-t használnak, de az adatokat a Windows Dev Center felületén lehet megtekinteni.

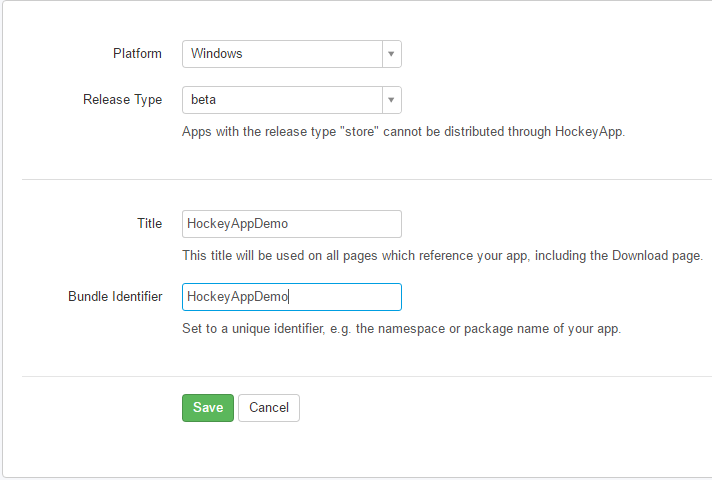
## HockeyApp használata

Regisztráljunk vagy jelentkezzünk be közösségi fiókunkkal a <http://hockeyapp.net/> oldalon. Első belépés / regisztráció során pipáljuk be az **I’m a developer** opciót is, hogy hozzáférjünk a gyűjtött adatokhoz. Regisztráció esetén erősítsük meg az email címünket.

A bal felső sarokban a Dashboard ikonra kattintva nyissuk meg a dashboardot, és ott kattintsunk a New App gombra és hozzunk létre egy új alkalmazást a HockeyApp szolgáltatásban.

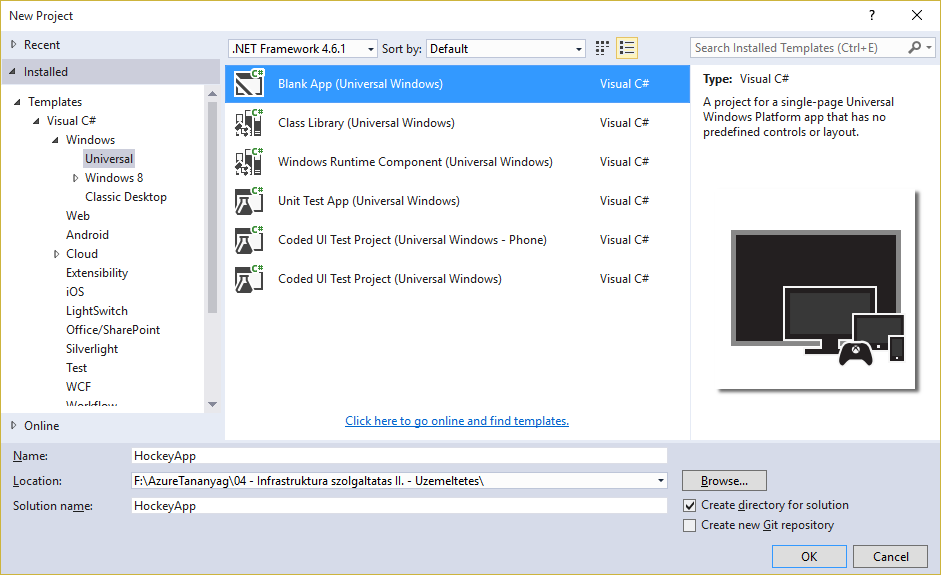


A felugró ablakban feltölthetnénk már készen lévő alkalmazásokat is, de most válasszuk a **Create the app manually instead.** opciót. A következő ablakban válasszuk a **Windows** platformot, a **Title**-nek és a **Namespace**-nek pedig adjuk meg a következőt: **HockeyAppDemo**



## UWP Kliens

Ezután váltsunk át a Visual Studio-ra és hozzunk létre egy új projectet a solution-be. A projekt típusa legyen **Visual C# / Windows / Universal / Blank App**. A projekt neve legyen **HockeyApp**.



Vegyünk fel a project referenciái közzé egy NuGet csomagot (References (jobb gomb) / Manage NuGet packages) és a Browse fülön keressünk rá a HockeySDK.UWP csomagra, majd telepítsük azt fel.

Nyissuk meg a projectben található App.xaml.cs fájlt (App.xaml-t lenyitva) és a benne található App osztály konstruktora végére illesszük be HockeyClient inicializációját.

public App()

{

this.InitializeComponent();

this.Suspending += this.OnSuspending;

HockeyClient.Current.Configure("Your\_App\_ID");

}

A configure metódus paraméterét cseréljük ki a saját alkalmazásunk azonosítójára, amit a HockeyApp weboldalán az alkalmazás adatai között találhatunk meg.



## Hibák küldése

Rakjunk ki a MainPage.xaml oldal felületére egy gombot, ami az eseménykezelőjében kivételt dob.

<StackPanel>

<Button Content="Hiba!" Click="Error\_OnClicked"/>

</StackPanel>

A MainPage.xaml.cs-be helyezzük az eseménykezelőt.

private void Error\_OnClicked(object sender, RoutedEventArgs e)

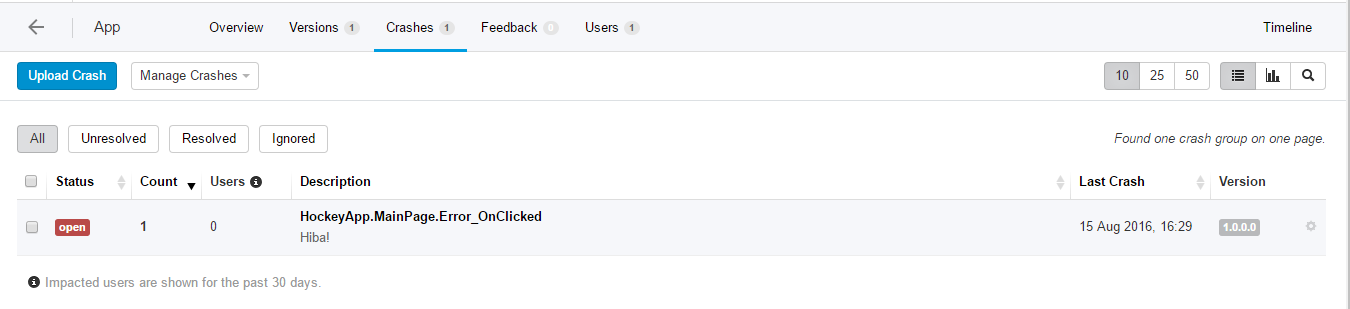
{

throw new Exception("teszt hiba!");

}

Próbáljuk ki az alkalmazást és kattintsunk a hiba gombra. A debuggolás során megálló program futást engedjük tovább a Continue gombbal (F5), amíg az app ki nem lép.

Ha ezután újra elindítjuk az alkalmazást, akkor a HockeyApp Dashboardon láthatjuk, hogy a hibaüzenetet naplózta a rendszer:



Megjegyzés: A HockeyApp támogat több funkciót is Windows Phone 8.1 platformon (például visszajelzés küldése a fejlesztőnek), azonban ezek UWP-n még nem érhetőek el a laboranyag elkészítésének időpontjában (2016. augusztus).