Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Evangelos Katsaros

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Evangelos Katsaros

Systémové inženýrství a informatika
Informatika

Název práce

Mobilní aplikace – rybářský lístek

Název anglicky

Mobile application – fishing license

Cíle práce

Cílem je navrhnout mobilní aplikaci elektronického rybářského lístku pomocí wireframe. Aplikace bude naprogramována pomocí technologie react native. Součástí práce bude studie proveditelnosti možného praktického nasazení aplikace Českým rybářským svazem.

Metodika

První část práce bude obsahovat teoretickou rešerši použitých nástrojů a technik. Druhá, praktická část práce bude dodržovat standardy softwarového inženýrství, především modelovací jazyk UML.

Doporučený rozsah práce

30 – 60 stran

Klíčová slova

Mobilní aplikace, wire frame, react native, user interface, android, use case

---

Doporučené zdroje informací

FOWLER, Martin. Destilované UML. Praha: Grada, 2009. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.

Rybářský řád a soupis mimopstruhových revírů pro držitele celosvazových povolenek. Praha: Český rybářský svaz, 2007.

Rybářský řád a soupis pstruhových revírů pro držitele celosvazových povolenek. Praha: Český rybářský svaz, 2007.

ŠUSTA, Marek. Průvodce systémovým myšlením. Praha: Proverbs, c2015. ISBN 978-80-260-7602-5.

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

doc. Ing. Vojtěch Merunka, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 23. 3. 2020

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 23. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 15. 03. 2021

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Rybářský lístek mobilní aplikace" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne _____

Poděkování

Rád bych touto cestou poděkoval Nině Doubkové, Tomáši Kyprému, Bc. Petru Kučerovi, Ing. Michalu Lieskovskému a panu doc. Ing. Vojtěchu Merunkovi, Ph.D. za kontrolu a konzultace ohledně mé bakalářské práce. Dále bych chtěl poděkovat své rodině za vytvoření přívětivého prostředí pro tvorbu mé bakalářské práce.

Rybářský lístek mobilní aplikace

Abstrakt

Práce se zabývá paralelním nasazením nebo úplným nahrazením papírových rybářských licencí za elektronickou verzi pomocí mobilní aplikace.

V práci budou vysvětleny jednotlivé funkcionality pro všechny role, které v tomto systému figurují.

Dále zde bude popis wireframů jednotlivých obrazovek mobilní aplikace a vysvětlení jejich funkčnosti.

Mobilní aplikace bude naprogramována s použitím frameworku React Native, který umožňuje vývoj aplikací jak pro Android, tak pro IOS.

Klíčová slova: Android, iOS, UML, wireframe, rybářský lístek, mobilní aplikace, React Native

Mobile application fishing licence

Abstract

This thesis deals with the parallel deployment or full replacement of a fishing license in paper form for an electronic version using a mobile application.

The thesis will explain the individual functionalities for all roles in this system.

Also, there will be a description of the wireframes of individual screens of the mobile application and an explanation of their functionality.

The mobile application will be programmed using React Native framework which allows development of mobile applications for Android and iOS.

Keywords: Android, iOS, UML, wireframe, fishing license, mobile application, React Native

Obsah

Seznam obrázků

Seznam použitých zkratek

RBAC – Role Base Acess Controll

UML – Unified Modeling Language

CRUD – Create, Read, Update, Delete

WF – Wireframe

SW – Software

JDK – Java Development Kit

SDK – Software Development Kit

ID – Identifikační dokument

URL – Uniform Resource Locator

1 Úvod

Motivací pro tuto práci mi byla hlavně osobní zkušenost se zapomenutými rybářskými papíry v jiné výbavě, než kterou jsem si ten den bral k vodě. V ten den jsem si řekl, proč vlastně v dnešní době neexistuje nějaká mobilní aplikace, která by mi tu licenci nahrazovala. Vždyť přeci v dnešní době se nikdo s výjimkou starší generace bez svého smartphonu nikam nehne. Proto jsem se rozhodl, že se tomuto problému budu věnovat více a výsledky jsem shrnul ve své bakalářské práci.

V první časti mé bakalářské práce popisuji, jak by se tato aplikace dala celá implementovat na jakémkoliv rybářském svazu. Ve své práci popisuji jednotlivé role od administrátora až po konečného uživatele. Dále se v práci zabývám jejich logickým rozvržením pomocí use case diagramů, jejich funkcionalitami a vymyšlením wireframů.
Na závěr tuto aplikaci naprogramuji s pomocí frameworku React Native a zhodnotím její využitelnost v porovnání s papírovou verzí.

1.1 Metodika práce

Práce je rozdělena do dvou části. První část práce je teoretická. Tato část práce obsahuje seznámení s potřebnými technologiemi pro praktickou část. V teorii bude obsaženo UML, pravidla pro tvorbu WF a vysvětlení frameworku React Native společně s nejpoužívanějšími komponenty.

Praktická část se skládá z návrhu implementace mobilní aplikace rybářským svazem pomocí UML diagramů. Dále jsou v této časti navrženy WF a popsány části zdrojového kódu mobilní aplikace Rybářský lístek.

1.2 Cíle práce

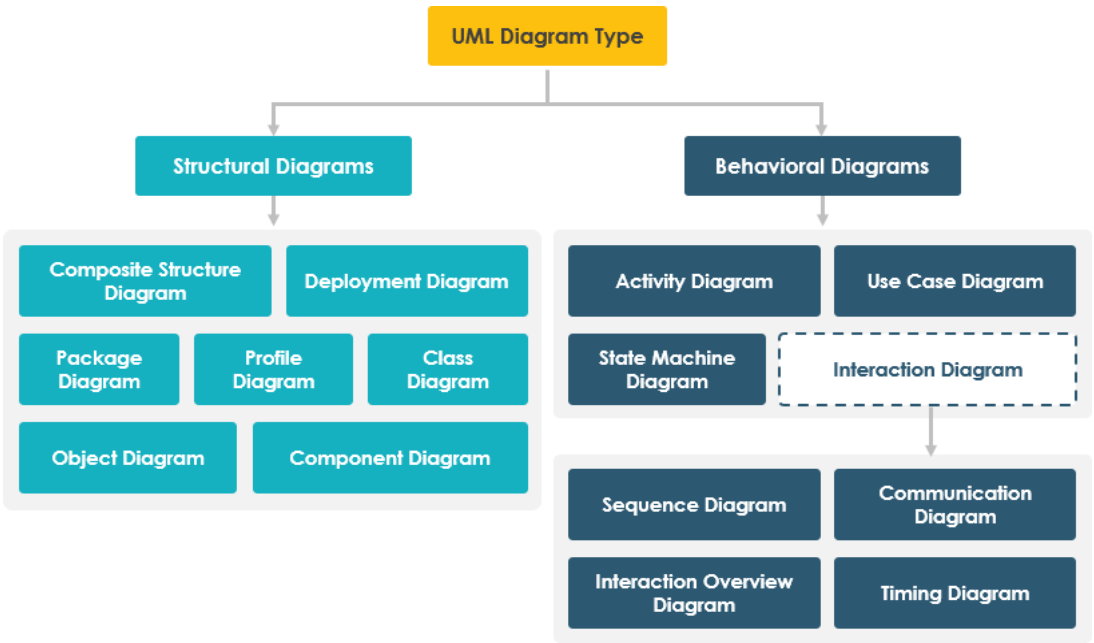
Prvním cílem je vymyšlení možnosti nasazení mobilní aplikace rybářský lístek do užívání rybářským svazem. Druhým cílem je vytvoření logického designu této aplikace, pomocí kterého následně tuto aplikaci naprogramuji. Součástí práce je také samotná funkční aplikace, která je naprogramovaná pomocí frameworku React Native.

