Fakulta informačních technologií
Vysoké učení technické v Brně

Internetové aplikace – 2. projekt

Rozšíření informačního systému studentského klubu U Kachničky

17. dubna 2023

Ondřej Ondryáš

Obsah

1 Úvod

Tento projekt řeší reimplementaci a rozšíření částí informačního systému *KIS* (Kachní informační systém), který se používá ve studentském klubu U Kachničky na FIT VUT. Systém poskytuje pokladní služby, zajišťuje správu skladových zásob občerstvení, umožňuje podávání elektronických přihlášek ke členství v klubu a správu členských příspěvků. Součástí je také webová aplikace pro návštěvníky klubu.

Systém je v provozu cca 4 roky, během kterých byly odhaleny některé nedostatky jeho dosavadního návrhu. Jejich důsledkem je především obtížnost dalšího rozvoje systému a rozšiřování o další služby, dále také jistá uživatelská nepřívětivost pro personál klubu i návštěvníky.

V rámci tohoto projektu byla navržena a implementována centrální autentizační služba, která s využitím standardizovaných protokolů *OAuth 2.0*[1] a *OpenID Connect 1.0*[2] poskytuje dalším částem systému prostředky pro autentizaci uživatelů i aplikací. Dále byla navržena a implementována služba umožňující automatizaci procesu vydávání občerstvení, které vyžaduje delší přípravu. Součástí je také návrh integrace se stávajícími službami.

2 Dosavadní stav

V původní koncepci byl systém zamýšlen především jako prostředek pro zajištění provozních záležitostí klubu. Potřeba celého systému vycházela zejména z nutnosti zabezpečit klub po legislativní stránce: umožnit jednoduše návštěvníkům podávat přihlášky ke členství v elektronické podobě s ověřenou identitou, sledovat poskytované členské příspěvky, jakož i tok financí a majetku. Kromě registrací se však prakticky nepočítalo s další interakcí s koncovými návštěvníky.

Systém v původním návrhu tvořily čtyři komponenty: backendová služba *KIS Backend*, aplikace pokladního systému *KIS Operator*, aplikace pro administraci *KIS Admin* a aplikace pro registraci členů klubu *KIS User Control Point (UCP)*. Poslední jmenovaná aplikace se již nevyužívá, princip registrací byl změněn a do systému přibyla samostatná zákaznická webová aplikace *Kachna Online*. Se systémem komunikuje také aplikace pro zobrazování nabídky *KIS Monitor*.

Backendová služba zajišťuje kompletní správu uživatelů (členů) i operací nad skladovými zásobami a členskými příspěvky. Implementována je v jazyce Python s využitím „spec-first“ frameworku pro budování REST API *Connexion*[3] (ten staví na bázi frameworku *Flask*[4]) a ORM *peewee*[5] nad databází PostgreSQL. Frontendové aplikace s ní komunikují pomocí HTTP REST API. Pro autentizaci jsou použity JWT Bearer tokeny vydávané backendovou službou, jejich obsah, životní cyklus a způsob autentizace uživatelů jsou nestandardní (viz dále).

Frontendové aplikace Operator a Admin využívají konceptu SPA, jsou postaveny na frameworku *Angular*[6]. Komunikují přímo s backendovou službou.

Aplikace UCP byla desktopová aplikace implementovaná v jazyce Python s využitím GTK. Počáteční návrh počítal s registracemi uživatelů na specializovaném kiosku v klubu. Důvodem bylo, že součástí registrace je také párování uživatelské karty, k čemuž musí registrační bod komunikovat se speciálním hardwarem. Tento přístup se později ukázal jako značně nepraktický. Registrační proces byl proto pozměněn tak, že se

[1] `https://oauth.net/2/`
[2] `https://openid.net/connect/`
[3] `https://connexion.readthedocs.io/en/latest/`
[4] `https://flask.palletsprojects.com/en/2.2.x/`
[5] `https://docs.peewee-orm.com/en/latest/`
[6] `https://angular.io/`

uživatelé registrují na svém zařízení (pomocí aplikace *Kachna Online*) a kartu si párují s asistencí obsluhy v klubu.

Kachna Online je komplexní aplikace s vlastním backendem na platformě ASP.NET Core a frontendem na bázi frameworku Angular. Uvedena byla asi dva a půl roku po vzniku systému KIS, a to v reakci na nové požadavky týkající se správy klubu a komunikace s jeho návštěvníky.

2.1 Registrace členů klubu

Základním požadavkem na systém KIS byla podpora správy registrovaných členů klubu. Občerstvení v klubu je dostupné pouze sympatizujícím členům Studentské unie FIT VUT v Brně, která klub provozuje. Sympatizujícím členem se libovolná fyzická osoba stává typicky podáním přihlášky v elektronické podobě pomocí systému KIS. Členové se při nákupu ověřují čipovou kartou, která je v systému spojena s jejich účtem.

Hlavním požadavkem tedy je jednoduchost „registrace“ (formálně podávání přihlášek ke členství), při kterém je ale nutné dostatečně spolehlivě zajistit identitu člena. K tomuto účelu byla využita česká akademická federace identit **eduID.cz**[7]. Uživatel při přihlašování vybírá svého poskytovatele identity (dále také *IdP*, identity provider) zapojeného v této federaci, přihlásí se v jeho informačním systému svými údaji a poskytovatel předá úzkou množinu informací o identitě žadateli, v tomto případě systému KIS. Pro systém je identita důvěryhodná, poskytovatel ji digitálně podepisuje. Poskytovatelem identity je ve federaci většina českých akademických institucí, mezi nimi i VUT v Brně, což je v kontextu studentského klubu na FIT ideální.

Pro přihlašování ve federaci je využit standard *SAML 2.0*[8], což je framework pro bezpečnou výměnu autentizačních a autorizačních informací mezi poskytovateli identity a služeb.

2.2 Proces registrace

Registrace nových členů v systému KIS s využitím federace eduID sestává z následujících kroků (při zanedbání možných chybových stavů):

1. Aplikace zavolá KIS endpoint `/auth/eduid` a předá mu jistou návratovou adresu.
2. KIS si poznamená požadavek, vrátí interní identifikátor sezení a adresu WAYF/DS[9] služby – rozcestníku federace eduID pro výběr IdP.
3. Aplikace přesměruje uživatele na WAYF/DS službu.
4. Uživatel vybere svého IdP.
5. WAYF/DS služba přesměruje uživatele na KIS endpoint `/auth/eduid/discovery`, v požadavku uvede adresu cílového IdP.
6. KIS sestaví SAML autentizační požadavek a přesměruje uživatele na adresu cílového IdP, v požadavku uvede zakódovaný autentizační požadavek a jeho digitální podpis.
7. IdP požadavek zpracuje a provede autentizaci uživatele: pokud dosud nebyl přihlášen, typicky jej přesměruje na přihlašovací formulář.

