PRINCIPAL

AT CRM MS Dynamics, jako POC pro klienty

V rámci interního rozvoje kompetence Microsoft v naší firmě PRINCIPAL jsme se rozhodli vytvořit automatizované POC pro CRM systém Microsoft Dynamics 365. Cílem bylo připravit funkční, spolehlivou a technicky kvalitní ukázku, kterou můžeme využívat při prezentacích pro klienty a zároveň jako základní stavební kámen pro budoucí testovací scénáře a automatizace.

Projekt vznikl jako čistě interní iniciativa a zároveň jako prostor pro hlubší porozumění specifikům Dynamics 365 z pohledu automatizace. V textu níže popisujeme celý průběh vývoje, použitou metodiku, technologie i překážky, se kterými jsme se museli vypořádat.

Pro ty, kteří neznají MS Dynamics – jedná se o sestavu podnikových aplikací které zahrnují systémy pro řízení vztahů se zákazníky (CRM – Customer Relationship Management), řízení podnikových zdrojů (ERP – Enterprise Resource Planning) a další, jako například řízení maloobchodu a personální. MS Dynamics 365 je hlavní platformou integrují tyto aplikace do cloudu a umožňuje společnostem automatizovat procesy v daném odvětví.

Nástrojem pro automatizaci se stal Playwright – automatizační framework od společnosti Microsoft. Jedná se o rychlý a jednoduchý nástroj, který lze použít v několika programovacích jazycích a umožňuje spouštět testy na několika emulovaných prohlížečích a jejich mobilních mutacích (Pozn. nejedná se o reálně mobilní prohlížeče, pouze o upravené rozlišení již zmíněných prohlížečů.). Pro mě preferovanou volbou programovacího jazyka byl jeden z nejrozšířenějších programovacích jazyků – JavaScript s nadstavbou TypeScript.

Použité metodiky a principy

Jasná volba principu pro vývoj při každé automatizace v Playwright je POM, který jsem i v případně automatizace MS Dynamics využil. Případně SOM (Service Object Model) pro vývoj automatizovaných testů backendu (Servisy), nebo také Fluent API, které mi není úplně blízké, ale nepotkal jsem se se situací, kde by oproti SOM byl tento přístup vhodný na opravdu velké projekty.

Ze sféry vývojářů automatizovaných testů nemusím moc představovat model POM (Page Object Model), skupina vývojářů bude znát princip OOP (Object Oriented Programming) a pro širší veřejnost nezainteresovanou v programování by mohl být dostačující následující velmi zjednodušený popis:

Page Object Model je postup vývoje a seskupování kódu podle stránek webové aplikace, kterou automatizujeme. Můžete si představit třeba E-shop, který má určitě nějakou homepage, produktové kategorie, detail produktu, košík, objednávku a další. S takovýmto rozdělením webové aplikace také pracujeme – vytvoříme si nové soubory například se zjednodušeným názvem dané stránky a do těchto souborů následně zapisujeme náš kód – neboli „rozsypaný čaj", jak tomu říká moje rodina

Automatizace Microsoft Dynamics 365 se ukázala jako ideální příležitost propojit naši kompetenci v oblasti Microsoft technologií a testování. Výsledkem je funkční, stabilní a snadno rozšiřitelná ukázka, která nám pomáhá nejen interně, ale i při prezentaci klientům.“

Dalším přístupem, který je důležité vyzdvihnout je atomizace testů, vývojáři budou znát podobný přístup pod názvem „Modulární programování“. Principem atomizace testů je příprava kroků testu tak, aby každý jednotlivý test byl nezávislý na předchozích nebo následujících testech. V praxi to může zjednodušeně znamenat, že pokud potřebujeme pro automatizaci například uživatelský účet, tak ho v rámci daného testu vytvoříme a použijeme pro scénáře a po dokončení testu proběhne čištění dat.

Jelikož MS Dynamics používá doménové ověření účtu, tak bylo potřeba zajistit i to, aby se citlivé údaje neukládaly do repositáře a moje heslo nebylo vidět nikým dalším. Nejjednodušší způsob je použití konfiguračního souboru env. Základem tohoto souboru je definování nějakého rozumného názvu například pro heslo a za tento název hned samotné heslo. V kódu následně používáte vámi definovaný název, který voláte a ten vám vrátí hodnotu hesla. Zajištění toho, že citlivé údaje nikam neutečou se zajistí tak, že se tento konfigurační soubor ignoruje při ukládání do repositáře – například GitLab, GitHub, nebo Azure DevOps.

Vývoj testů a překážky

První problém u MS Dynamics, se kterým jsem se setkal bylo při paralelizaci exekuce – spuštění testu na 4 prohlížečích najednou. I přes to, že jsem využil techniky, které by tomu měly předejít a postarat se o stabilitu testů, tak se to nepotkalo s úspěchem kvůli metodě autorizace a zabezpečení. Našly se nakonec 2 řešení, které jsem mohl využít a vybral si jedno z nich. Zajímavostí bylo, že tato chyba se mi stávala pouze na prohlížečích WebKit (Safari) a MS Edge.