2 Role z pohledu informačního systému

Co jsou role rozvádí článek [16]. Role z pohledu informačního systému umožňují definovat subjekty a jejich práva. Této metodě se říká RBAC. Je-li v systému více uživatelů, kteří do jeho chodu nějak zasahují, je dobré jim vymezit oprávnění, protože by bylo nevhodné, aby všechny role měly přístup k některým citlivým informacím nebo měly třeba právo na vytvoření nových uživatelů. Proto je dobré při tvorbě některých aplikací si tyto role nadefinovat a poté jim přiřadit jejich práva. Role se dají dobře definovat pomocí UML diagramu, a to konkrétně pomocí Use case diagramu. Pomocí Use case diagramu budu definovat role v mé bakalářské práci.

2.1 Pravidla pro tvorbu use case diagramů

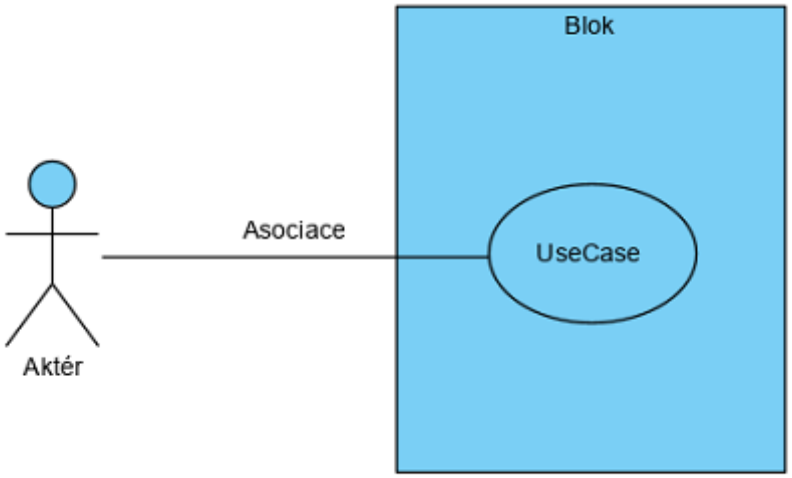
Use case diagramy jsou nástrojem UML a konkrétně patří do behaviorálních diagramů. UML diagramy se dělí do dvou základních kategorií, a to do strukturálních a behaviorálních diagramů. Rozdíl mezi strukturálními a behaviorálními je v tom, co pomocí nich popisujeme. Strukturované diagramy ukazují věci, které jsou v modelovaném systému. Popisují například třídy, objekty nebo komponenty které systém tvoří. Zatímco behaviorální diagramy se používají pro popis chování daného systému.



Obrázek 1: Rozdělení UML diagramů [22]

Vybral jsem si Use case diagramy, protože jsou vyhovující právě pro tvorbu takzvaných Business rolí, a právě pomocí těchto rolí lze zobrazit, jak by mohlo fungovat nasazení této mobilní aplikace [11][4][1]. Use case diagramy jsou tvořeny aktéry, bloky use case, relacemi,

a bloky. Aktéři jsou role, které nějakým způsobem interagují s „use case“ v konkrétním systému. Use case jsou funkcionality systému a jsou pomocí asociací propojeny s aktéry. Blok je systém, který obsahuje „use case“.



Obrázek 2: Základní prvky use case diagramu

3 Wireframe

Wireframe (dále už jen WF), nebo-li drátěný model, který popisuje vizuální a logický vzhled webu nebo aplikace. Pomocí WF je dobré si navrhnout vizuální vzhled každé obrazovky aplikace, jako například rozložení tlačítek, formulářů nebo zobrazení nějakého obsahu. WF by neměl být pouze o vhledu z grafického pohledu, ale hlavně o funkčnosti a provázanosti jednotlivých obrazovek aplikace. WF je tvořen pouze čárami a textem bez obrázku a barev.

Další věcí, s kterou by nám WF měl pomoci, alespoň podle [5] je optimalizováním množství stránek aplikace. Toto je myšleno, že pro orientaci a snadnější přehlednost v aplikaci nám pomáhá i menší množství stránek aplikace.

Jsou dva způsoby tvorby WF. První a velice efektivní způsob, je tvorba WF náčrtkem rukou na papír nebo tabuli. Vůbec nevadí, že náčrtek je nesouměrný a nepřesný. Má sloužit jen pro vizualizaci myšlenek. Jeho hlavní výhodou je rychlost vytvoření.

Druhým způsobem je tvorba WF pomocí nějakého SW. Tento způsob se vyplatí při tvorbě rozsáhlejších webů nebo aplikací. Umožňuje nám dělat přesnější a detailnější návrhy, také nám umožňuje snadné předělávání na rozdíl od první popisované metody.

4 React Native

React Native je open-source framework pro tvorbu mobilních aplikací vytvořený společností Facebook v roce 2015[9]. Kromě této společnosti se na vývoji tohoto frameworku podílí také komunita [12]. Pomocí React Native lze programovat nativní mobilní aplikace jak pro zařízení s operačním systémem Android, tak i s iOS. React Native používá ReactJS pro vytváření uživatelského rozhraní. ReactJS je JavaScriptová knihovna vytvořená stejnou společností, kterou je vytvořen React Native s tím rozdílem, že ReactJS je vytvořen pro vytváření webových aplikací.

Komponenta je základní stavební prvek Reactu, pomocí kterého se vytváří uživatelské rozhraní. V praxi se používají komponenty vlastní a komponenty třetích stran obsažené v knihovnách. Komponenty se velmi podobají HTML tagům.