[7] `https://www.eduid.cz/`
[8] `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security`
[9] Where Are You From / Discovery Service

8. IdP zašle po úspěšné autentizaci POST požadavek na KIS endpoint `/auth/eduid/assertion`, ve kterém zahrne SAML odpověď s informacemi o identitě uživatele.
9. KIS odpoví adresou pro přesměrování – jde o návratovou adresu z bodu 1 doplněnou v query parametru o identifikátor sezení z bodu 2.
10. IdP přesměruje uživatele na dodanou adresu.
11. Aplikace uživateli zobrazí stránku s potvrzením registrace.
12. Uživatel explicitně odsouhlasí podání přihlášky k sympatizujícímu členství v SU FIT.
13. Aplikace zašle POST požadavek na KIS endpoint `/auth/eduid/register`, ve kterém zahrne identifikátor sezení.
14. KIS vytvoří uživatele, vytvoří autentizační token typu JWT a zašle jej aplikaci.

2.3 Proces přihlášení

KIS nabízí dva způsoby přihlášení uživatele: pomocí eduID, nebo pomocí čipové karty a PIN kódu. První způsob je univerzální, uživatel získává token se všemi právy, které mu náleží. Druhý způsob se používá pro přihlašování obsluhy do pokladní aplikace, při kterém by přihlašování jménem a heslem na dotykovém tabletu bylo příliš pomalé a nepraktické. V tomto případě má navrácený token práva omezená na úroveň nutnou pro provádění pokladních operací.

Při přihlašování pomocí eduID je proces prakticky stejný jako při registraci, liší se od bodu 11, kde aplikace nezobrazuje speciální obsah, ale rovnou posílá POST požadavek na endpoint `/auth/eduid/login`, který vrací JWT token.

Při přihlašování pomocí karty je proces jednodušší:

1. Aplikace načte identifikátor karty (způsob je zde nepodstatným, viz TODO).
2. Uživatel zadá PIN a potvrdí přihlášení.
3. Aplikace zavolá KIS endpoint `/auth/rfid_token`, identifikátor karty a PIN zasílá jako parametry.
4. Pokud karta existuje a PIN odpovídá, KIS vytvoří autentizační token a zašle jej aplikaci.

2.4 Prostředky autorizace a JWT tokeny

KIS využívá tokeny typu JSON Web Token[10] pro autentizaci i následnou autorizaci požadavků. Použit je hybridní systém rolí a přístupových oprávnění (*scopes*). Každý endpoint vyžadující autorizaci je označen jedním z přístupových oprávnění, kterých je v systému 13 (např. „general_display“, „stock_management“ apod.). Z administrativního pohledu jsou však uživatelům přidělovány role: sympatizující člen, běžný člen, správce skladu apod. V systému jsou definovaná mapování mezi rolemi a přístupovými oprávněními. Při vytváření tokenu je spočtena množina oprávnění podle rolí uživatele a použité přihlašovací metody. Oprávnění jsou poté uložena jako součást identity do těla JWT tokenu.

Protože tokeny vydává i pro autentizaci přijímá výhradně služba KIS backend, mohou být podepsány symetricky tajným klíčem uloženým v konfiguraci. Výhodou také je, že jsou tokeny vcelku malé: obsahují pouze čas expirace, ID uživatele a množinu oprávnění.

[10] `https://datatracker.ietf.org/doc/html/rfc7519`

Endpointy očekávají podepsaný token v hlavičce `Authorize`. V rámci autorizace pouze kontrolují obsah tokenu, vychází tedy pouze z tohoto nosiče identity a neprovádějí další kontrolu např. v databázi.

2.5 Nedostatky způsobu autentizace

Nejvýznamnějším „praktickým“ nedostatkem dosavadního přístupu je velmi omezená možnost integrace KIS backendu s dalšími službami. Je patrné, že tento požadavek při původním návrhu neexistoval, a tak byla služba navržena skutečně jako backend pro webové aplikace, ne jako služba využitelná z jiných služeb. Veškerá autorizace v API počítá s tokeny, které nesou identitu a oprávnění jistého uživatele, a systém v principu neumožňuje autentizaci jiných aplikací.

Prvním rozšířením systému, které se potýkalo s tímto problémem, byl KIS Monitor – jednoduchá aplikace pro zobrazování aktuální nabídky a žebříčku přispěvatelů na obrazovce v klubu. Monitor tedy potřebuje volat příslušné metody API, které však nejsou bez ověření přístupné. Identita monitoru by neměla být vázána na žádného uživatele, v ideálním případě by měla spočívat v unikátním předsdíleném tajném klíči, který je uložen v konfiguraci a není nutné jej měnit.

Jako záplata bylo do backendu přidáno generování *display tokenů*, které mají dlouhou životnost (v řádech měsíců). Systém jim přidělí pouze oprávnění „general_display“. Je zřejmé, že tento přístup není ideální: tokeny není možné zneplatňovat, zároveň je nutné každých několik měsíců vygenerovat nový token a rozdistribuovat jej na zařízení. Token je stále navázaný na identitu uživatele, který jej vytvořil, při jeho úniku by tak bylo například možné změnit uživateli přezdívku nebo PIN kód karty (provedení těchto operací je podmíněno pouze identitou tokenu, ne oprávněními).

Později bylo nutné se stávajícím systémem KIS integrovat novou aplikaci Kachna Online. Zde muselo být použito složité řešení spočívající ve výměně KIS tokenů za vlastní interní tokeny a synchronizace uživatelů při přihlašování.

3 KIS Auth: Služba pro autentizaci a správu identit

Autentizační služba KIS Auth slouží jako centrální bod pro vzájemné ověřování služeb i uživatelů v systému KIS. Implementuje autentizační protokoly OAuth 2.0 a OpenID Connect 1.0 (dále jen *OIDC*). Umožňuje registraci (a přihlašování) prostřednictvím federace identit eduID (s využitím protokolu SAML 2.0), alternativně pomocí Discordu nebo jména a hesla. Poskytuje také alternativní metodu přihlašování pomocí čipových karet s PIN kódem, při které jsou využity vlastní čtečky komunikující pomocí rozhraní WebSocket.

3.1 Použité technologie a knihovny

Služba je implementována v jazyce C# na platformě ASP.NET Core[11] ve verzi 7. Pro perzistenci dat je využit SŘBD PostgreSQL a ORM Entity Framework Core[12] s poskytovatelem Npgsql[13].

Přihlašování do externích služeb zajišťují balíky Sustainsys.Saml2[14] a AspNet.Security.OAuth.Discord[15].