Metoda, kterou jsem si zvolil byla na principu ošetření nespecifikované chybové hlášky, která vyskakovala zcela náhodně. Řešení tedy spočívalo v odchytávání správných lokátorů v kombinaci s polováním.

Stabilita test scénářů a data

iFrame

Jak už jsem výše zmínil, problematika iFrame pro některé není standardní, protože iFrame v podstatě

nejsou „součástí“ aktuální aplikační HTML struktury, představte si je zjednodušeně jen jako odkaz na jednou stránku, nebo zobrazení, které se vám dynamicky načte, pokud dojde k aktivaci iFrame.

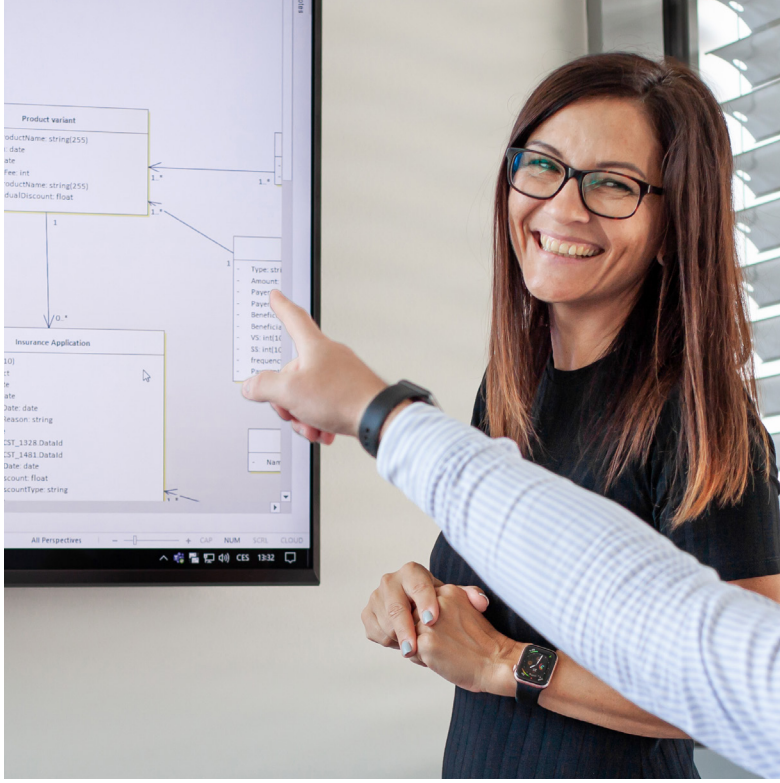
Přístup k iFrame je tedy odlišný, než je to standardně u lokátorů, nebo vyhledávání textu na stránce, protože prvním úkonem k tomu, jak interagovat s iFrame je zaměření kořene iFrame jako takového. V praxi to není žádná věda, pokud uvidíme iFrame a uvnitř potřebujeme provést akci, tak se před tím nejdřív do iFrame zanoříte a teprve potom klikáte, kontrolujete, nebo vyplňujete.

Dynamické prvky

Obecně dynamické prvky v nástroji Playwright už jsou minimální problém, avšak narazil jsem na jeden zajímavý, který uměl z perfektně stabilního testu udělat flaky testovací sadu.

Konkrétně v případě MS Dynamics se jedná o pole vyhledávání vytvořených záznamů. Dám příklad – v rámci MS Dynamics si vytvářím záznam kontaktu pomocí načtené vizitky, po vytvoření kontaktu na tento záznam vytvářím například email, nebo úkol. Problém nastávám v momentě, kdy u vytváření emailu, nebo úkolu potřebuji vyhledat testem vytvořený kontakt. Kromě orientace ve zbytečně složité struktuře html, obrovského množství lokátorů, tak dynamické načtení modalu s jeho daty mi přidalo nejeden šedivý vlas na hlavě. Z nejasného důvodu totiž docházelo k tomu, že data se vůbec nenačetly i když byl kontakt vytvořený, manuálně dohledatelný, ale playwright nebyl schopný vyhledat žádný záznam a zobrazilo se jen prázdné okno. Bohužel na přesné zjištění příčiny nebylo více času, ale trochu to dávám za vinu i sandbox prostředí i kterém budu mluvit níže.

> **Projekt nám potvrdil, že i komplexní platformy jako Dynamics 365 lze automatizovat efektivně a s důrazem na škálovatelnost a kvalitu.“**



Komplikovaná HTML struktura

Jak už jsem naznačil, HTML struktura v MS Dynamics je značně košatá a komplikovaná na přehlednost. Většinou nebyl ani problém najít potřebné lokátory podle kterých se Playwright orientoval, tak největší problémy dělaly právě iFrame, dynamické prvky a do načítající se elementy.