Mezi příklady nejpopulárnější mobilních aplikací používajících React Native patří například Facebook, Instagram, Skype, Uber Eats nebo také Discord podle zdroje [19].

4.1 Základní komponenty React Native

Knihovna React Native poskytuje řadu předpřipravených komponent. Přehled těchto komponent lze nalézt na stránkách [21]. Navíc tato knihovna je stále obohacována tisícími dalších vývojářů, kteří přispívají k tvorbě nových komponent. Mezi základní komponenty patří View, Button, Image, Flatlist, TextInput a WebView. Tyto základní komponenty byly využity i při tvorbě této bakalářské práce, a proto budou jejich funkcionality vysvětleny.

4.1.1 Komponenta View

Nejzákladnější komponentou pro vytváření uživatelského rozhraní je komponenta View [17]. Tato komponenta je kontejner, který podporuje stylování a ošetření dotyků. Ekvivalentem této komponenty je v HTML párový tag .

4.1.2 Komponenta Button

Button je základní komponenta, která by se měla dobře vykreslovat na jakémkoliv zařízení. Její nevýhodou je minimální možnost stylování [6]. V dokumentaci této komponenty lze z toho důvodu nalézt komponentu TouchableOpacity [15]. Tuto komponentu lze na rozdíl od Button komponenty daleko lépe stylovat a je dokonce doporučená pro tvorbu vlastních tlačítek.

4.1.3 **Komponenta Image**

Image je komponenta, která umožňuje zobrazování obrázků [7]. Obrázky v této komponentě mohou být z různých zdrojů. Komponenta obsahuje atribut source, pomocí kterého se zadává cesta k obrázku. Dále komponenta obsahuje stylování.

4.1.4 **Komponenta FlatList**

FlatList je komponenta, která má široké využití. Podporuje tvorbu například záhlaví, zápatí nebo posouvatelného obsahu [10]. Kompletní výčet funkcionalit lze opět nalézt v dokumentaci React Native.

4.1.5 **Komponenta TextInput**

TextInput je základní komponenta pro zadávání textu do aplikace pomocí klávesnice [14]. Umožňuje nastavení několika funkcí. Mezi tyto funkce patří například automatické opravování napsaného textu, vložení placeholderu nebo otevírání různých typů klávesnic. Komponenta je ekvivalentní k HTML tagu <form>.

4.1.6 **Komponenta WebView**

WebView je komponenta, která umožňuje zobrazování webového obsahu načítaného přes dané URL [18].

.

5 Definice rolí pro možnost nasazení

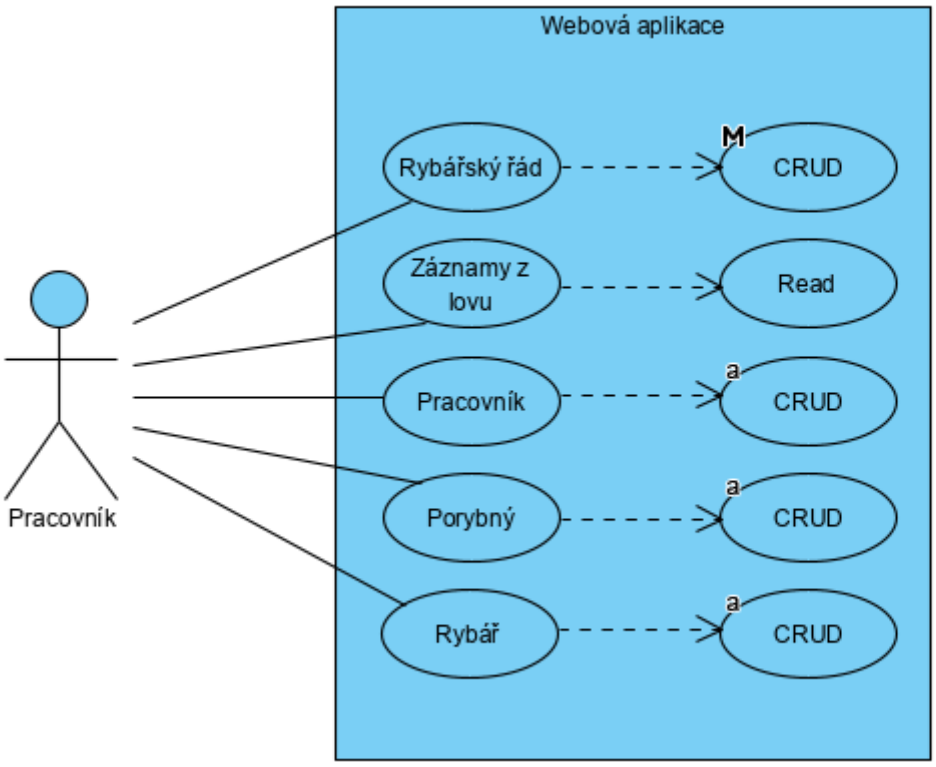
První rolí je pracovník, pod ním si můžeme představit například úředníka s oprávněním vydávat rybářské povolenky, předsedu rybářského svazu nebo jakéhokoliv pracovníka rybářského svazu, který potřebuje mít přístup k záznamům z lovu pro další zpracování.

Druhou důležitou rolí je porybný – vykonavatel rybářského práva, který má za úkol kontrovat, zda rybáři svojí činností neporušují rybářský řád.

Třetí rolí pro tuto bakalářskou práci je samotný rybář, pro kterého v této bakalářské práci naprogramuji mobilní aplikaci.

5.1 Role pracovník

Pracovník je role, která může dělat totéž, co běžný úředník nebo předseda rybářského svazu. Tato role spravuje všechny ostatní role a jejich uživatele, vytváří nové členy nebo odebírá stávající. Dále má tato role právo na úpravu rybářského řádu a čtení databáze lovů, kterou potřebuje ke zpracovávání dat a následnému vytvoření plánu zarybňování. Ke všem těmto funkcionalitám má přístup například přes webovou aplikaci, která není předmětem této bakalářské práce, ale je potřeba jí vysvětlit z důvodu možného nasazení této aplikace.