Podporu výše uvedených protokolů zajišťuje knihovna Duende IdentityServer[16]. Jde o komerční produkt,

[11] `https://learn.microsoft.com/aspnet/core/?view=aspnetcore-7.0`
[12] `https://learn.microsoft.com/ef/`
[13] `https://www.npgsql.org/`
[14] `https://github.com/Sustainsys/Saml2`
[15] `https://github.com/aspnet-contrib/AspNet.Security.OAuth.Providers`
[16] `https://duendesoftware.com/products/identityserver`

pro neziskové a malé organizace je poskytována licence zdarma. Využití knihovny při vývoji je možné i bez licenčního klíče.

Dále jsou využity podpůrné knihovny pro hashování (Argon2[17]), logování (Serilog[18]) a mapování objektů (AutoMapper[19]). Všechny dosud uvedené knihovny jsou při překladu automaticky staženy a nainstalovány balíkovacím systémem platformy .NET.

Frontendová část služby využívá vlastní knihovnu pro komunikaci se čtečkami[20], uživatelské rozhraní využívá knihovny Bootstrap[21], jQuery[22] a KioskBoard[23]. Soubory použitých knihoven jsou zahrnuty ve statických datech aplikace.

3.2 Autentizační protokoly

OAuth je protokol, který definuje způsob, jakým může klientská aplikace (*client*) získat token potvrzující identitu svou nebo identitu uživatele (*resource owner*) a využít jej při komunikaci se třetí stranou (např. API, *resource server*). Identitu zprostředkovává *authorization server*. OAuth je v zásadě celkem obecným popisem několika možností, jak si jednotlivé strany mohou vyměňovat přístupové tokeny a předávat tak identitu.

OIDC je autentizační vrstva nad OAuth 2.0, která především definuje principy ověřování a předávání identity **koncových uživatelů** a základních informací o nich, Součástí je také specifikace *discovery* služby, pomocí které mohou klientské aplikace automaticky získat z ověřovacího serveru adresy příslušných endpointů a informace o klíčích použitých pro podepisování tokenů.

Služba KIS Auth podporuje protokol OIDC v prakticky úplném rozsahu specifikací Core, Discovery, Session Management, Front-Channel Logout a Back-Channel Logout.

3.3 Oprávnění

OAuth a OIDC využívají koncept oprávnění (*scopes*), což jsou textové řetězce s především aplikací určenou sémantikou, které vyjadřují právo využít token v určitém kontextu (typicky pří přístupu k některé službě, volání konkrétního API endpointu). Oproti dosavadní verzi KIS, kde oprávnění vycházela z identity uživatele, jsou však oprávnění v OIDC v zásadě nezávislá na uživateli nebo metodě přihlášení, zde jde výhradně o práva udělená klientské aplikaci.

Pokud aplikace získá token s uživatelskou identitou a používá jej k volání endpointů některé služby, tato služba musí sama zajistit autorizaci požadavku ve smyslu určení, zda může danou akci provést daný uživatel. Může k tomu využít další atributy identity (*claims*), které služba KIS Auth zahrnuje ve vydávaných tokenech. V rámci systému KIS se bude využívat autorizace na základě **rolí**: součástí JWT tokenu je tedy kromě atributu *scopes*, který vyjadřuje práva **aplikace**, také atribut *roles*, který vyjadřuje oprávnění udělená (typicky administrátorem) uživateli.

[17] `https://github.com/mheyman/Isopoh.Cryptography.Argon2`
[18] `https://serilog.net/`
[19] `https://automapper.org/`
[20] `https://github.com/su-fit-vut/kis-reader-lib`
[21] `https://getbootstrap.com/`
[22] `https://jquery.com/`
[23] `https://furcan.github.io/KioskBoard/`

3.4 Ověřování identity

Služba povoluje přihlášení pouze uživatelům, kteří ověří svou identitu. Při registraci prostřednictvím eduID je identita potvrzena automaticky, v ostatních případech musí uživatel provést ověření u personálu klubu, typicky s využitím nějakého identifikačního dokladu.

Pokud se uživatel s neověřenou pokusí přihlásit, služba vygeneruje krátký ověřovací kód (*identity confirmation token*), který uživateli zobrazí. Kód je na serveru uložen jako položka cache (potenciálně distribuované) s relativně krátkou platností (dle konfigurace). Uživatel kód ukáže či nadiktuje obsluze baru, která vyplní kód v pokladní aplikaci. Ta zobrazí jméno ověřovaného uživatele a umožní identitu potvrdit. Po potvrzení se uživatel znovu přihlásí.

3.5 Využité identifikátory

Systém jednoznačně identifikuje uživatele vlastními číselnými ID. Zároveň však každý uživatel musí mít přiřazen jednu **nebo více** e-mailových adres, které jsou všechny unikátní, uživatel je tedy jednoznačně identifikován i všemi svými adresami. E-mailové adresy přicházejí typicky z federace eduID nebo jiného externího poskytovatele identity a není výjimkou, že jich je více (např. u studentů FIT, kteří se přihlašují přes eduID a VUT, je k dispozici jak adresa `@stud.fit.vutbr.cz`, tak adresa `@vutbr.cz`). Tento způsob byl zvolen pro maximální možné pokrytí případů, kdy se stejná osoba přihlašuje různými způsoby: shoda libovolného e-mailu s libovolným e-mailem už zaregistrovaného uživatele povede k zabránění nové registrace a umožnění připojit ke stávajícímu účtu nový přihlašovací prostředek.

Externí poskytovatelé vždy připojují také svůj jednoznačný identifikátor uživatele. Tyto jsou použity v interních mechanismech ASP.NET pro párování vlastních uživatelských objektů s příchozími externími identitami.