Velkou výhodou MS Dynamics byla také implementace testovacích ID, což značně napomohlo a zrychlilo přípravu automatizace a kroků testů. Měl jsem až později menší problém, že část MS Dynamics, která měla očividně zastaralou strukturu html, neobsahovala testovací ID.

Lokátory

Již jsem načal téma lokátorů, konkrétním problémem byla teda část aplikace, která nebyla pokrytá unikátními testovacími ID, ale byl tu i jiný problém, a to struktura takových test ID a jejich přehlednost.

Přiložím příklad:

[data-id="primarycontactid.fieldControl-Lookup_primarycontactid"] [data-id="primarycontactid.fieldControl-LookupResultsDropdown_primarycontactid_textInputBox_with_filter_new"]

Toto jsou 2 lokátory, přes které se Playwright orientoval. V menším množství to určitě není problém, ale zkuste si vyhledávat desítky až stovky takových elementů, když html struktura obsahuje přes 4000 řádků kódu – moje doporučení k takovému případu je se obrnit trpělivostí a spoustou času.

Komplikace a implementace opatření

Sandbox prostředí a HW prostředky

Kromě již sepsaných standardních komplikací tu byla i jedna, která alespoň u prvotních spouštění testů v daném dnu způsobovala kompletní selhání. Jednalo se o prostředky alokované pro prostředí. Bohužel pro nás mělo toto prostředí velmi omezené prostředky na provoz – z povahy sandbox prostředí provozované MS je to určitě cenově nejvíc optimální přístup. Zkrátka pokud je prostředí neaktivní a není na něm žádný provoz, tak se prostředky na stáhnou na úplné minimum a aplikace se může zprvu zdát velmi pomalá. Značný problém to byl pro mě hned po ránu, kdy jsem na AT začal pracovat a první testy vypadaly příšerně a ze začátku nebylo hned jednoznačné, že se jedná o toto HW omezení.

Jedním řešením tohoto problému bylo spuštění prvního testu s vědomím, že spadne a dalším bylo navýšení základních timeoutů pro konkrétní akce playwrightu na „rozumných" 20 sekund. Druhý způsob řešení umožnil i při první exekuci na vychladlém prostředí to, že byl dostatek času na rozehřátí prostředí, aby test bylo možné dokončit.

AI a dynamické návrhy hodnot v polích

Náš kamarád AI a jeho vtíravost i tam, kde jeho špatná implementace dělá více problémů než užitku – MS Dynamics má pár částí, kde byla právě implementace AI víc na škodu než, aby to pomohlo, nebo se to dalo alespoň ignorovat.

Konkrétní problém, na který jsem u AT narazil byly vyplňovací pole… jaký nesmysl, vždyť je to základní operace, ALE co někoho v MS napadlo udělat, je to, že když vytváříte nějaký záznam, například kontakt na osobu, tak po otevření se vám zobrazí přibližně 30 polí, které můžete vyplnit a v tenhle moment nastupuje problém s AI, kdy až po cca 10 až 15 vteřinách od otevření formuláře se vám v pravém horním rohu ukáže tlačítko „Apply suggesions". Kdo s touto hrůzou pracoval, tak už může tušit. Toto tlačítko se Vám ukáže v moment, kdy se také všechny náhodně vybrané pole ve formuláři zakryjí doporučením od AI na hodnoty, které si myslí, že by tam měly být vyplněné… Problém u automatických testů byl ten, že pokud to není ošetřeno, tak v momentě, kdy se zobrazí doporučení AI a již máte automatem vyplněný formulář hodnotami, tak tyto hodnoty jsou úplně přepsány, nebo smazány a ani vás to nepustí dále dokud nepotvrdíte, nebo nezrušíte změny od AI, tak či tak vám zmizí již vyplněná data.

Na závěr:

- CRM aplikace jsou výzva i zábava k automatizaci, ale chce to hodně kávy.
- AI pomáhá, ale jeho implementace, nebo práce s dany přímo odráží to, jak se nad tím, kdo to implementuje zamyslí a taky s jakými daty AI pracuje.
- Kdo říká, že už umí všechno, ten toho ještě moc nezažil.
- Vždy připravte vaše projekty na možnost škálování, nikdy nevíte, do jakých rozměrů vám to vyroste pod rukama.

Automatizace Microsoft Dynamics 365 se ukázala jako ideální příležitost propojit naši kompetenci v oblasti Microsoft technologií a testování. Výsledkem je funkční, stabilní a snadno rozšiřitelná ukázka, která nám pomáhá nejen interně, ale i při prezentaci klientům.

Projekt nám potvrdil, že i komplexní platformy jako Dynamics 365 lze automatizovat efektivně a s důrazem na škálovatelnost a kvalitu.

Pokud zvažujete automatizaci v MS Dynamics nebo hledáte spolehlivé řešení v oblasti testování, rádi s vámi probereme možnosti a ukážeme naše přístupy. Ozvěte se nám na testing@principal.cz.

Autor článku:
Tadeáš Topol,
QA konzultant - automatizace