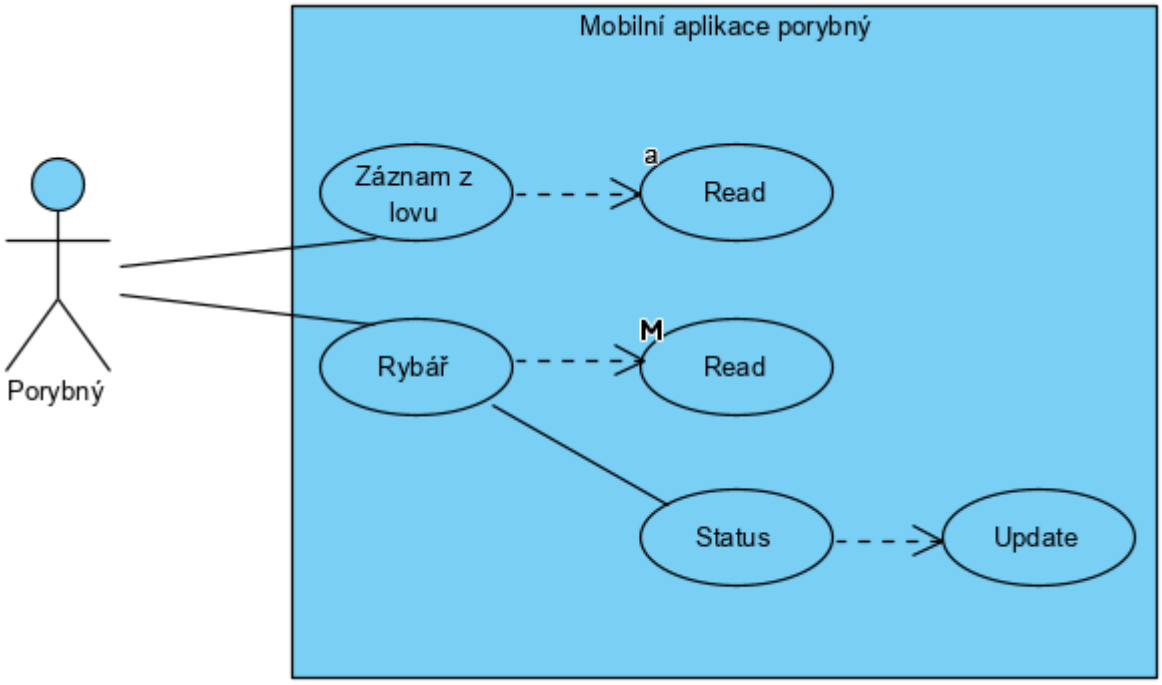
Obrázek 3: Use case diagram pro pracovníka

Role pracovník by tedy podle use case diagramu mohla, jakkoliv upravovat rybářský řád, protože nad tím to dokumentem by měla povolené všechny operace CRUD. Dále by mohla pouze číst záznamy z lovu, které by dále zpracovávala. Mohla by vytvářet roli porybný a různě jí upravovat, třeba u ní měnit kontaktní údaje atd. A nakonec by mohla vytvářet roli rybář a také jí různě upravovat, aktualizovat, případně vymazat.

Při definování této role jsem se inspiroval vlastní zkušeností a také obsahem dokumentu [2][3].

5.2 Role porybný

Roli porybný by stačily pouze dvě funkcionality. Za prvé, aby mohl zjistit, jestli má rybář zapsanou docházku v záznamu lovů. Za druhé, aby v případě nějakého závažného porušení mohl rybáři odebrat povolenku k lovu. Odebrání povolenky k lovu by se mohlo v aplikaci povést změněním statusu u rybáře. K těmto funkcím by měl přístup přes vlastní mobilní aplikaci Porybný. Tato aplikace taktéž není součástí této bakalářské práce.

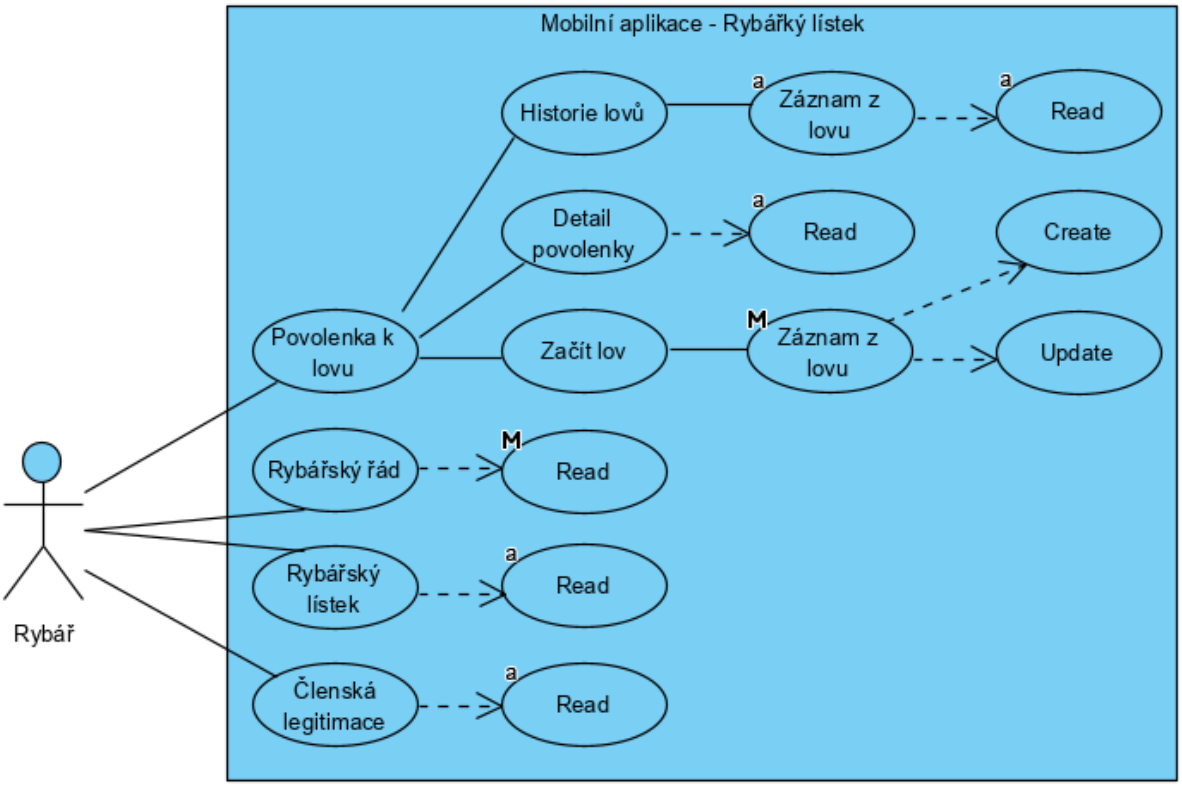


Obrázek 4: Use case diagram role porybný

Porybný tedy u vody při kontrole zadá jen rybářské ID, které by si zjistil z mobilní aplikace rybářský lístek od rybáře a viděl by v záznamu z lovu, zda má rybář ten den vyplněnou docházku v mobilní aplikaci. Do budoucna bych pro tuto kontrolu navrhnul skenování QR kódu místo zadávání rybářského ID, který by si porybný v aplikaci načetl ze zařízení rybáře. Dále má porybný právo na čtení z databáze Rybář, kde by viděl jméno, rok narození a také status, který rybář aktuálně má. Status bych rozdělil na aktivní a na neaktivní.