3.6 Proces přihlášení

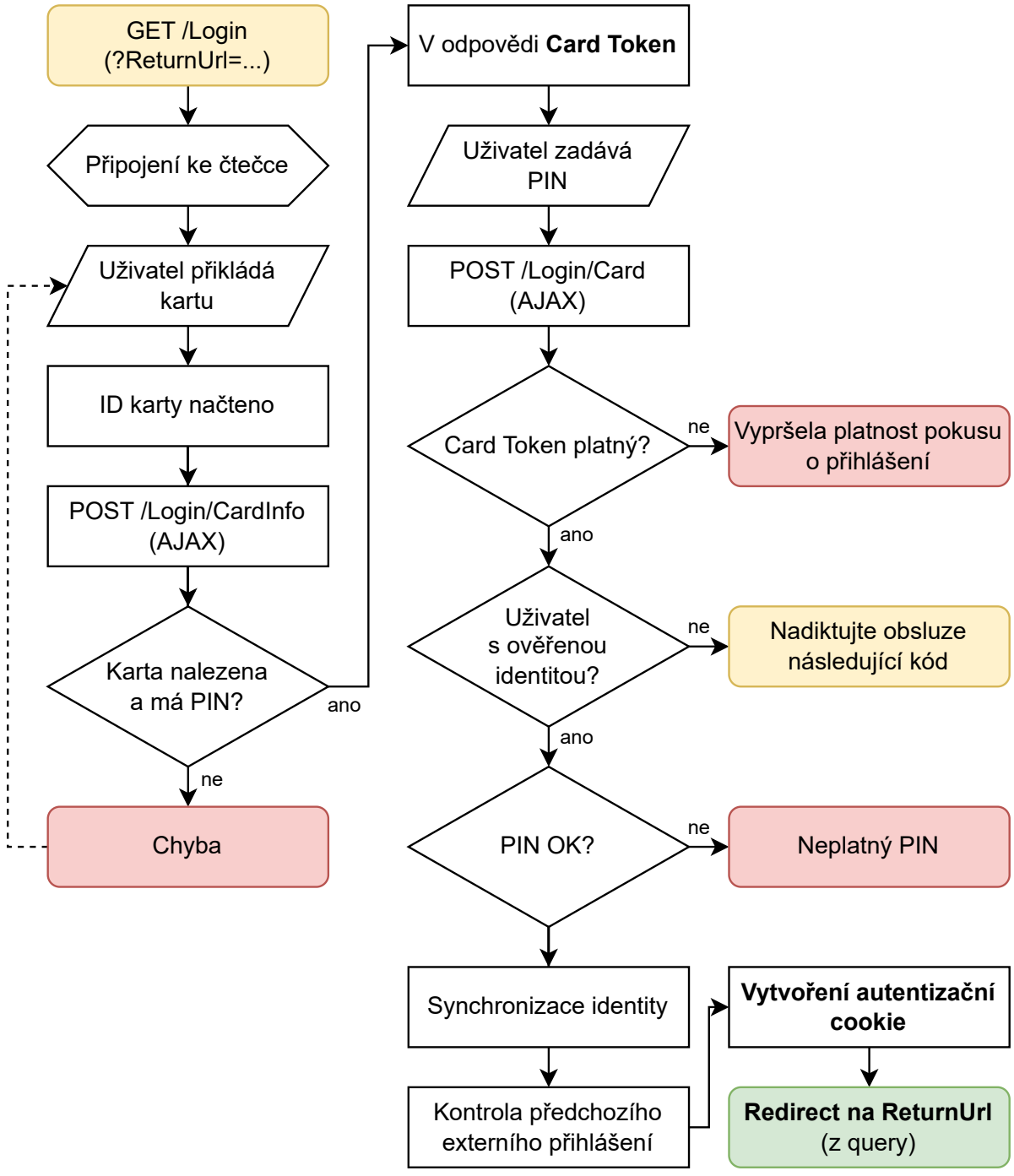
Systém podporuje v zásadě čtyři možné toky přihlášení:

- přihlášení kartou,
- přihlášení přes eduID,
- přihlášení „nedůvěryhodným“ externím poskytovatelem (Discord),
- přihlášení jménem a heslem.

V každém z nich je nutné uvažovat dva způsoby inicializace: buď uživatel sám navštíví adresu portálu (ačkoliv je to nepravděpodobné), nebo je zde přesměrován v rámci příchozího OIDC autentizačního požadavku. V tomto případě je přihlašovací stránce jako query parametr předána návratová adresa, kterou je nutné v rámci celého toku uchovat a uživatele na ni v závěru přesměrovat.

Přihlášení vždy začíná přechodem na stránku `/Login`. Pokud iniciátor nezvolí parametrem `mode` přímý přechod na přihlášení jinou metodou (`eduid`, `discord`, `pass` nebo `join` pro přechod na registrační stránku), zobrazí se stránka, která okamžitě umožňuje přihlášení kartou nebo přechod na jinou metodu přihlášení.

V budoucí verzi by bylo vhodné doplnit systém o možnost „passwordless“ přihlašování, kdy uživateli pro každé přihlášení přijde speciální odkaz na e-mail.



Obrázek 1: Přihlašování kartou.

Přihlášení kartou

Přihlašování kartou probíhá výhradně na stránce `/Login`. Tento proces obnáší také největší část logiky prováděné v prohlížeči a implementované s využitím jazyka JavaScript (skript `wwwroot/js/login.js`). Součástí je totiž komunikace se čtečkami čipových karet vlastního návrhu, které komunikují pomocí technologie WebSocket.

Pokud byla v minulosti zadána adresa čtečky, je uložena v *local storage* prohlížeče a aplikace se ihned pokouší o připojení. Jinak je možné adresu vždy změnit ručně a připojení inicializovat tlačítkem.

Z pohledu logiky má přihlášení kartou dva kroky: výměnu ID karty za *card token*, následné zadání PIN kódu a ověření tokenu i PIN kódu. V případě úspěchu přihlašovací logika ASP.NET vydá vlastní identifikační cookie a logika aplikace přesměrovává na uloženou návratovou adresu. V případě neúspěchu je zobrazeno modální okno, spojení se čtečkou se nepřerušuje. Jedním z možných důvodů neúspěchu je neověřená identita: v takovém případě se rovnou zobrazí i kód pro ověření identity. Celý proces je naznačen v diagramu na obrázku 1.

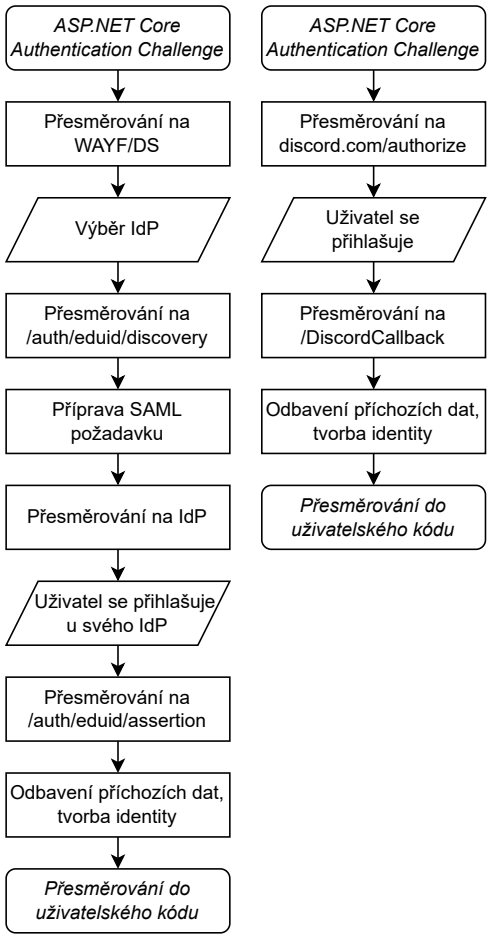
Dvoufázový postup byl zvolen pro lepší uživatelskou odezvu systému: v předchozí verzi se posílalo číslo karty až s PIN kódem, uživatel tedy nemohl rozlišit mezi neplatnou kartou a neplatným PIN kódem. Zde se dozví o neplatnosti karty ještě před zadáváním čísla. Na vydávání card tokenů je aplikována striktní regulace provozu (*rate limiting*), což zabraňuje útoku sekvenčním procházením stavového prostoru karet nebo PIN kódů.

Přihlášení externím poskytovatelem

Při přihlašování externím poskytovatelem identity, kterým v tuto chvíli může být eduID nebo Discord, existuje značné množství stavů, ve kterých se uživatel anebo dodaná identita mohou nacházet. Je třeba rozlišovat:

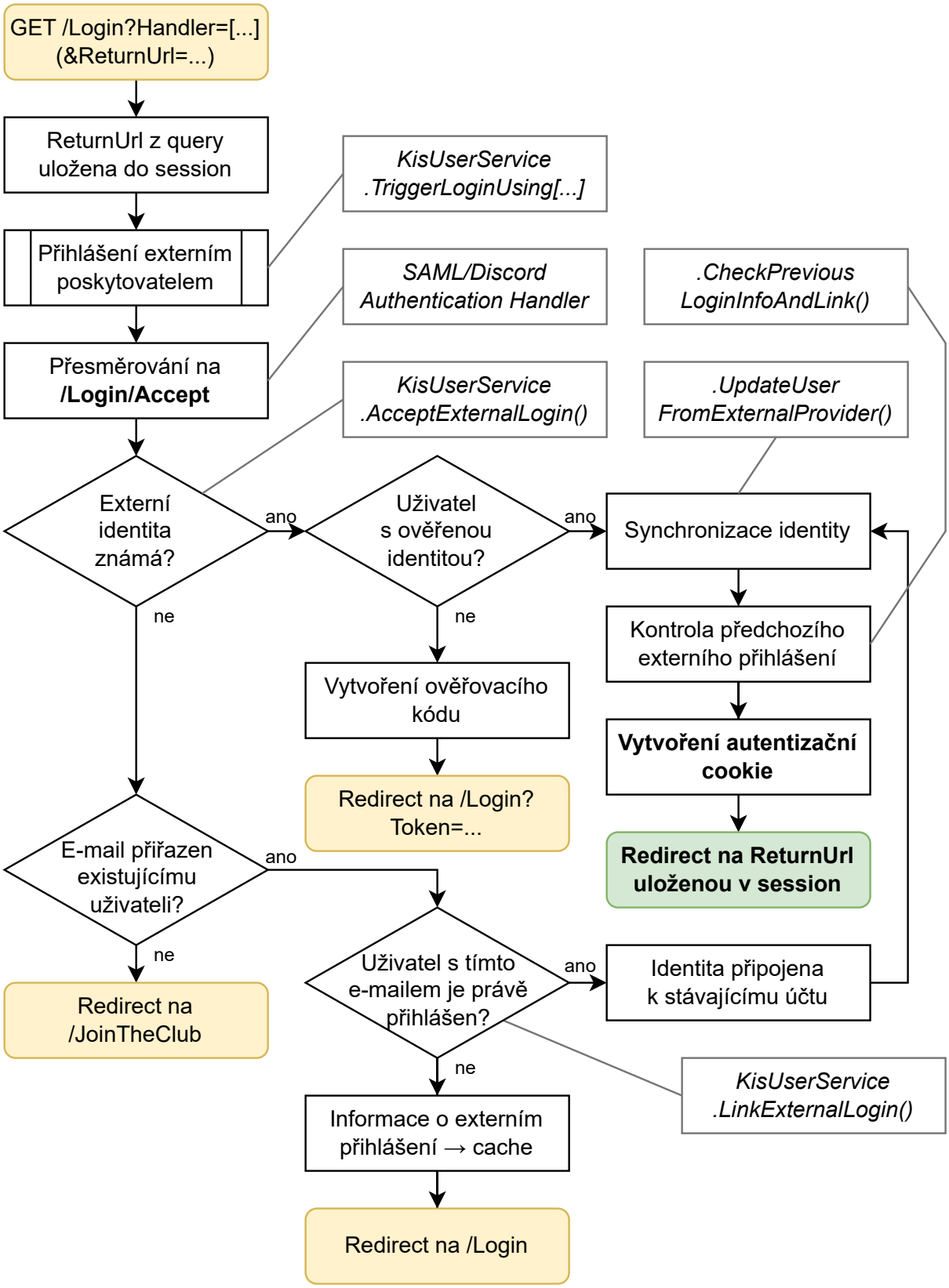
- už známé a dosud neznámé externí identity,
- neznámé externí identity s e-mailem, který už je přiřazen jinému uživateli,
- uživatele s ověřenou a neověřenou identitou,
- externí identity s ověřenou a neověřenou identitou (eduID vs. Discord).

Kompletní pohled na proces poskytuje diagram na obrázku 3. Obrázek 2 pak naznačuje, jak probíhá samotná komunikace s externími poskytovateli – zde je možné sledovat podobnost s procesem popsaným v kap. 2.2.



Obrázek 2: Přihlášení externími poskytovateli z pohledu logiky ASP.NET.

Speciální pozornost bylo nutné věnovat případu, kdy se uživatel přihlásí externím poskytovatelem, toto přihlášení není připojeno k žádnému účtu, ale identita z poskytovatele obsahuje e-mail, který už je nějakému známému uživateli přiřazen. Nejjednodušší možností by bylo prohlásit tyto identity za ekvivalentní a nové externí přihlášení rovnou připojit ke stávajícímu účtu, to však nepovažuji za příliš bezpečné řešení. Zvoleno tedy bylo řešení, kdy se uživatel musí přihlásit



Obrázek 3: Přihlašování externími poskytovateli z pohledu logiky služby KIS Auth. Samotný proces přihlašování je naznačen na obrázku 2. V blocích s poznámkami jsou uvedeny metody třídy `KisUserService`, které primárně obsluhují jednotlivé procesy.

svým stávajícím účtem. Je však pravděpodobné, že toto provede jiným externím přihlášením, proto je nutné původní, zatím „neznámou“ externí identitu uložit a při příštím přihlášení případně spojit s účtem.

Registrace

Registrace je vyvolána primárně jako následek „neznámé“ příchozí externí identity, je však možné vyvolat ji i přímou návštěvou stránky `/JoinTheClub` – v takovém případě jsou zobrazena tlačítka zahajující externí přihlášení. Alternativním způsobem je registrace pomocí e-mailu, pro kterou je vyčleněna speciální stránka `/Register`.

Pokud registrace proběhla na základě identity z Discordu, uživatel musí na stránce s potvrzením doplnit své celé jméno (v případě eduID je jméno vyžádáno v externí identitě). Navíc je poté přesměrován na stránku `/IdentityConfirmation` s ověřovacím kódem. V případě externího přihlášení zůstávají informace o externí identitě uloženy v cookie, takže uživatel nemusí po ověření účtu znovu provádět přihlášení přes Discord. V případě registrace ruční je však nutné opětovné zadání e-mailu a hesla.