5.3 Role rybář

Role rybář má právo pouze na čtení historie lovů, rybářského řádu, detailu povolenky, rybářského lístku a členské legitimace. Na těchto dokumentech by se rybářovi zobrazovaly stejné informace, jako jsou na papírových verzích stejnojmenných dokumentů, které rybář musí sebou nosit k vodě. Co se týče zahájení lovu, tak by rybář jako jediná role směl vytvářet záznamy v databázi záznamy lovů.



Obrázek 5: Use case diagram role rybář

V praxi by to mělo vypadat tak, že rybář bude hlavně využívat povolenku k lovu, proto aby mohl vytvořit záznam z lovu. Během lovu může pomocí update připsat úlovky, které si ponechává. Obdobně jako se současně vyplňuje papírová verze povolenky k lovu. Dále by mohl využíval rybářský řad, například pro vyhledávání čísel revírů, povolené denní doby lovů, nebo lovné míry ryb. Rybářský lístek a členskou legitimaci rybář potřebuje v případě kontroly porybným.

6 Tvorba WF

V následujících kapitolách budou popsaný obrazovky jednotlivých WF mobilní aplikace rybářský lístek. Tvorba těchto WF byla inspirována aktuálními dokumenty.

6.1 WF přihlášení

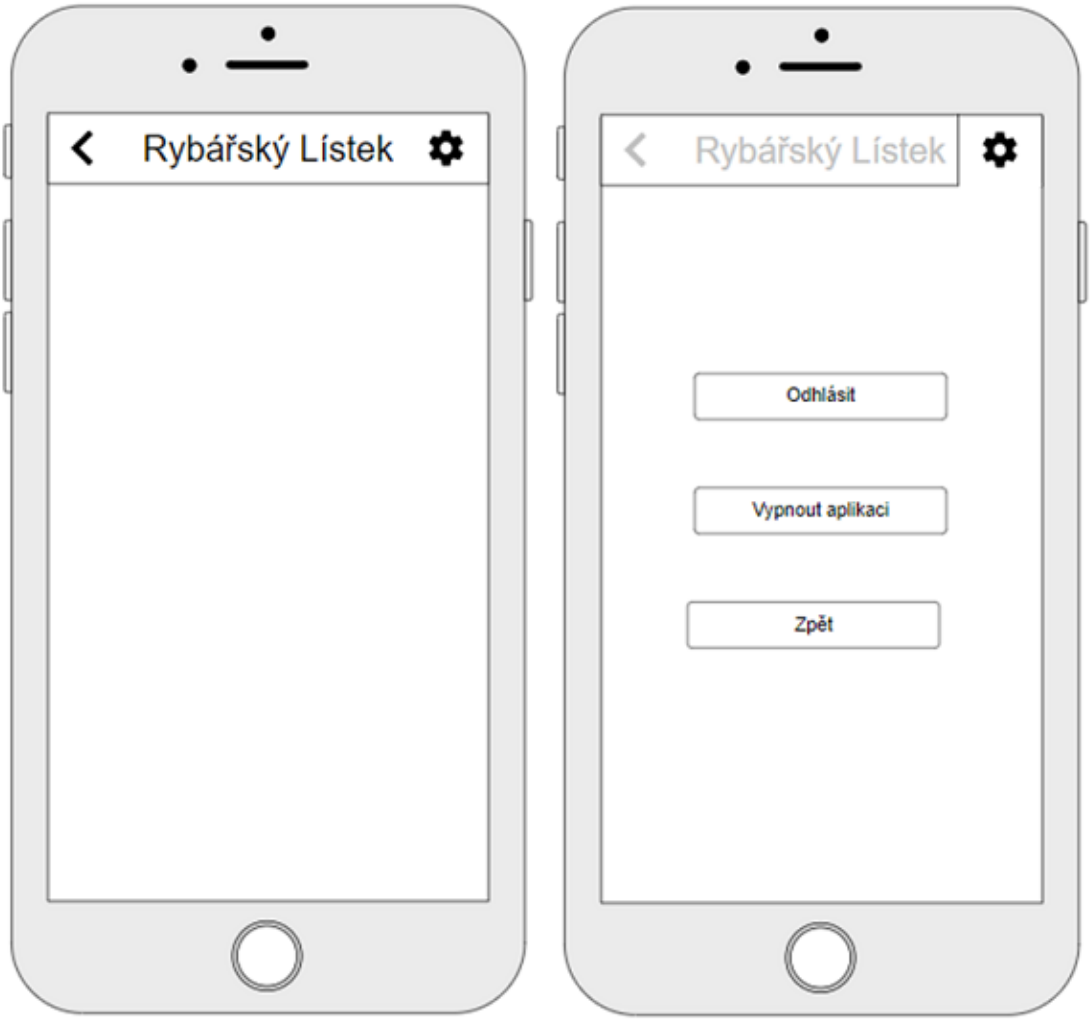
Na následujícím WF ukážu, jak by měla vypadat úvodní obrazovka pro přihlášení. Rybář by se měl přihlásit pomocí rybářského ID a hesla, které dostane na místní organizaci rybářského svazu, pod kterou je přihlášený.

Obrázek 6: WF přihlašovací obrazovky

Na tomto WF jsou dva formuláře a jedno tlačítko. Do prvního formuláře zadá uživatel ID a do druhého heslo. Po vyplnění těchto dvou formulářů klikne uživatel na tlačítko přihlásit. V případě správného vyplnění ID i hesla se uživateli zobrazí základní menu se základním uživatelským rozhraním.

6.2 WF základního uživatelského rozhraní

Základní uživatelské rozhraní a funkce pro pohyb v aplikaci budou umístěny v horní části obrazovky. Toto základní uživatelské rozhraní bude obsahovat tlačítko zpět a tlačítko nastavení.