Celý registrační proces je naznačen v diagramu na obrázku 4.

3.7 Operace s kartami

Systém umožňuje uživateli přiřazení neomezeného množství identifikačních karet a individuální nastavování PIN kódů. Je navržen s ohledem na možné budoucí změny způsobu identifikace a uložení konkrétní karty, u každé karty se nachází pole označující její typ – v současné době je možná pouze jedna hodnota, `OriginalKisHash`.

Karty se v systému používají ke dvěma účelům, přihlašování (popsáno výše) a identifikace zákazníka při nákupu občerstvení. Pro účel identifikace je aktuálně přidán vlastní rozšiřující typ OIDC požadavku na token, kde je jako parametr zasláno ID karty, odpovědí je token obsahující informace o uživateli.

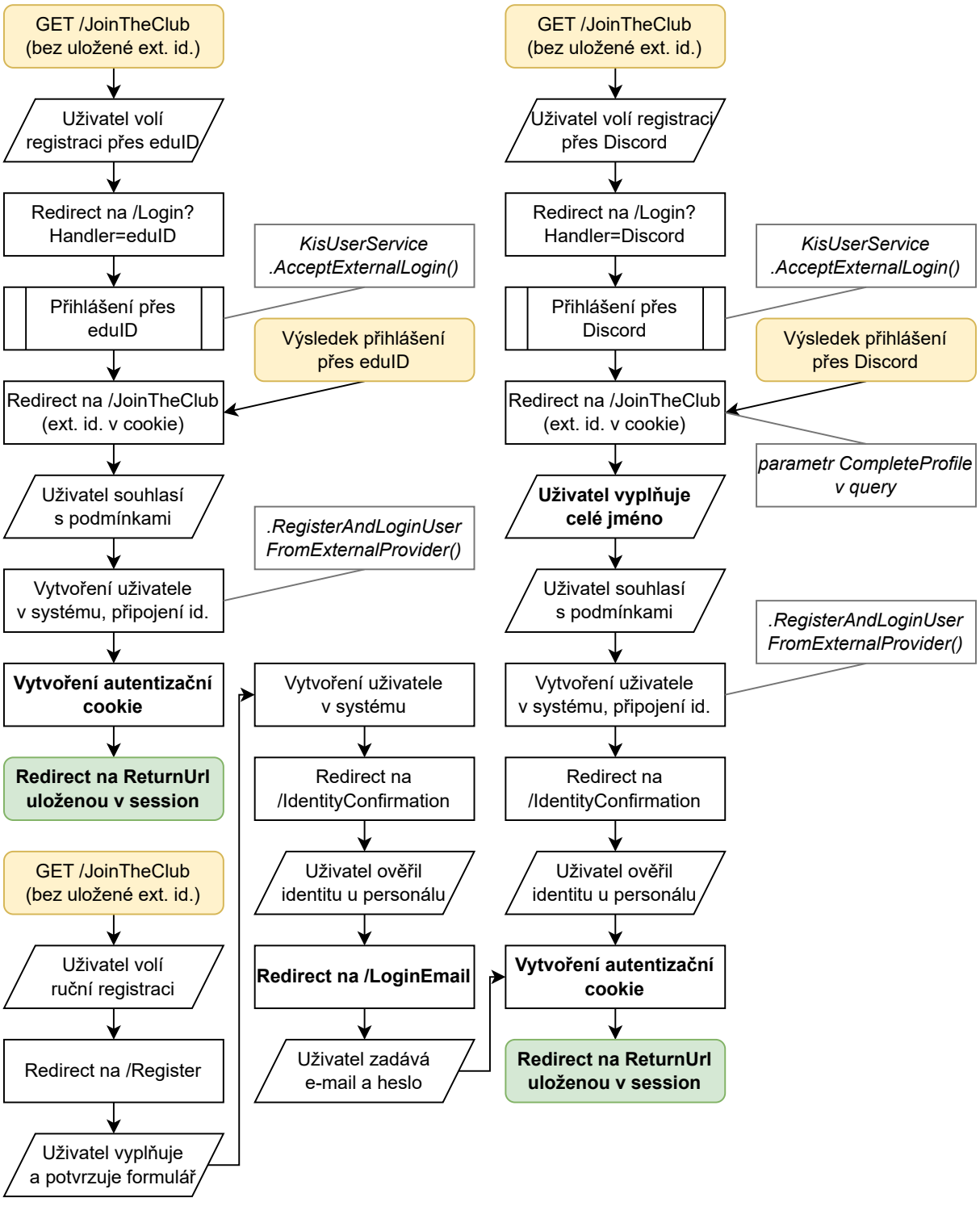
Systém umožňuje jednoduchou migraci stávajících dat: ID karty se hashuje stejným algoritmem, který byl využit v původní implementaci (Argon2). Původní systém však nehashoval PIN kódy, což je považováno za bezpečnostní riziko – při přechodu na novou verzi tedy bude nutné všechny kódy přehashovat.

Párování karty

Pro počáteční přiřazení karty k uživatelskému účtu byl reimplementován proces z dosavadní verze systému. Uživatel nejprve provede registraci běžným způsobem. Po přihlášení do účtu si u pokladny vyžádá párování karty a přiloží kartu ke čtečce. Pokladní aplikace načtené ID vymění voláním `PUT /users/rfids/pool` za *card pairing token*, krátký řetězec, který obsluha sdělí uživateli. Uživatel zadá token do klientské aplikace (např. webové aplikace Kachna Online), která voláním `PUT /users/me/rfid` s tokenem zajistí spárování karty.

Správa karet

Odevzdaná verze systému neposkytuje API pro jiné operace s kartami kromě párování, nicméně v kódu správce uživatelů `KisUserManager` jsou připraveny všechny potřebné obslužné metody: pro nastavování PIN kódů, aktivace a deaktivace karet, přidávání i odstraňování karet. V současnosti nejsou stanovena pravidla pro změny karet, ta budou spolu s potřebnými API operacemi doplněna na základě diskuze s dalšími zainteresovanými osobami před integrací se současnou verzí systému.



Obrázek 4: Registrace (podávání přihlášky ke členství v klubu).

4 KIS Food: Pořadníkový systém

Služba KIS Food je implementací zcela nové komponenty systému KIS, která slouží pro správu front na déle připravované občerstvení, např. toasty, palačinky nebo párky v rohlíku. Jde v zásadě o vyvolávací, resp. pořadníkový systém. Cílem služby je automatizace stávajícího postupu, kdy obsluha u pokladny po objednávce takového občerstvení vypisuje a vydává zákazníkům a osobě připravující (resp. vydávající) občerstvení šatnové bločky s čísly.