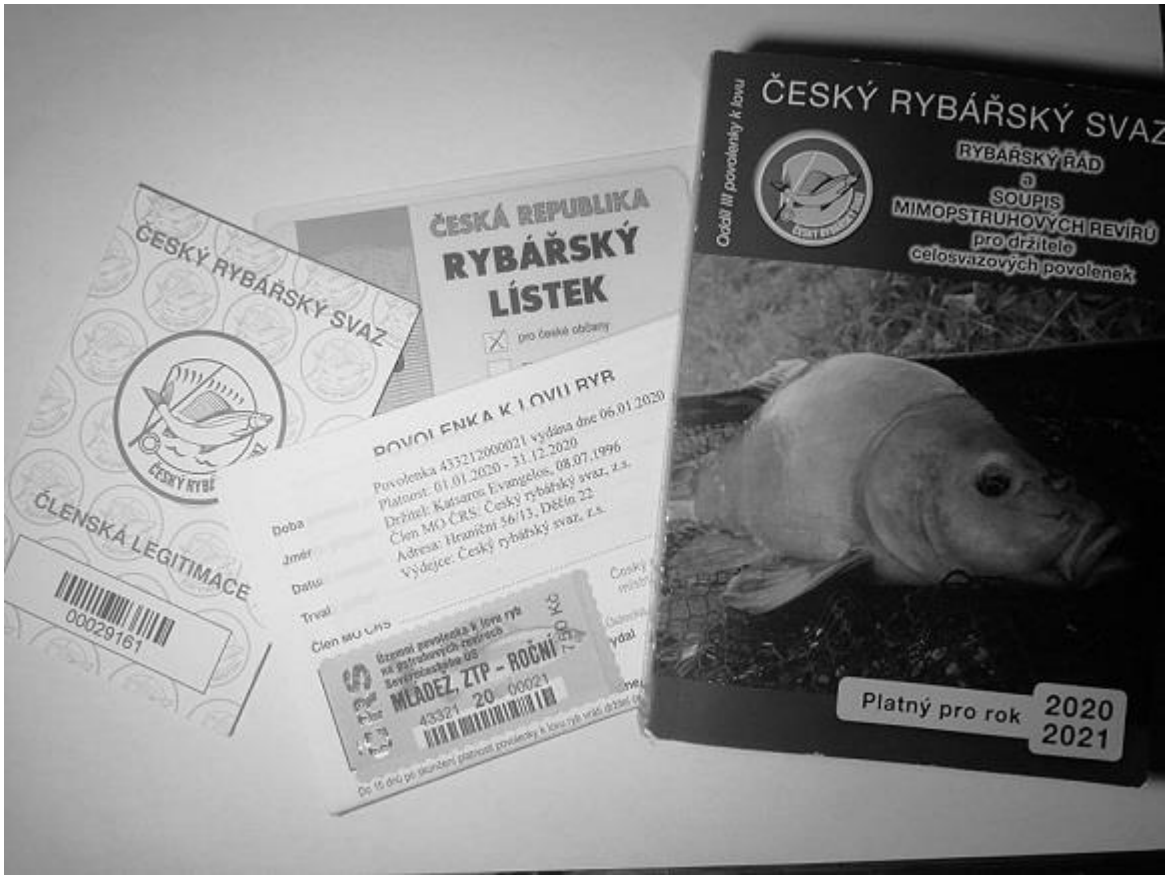


Obrázek 7: Základní uživatelské rozhraní a proklik nastavení

Tlačítko zpět bude umístěno v horním levém rohu obrazovky a bude reprezentováno zpětnou šipkou. Toto tlačítko vrátí uživatele vždy na předchozí obrazovku. Tlačítko nastavení bude v horním pravém rohu obrazovky. Po rozkliknutí tlačítka nastavení reprezentovaného ozubeným kolečkem se zobrazí tři možnosti: odhlásit, vypnout aplikaci a zpět. Možnost odhlásit způsobí odhlášení uživatele. Druhé tlačítko vypnout aplikaci ukončí chod aplikace. A tlačítko zpět zavírá menu s nastavením.

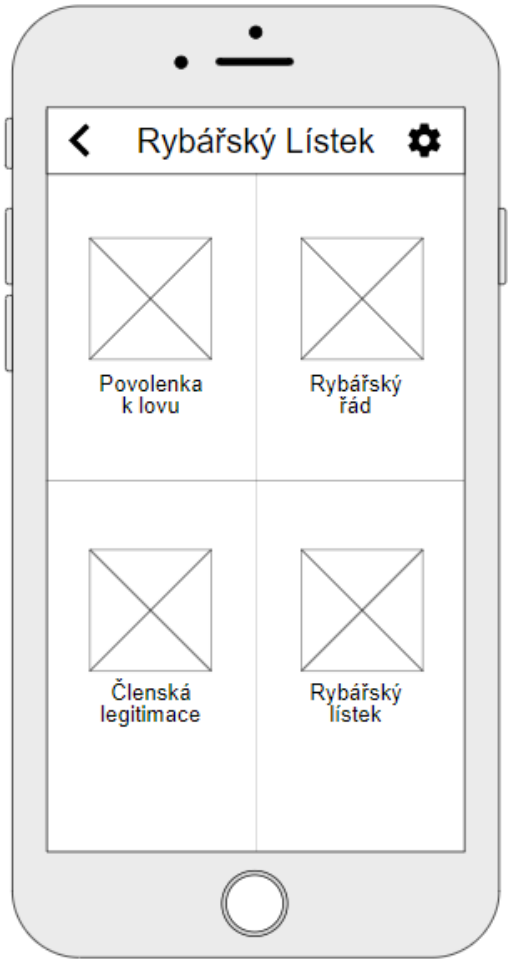
6.3 WF základní menu

Základní menu je to, co se uživateli zobrazí po přihlášení do aplikace. Toto menu bude obsahovat jednoduchou orientaci mezi základními funkcemi, které uživatel bude potřebovat. Menu bude rozděleno do čtyř kvadrantů a v každém kvadrantu bude popisek a ikonka k dané funkcionalitě nebo dokumentu, který se za daným tlačítkem bude skrývat. Tyto čtyři kvadranty boudu kopírovat názvy papírových dokumentů, které tato celá aplikace má nahrazovat.



Obrázek 8: Nynější dokumenty k rybolovu

Na tomto obrázku jsou vyfoceny aktuální dokumenty, které rybář musí mít stále u sebe. Tyto dokumenty obsahují členskou legitimaci, povolenku k lovu, rybářský lístek a rybářský řád. Na následujícím obrázku WF je logické zobrazení pro základní orientaci mezi těmito dokumenty.