Součástí vyvolávacího systému jsou monitory, které zobrazují stav přípravy jednotlivých objednávek a vyhlašují dokončené objednávky pomocí TTS; a zařízení (dále jen *MD – management device*) pro osoby vydávající občerstvení (dále jen *výdej*), které jej využívají pro zadávání stavu přípravy do systému. K tomuto zařízení je připojena pokladní tiskárna, která tiskne lístky pro zákazníka a pro kuchaře (vzhledem k blízkosti těchto osob stačí v případě klubu jedna tiskárna, v běžném provozu by samozřejmě byly využity samostatné tiskárny u pokladny a u výdeje). Systém je samozřejmě propojen s pokladním systémem, který do něj zasílá vytvořené objednávky.

V rámci tohoto projektu byla implementována backendová služba, která řídí stav front a zprostředkovává komunikaci všech ostatních komponent. Další části systému budou implementovány jinými členy SU.

4.1 Použité technologie a knihovny

Služba je, podobně jako KIS Auth (viz kap. 3.1), implementována v jazyce C# na platformě ASP.NET Core ve verzi 7. Pro perzistenci dat je využit SŘBD PostgreSQL a ORM Entity Framework Core s poskytovatelem Npgsql.

Základním stavebním kamenem je ASP.NET Core SignalR[24]. Jde o skupinu knihoven, které poskytují vzdálené volání procedur (RPC) mezi serverem a klienty. Je určená pro aplikace, které vyžadují efektivní komunikaci v reálném čase. Protokol SignalR je dobře zdokumentovaný[25] a pro několik jazyků jsou dostupné stávající klientské knihovny. Tato volba tedy umožňuje snadnou implementaci ostatních komponent systému. SignalR podporuje několik přenosových médií: technologii WebSocket, Server-Sent Events a polling. Knihovny automaticky zvolí nejvhodnější médium podle dostupnosti a kompletně zajišťují plné řízení spojení.

Dále jsou využity pouze drobné podpůrné knihovny pro práci s časovými informacemi (NodaTime[26]), automatickou správu tokenů v rámci KIS Auth (Duende.AccessTokenManagement[27]) a mapování objektů (AutoMapper). Všechny knihovny jsou při překladu automaticky staženy balíkovacím systémem platformy .NET.

4.2 Proces objednávky

1. Zákazník provede objednávku běžným způsobem.
2. Obsluha objednávku běžným způsobem obslouží, je zaslána do pokladního systému KIS Sales.
3. Pokladní systém detekuje produkty s nastaveným příznakem „produkt s pořadníkem“.
4. Pokladní systém zašle objednávku do systému KIS Food pomocí REST API.

[24] `https://dotnet.microsoft.com/apps/aspnet/signalr`
[25] Dokumentace protokolu SignalR: `https://github.com/dotnet/aspnetcore/tree/main/src/SignalR/docs/specs`.
[26] `https://nodatime.org/`
[27] `https://github.com/DuendeSoftware/Duende.AccessTokenManagement`

5. KIS Food pro každý samostatný *typ produktu* v objednávce vygeneruje vyvolávací číslo (v kódu označeno jako *queue item*), které je svázáno s danou objednávkou, osobou, produktem a číslem fronty.
6. Vygenerovaná čísla jsou zaslána v odpovědi systému KIS Sales, který je obratem zasílá do pokladní aplikace (pro případy, kdy není dostupná tiskárna).
7. KIS Food dále zaslílá všem připojeným MD požadavek na tisk čísla. Zároveň zasílá informaci o číslu všem monitorům, které jej mohou zobrazit jako „v přípravě“.
8. MD tiskne pro každé číslo dva lístky:
    - Jeden je určen pro zákazníka, obsahuje číslo a obsah objednávky.
    - Druhý je určen pro výdej, obsahuje číslo, obsah objednávky a **čárový kód** obsahující identifikátor čísla.
9. Jakmile je objednávka připravena k výdeji, výdej naskenuje čtečkou připojenou k MD čárový kód na příslušném lístku.
10. MD zasílá do služby KIS Food informaci o naskenování lístku.
11. KIS Food zasílá všem monitorům informaci o změně stavu čísla na „připraveno k vydání“.
12. Zákazník si u baru přebírá objednávku.
13. Výdej znovu naskenuje čárový kód na svém lístku, čímž potvrdí vyzvednutí.
14. MD zasílá do služby KIS Food informaci o naskenování lístku.
15. KIS Food zasílá všem monitorům informaci o změně stavu čísla na „dokončeno“, monitory odstraňují číslo.

4.3 Vyvolávací čísla

Vyvolávací čísla jsou generována prostřednictvím cyklického čítače (sekvence) v databázi, čítač je tedy jeden globální pro všechny fronty a produkty. Z pohledu UI by mělo být číslo vždy dvouciferné, služba generuje čísla od 0 do 99. Hypoteticky může dojít k situaci, kdy se znovu použije ještě nevyzvednuté číslo, tyto případy jsou však vzhledem k nízké pravděpodobnosti výskytu záměrně zanedbány. Sekvence se nikdy nevrací zpátky, až na případ přechodu z 99 na 0.

Samotné číslo je pouze uživatelský identifikátor. Pro identifikaci vyvolávacího čísla jakožto systémového objektu (tj. při výměně informací o tomto objektu nebo různých pokynech k němu) je využit interní číselný identifikátor, který je vždy rostoucí (omezen je pouze velikostí datového typu).

Vyvolávací číslo jakožto systémový objekt prochází následujícími stavy:

- `InPreparation` – ihned po vytvoření,
- `ReadyToCollect` – po prvním naskenování,
- `Completed` – po druhém naskenování,
- `Cancelled` – při zrušení objednávky nebo konkrétního čísla (různými způsoby).

Z důvodu jednoduchosti se nerozlišuje mezi vytvořenou objednávkou a objednávkou v přípravě. Služba si pro čísla udržuje také příznak „vytisknuto“ – dokud některé MD nepotvrdí vytisknutí lístků, služba se každých několik minut (podle konfigurace) pokouší znovu zadat nevytisknuté lístky k tisku.

Číslo přechází do stavu `Cancelled` pouze v těchto případech:

- zavoláním metody REST API (např. z pokladního systému) pro zrušení konkrétního čísla (`DELETE /qi/[ID čísla]/`) nebo celé objednávky (`DELETE /order/[ID objednávky]`),
- pokud bylo ve stavu `InPreparation` déle než po nakonfigurovanou dobu (24 hodin),
- pokud bylo ve stavu `ReadyToCollect` déle než po nakonfigurovanou dobu (1 hodina).

Při zrušení čísla zasílá služba na MD požadavek na vytisknutí lístku s informací o zrušení čísla.