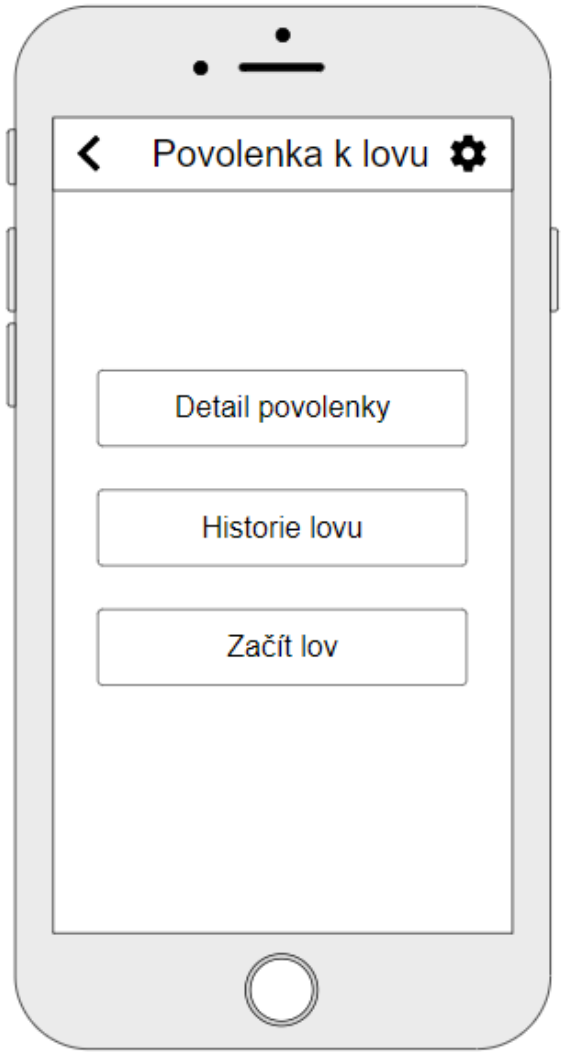


Obrázek 9: Základní menu

Na tomto WF vidíme rozmístění těchto čtyř dokumentů do čtyř kvadrantů. V levém horním kvadrantu se nachází povolenka k lovu. Další nejvyužívanější dokument bude pro uživatele rybářský řád. Rybářský řád by uživatel využíval například pro zjištění čísla revíru nebo lovné míry ryb. Co se týče členské legitimace a rybářského lístku, toto jsou pouze dokumenty, které sám uživatel aplikace nevyužívá, ale jsou potřeba při kontrole porybným.

6.4 WF povolenka k lovu

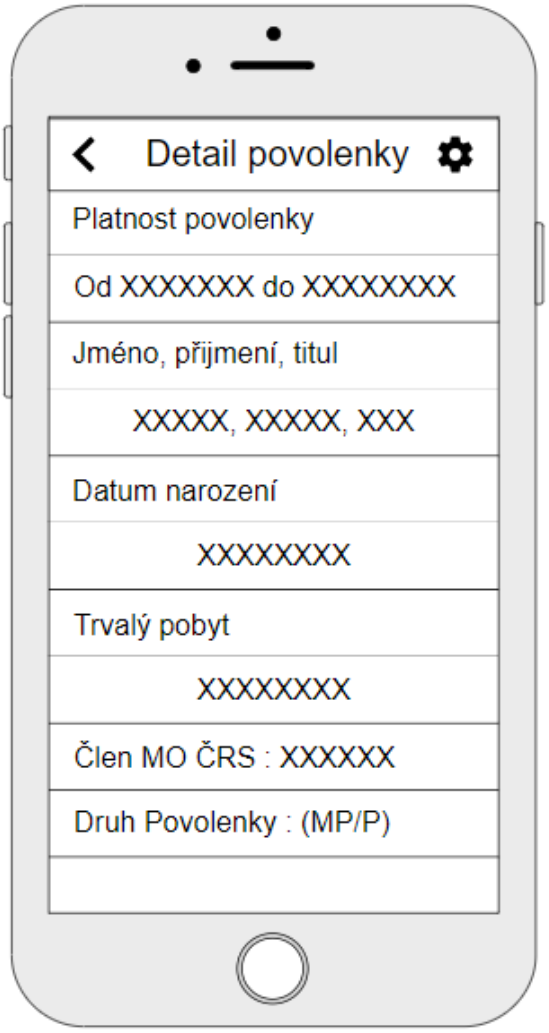
Povolenka k lovu je rozsáhlejší dokument. Jelikož papírová verze obsahuje informace typu platnost povolenky, druh povolenky atd., první sekce se bude jmenovat detail povolenky a bude obsahovat informace o povolence. Druhá sekce této povolenky slouží k zapisování rybářovy docházky. A jelikož při zapisování docházky rybář může vidět i předchozí záznamy z lovů, tak jsem se rozhodl druhou sekci rozdělit na dvě funkce. Tyto funkce si nazveme historie lovu a začít lov.



Obrázek 10: Povolenka k lovu

6.4.1 **WF detail povolenky**

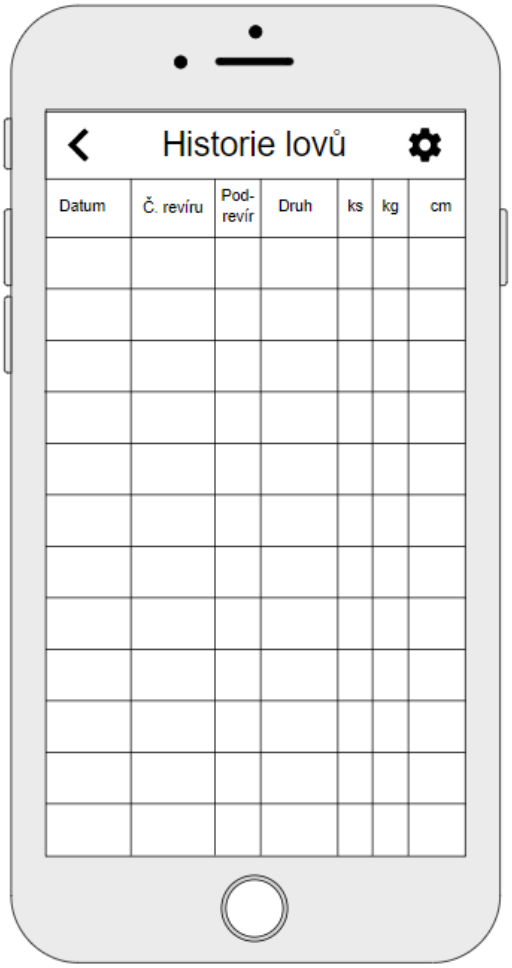
Detail povolenky obsahuje všechny informace, které obsahuje také papírová verze povolenky. Při vytváření tohoto logického designu jsem se co nejvíce držel reálné předlohy. Budeme zde zobrazovat platnost povolenky, jméno uživatele, trvalý pobyt, název organizace, pod kterou uživatel spadá a druh povolenky. Povolenky jsou u nás pstruhové a mimopstruhové. Logický design detailu povolenky k lovu by vypadal takto.