Typicky číslo prochází stavy postupně a končí ve stavu `Completed`. MD však může implementovat způsob pro navracení lístku do předchozího stavu (například dvěma po sobě jdoucími naskenováními), KIS Food poskytuje pro MD příslušnou metodu pro „krok zpět“. Pokud byl lístek ve stavu `Cancelled`, při jeho „oživení“ se znovu zasílá k tisku. REST API endpoint `PUT /qi/[ID čísla]/state` umožňuje vynuceně převést libovolné číslo do libovolného stavu. Endpoint `POST /qi/[ID čísla]/print` umožňuje vynuceně zadat nový požadavek na vytisknutí libovolného čísla.

Průchod jednotlivými stavy včetně časových informací je zaznamenáván ve speciální tabulce přechodů. V současné verzi implementace se tyto informace v rámci služby nijak nevyužívají, počítá se však s možným využitím pro sestavování statistik o délce přípravy a výdeje občerstvení.

4.4 Fronta

Fronta je systémový objekt, který slouží především pro odlišení jednotlivých skupin produktů při prezentaci (na monitorech). Fronta je identifikována unikátním číselným ID, každá fronta má název. Jediným dalším atributem fronty je kolekce ID produktů, které jsou do této fronty zařazovány. Pokud není tato kolekce definována, fronta se používá jako výchozí možnost pro jinam nezařazené produkty.

K produktu může být v objednávce určena preferovaná fronta. Při příchodu objednávky se vytvářená čísla zařazují do front následujícím způsobem:

1. Pokud je určena preferovaná fronta, produkt se do ní zařadí, pokud je jeho ID v kolekci produktů fronty nebo pokud preferovaná fronta tuto kolekci nedefinuje.
2. Pokud preferovaná fronta není určena, produkt je zařazen do fronty, která obsahuje jeho ID ve své kolekci produktů.
3. Pokud je takových front více, vybrána je fronta s nejnižším ID.
4. Pokud žádná fronta nevyhovuje, produkt je zařazen do fronty, která nemá definovánu kolekci produktů.
5. Pokud je takových front více, vybrána je fronta s nejnižším ID.
6. Pokud žádná taková fronta neexistuje, **celá objednávka končí chybou**.

Fronty jsou zamýšleny jako především statické objekty, nicméně služba poskytuje REST API rozhraní pro vytváření nových front a nastavování seznamu produktů (endpointy začínající `/queue`). Frontu není možné odstranit, ale je možné ji zablokovat definováním prázdné kolekce produktů.

Ve většině případů se budou používat pouze jednotky front pro různé skupiny produktů – např. jedna fronta pro všechny různé varianty toastů, druhá pro různé varianty palačinek. Pokud se vhodně implementují monitory a MD, je však s tímto návrhem možné provozovat v případě potřeby nezávislé výdeje i mimo klub (např. na speciálních akcích v P108).

4.5 Připojování a ověřování zařízení

Ostatní komponenty systému (monitory, MD, pokladní systém) se při komunikaci se službou KIS Food ověřují JWT tokenem vydaným službou KIS Auth. Pro službu KIS Food jsou v systému definována následující oprávnění (*scopes*):

- `kf:r` (KIS Food Read) – umožňuje čtení informací o frontách a číslech v nich,
- `kf:w` (KIS Food Write) – umožňuje vytváření a rušení objednávek, správu čísel a správu front,
- `kfm:e` (KIS Food Monitor Execute) – umožňuje navázat SignalR spojení s uzlem pro monitory a volat jeho metody,
- `kfmd:e` (KIS Food MD Execute) – umožňuje navázat SignalR spojení s uzlem pro MD a volat jeho metody.

Autentizační JWT tokeny vydané službou KIS Auth s některými z těchto oprávnění budou obsahovat v položce `aud` hodnotu `kis-food`.

Služba nijak nepracuje s identitami uživatelů a nekontroluje jejich role. Předpokládá se, že rozhraní služby KIS Food budou využívat pouze jiné služby, ne uživatelské aplikace. V budoucnu bude pravděpodobně doplněna kontrola vhodné role při uživatelském využití „zápisových“ endpointů.

Monitory se pomocí klientské knihovny SignalR připojují na uzel (*hub*) s adresou `/hub/monitor`. MD se pomocí klientské knihovny SignalR připojují na uzel s adresou `/hub/md`. Autentizační token přenáší v HTTP hlavičce `Authorization` nebo v query parametru `access_token`.

4.6 Rozhraní pro zařízení

Monitory mohou využít metody uzlu:

1. `GetQueueInfo(int queueId, bool metadataOnly)` – pro získání informací o jedné frontě,
2. `GetAllQueuesInfo(bool metadataOnly)` – pro získání informací o všech frontách.

Služba očekává, že monitory vhodně implementují obsluhu metod:

1. `InPreparation(QueueItem item)` – při vzniku čísla (nebo jiném přechodu do příslušného stavu),
2. `ReadyToCollect(QueueItem item)` – při přechodu do příslušného stavu,
3. `Completed(int itemId)` – při přechodu do příslušného stavu,
4. `Cancelled(int itemId` – při přechodu do příslušného stavu.

MD využívají metody uzlu:

1. `NotifyScanned(int queueItemId)` – při naskenování čísla,
2. `ReviveItem(int queueItemId)` – pro návrat čísla do předchozího stavu,
3. `NotifyPrinted(int queueItemId)` – po vytisknutí čísla.

Služba očekává, že MD vhodně implementují obsluhu metod:

1. `Print(QueueItemDetails itemDetails)` – pro vytisknutí čísla,
2. `PrintCancelled(QueueItemDetails itemDetails)` – pro vytisknutí informace o zrušení čísla.

Detaily použitých datových struktur a aplikační logiky je možné nalézt ve vygenerované dokumentaci kódu. Informace o REST API je možné zobrazit prostřednictvím Swagger UI na adrese `/swagger/index.html`.

4.7 Konfigurace a spuštění

Před spuštěním je nutné v konfiguračním souboru `appSettings.json` (nebo jiným způsobem, viz dokumentace ASP.NET Core[28]) nastavit alespoň následující položky:

- `ConnectionStrings:FoodDb` – připojovací řetězec pro PostgreSQL databázi,
- `Kis:ClientId` – ID klienta v KIS Auth (KIS Food vystupuje jako klientská aplikace pro získávání informací z KIS Sales),
- `Kis:ClientSecret` – tajný klíč klienta,
- `Kis:KisAuthAuthority` – adresa KIS Auth,
- `Kis:KisApiUrl` – adresa KIS Sales.

V sekci `Queue` ke možné nastavit také časové limity pro rušení čísel a opětovný tisk lístků.

Při prvním spuštění je nutné použít parametr příkazové řádky `--migrate-db`, který vytvoří tabulky v databázi (je možné jej použít při každém spuštění pro zajištění správného schématu databáze při změně verze služby).

[28] `https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-7.0`