Obrázek 11: Detail povolenky

6.4.2 **WF historie lovů**

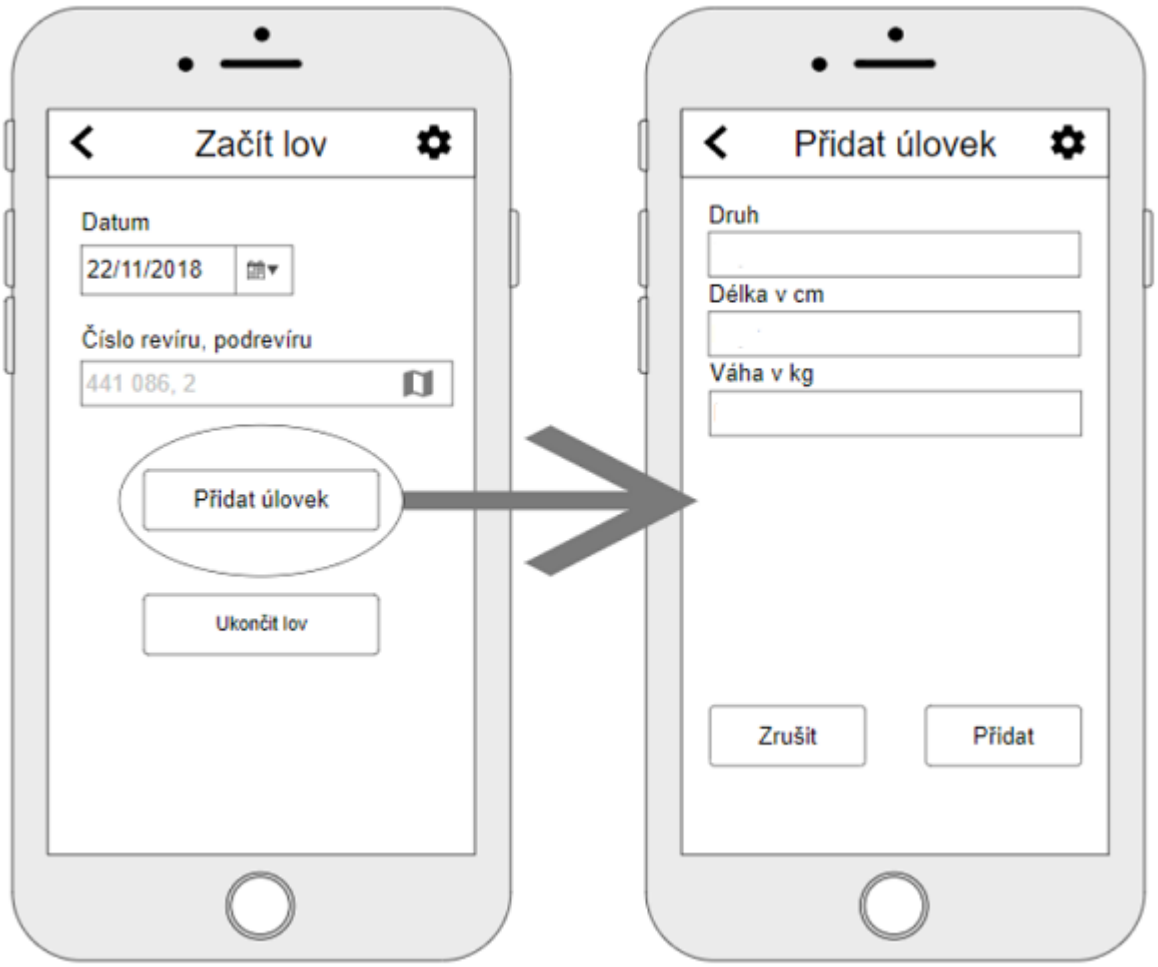
V historii lovů se budou zobrazovat již absolvované lovy uživatele. Konkrétně se zde v tabulce bude zobrazovat datum lovu, číslo revíru, případně číslo podrevírů, druh ulovené ryby, počet úlovků, váha a délka. Tyto údaje by se načítaly přímo z databáze historie lovů a měly by se zobrazovat pouze ty záznamy, u kterých je uživatelovo ID. Logický design historie lovu by vypadal následovně.



Obrázek 12: Historie lovů

6.4.3 **WF začít lov**

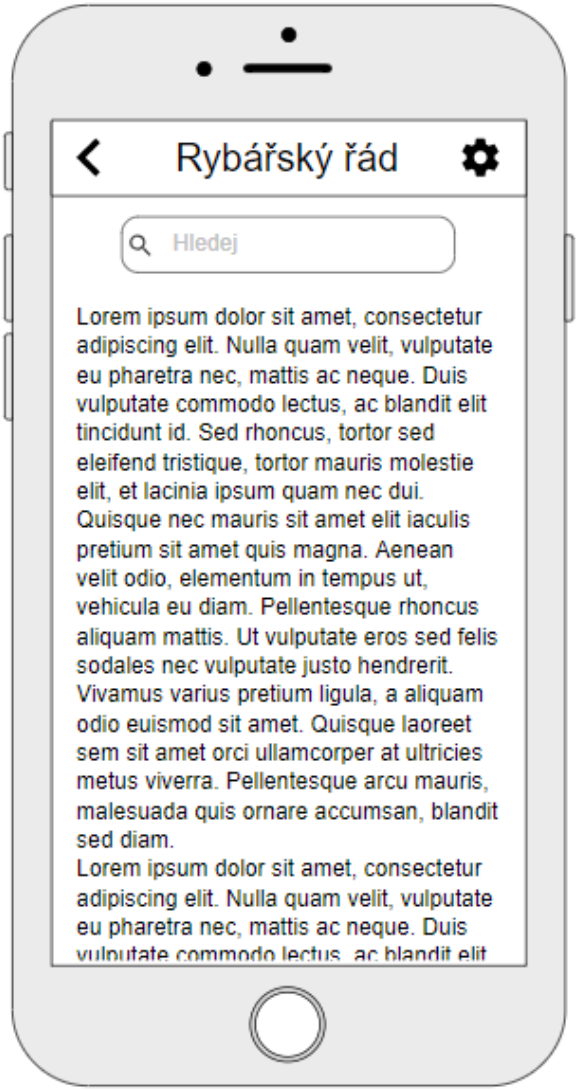
Sekci začít lov jsem navrhl tak, že při zahájení lovu uživatel musí vždy nejdříve napsat datum a číslo revíru stejně jako tomu je ve skutečné povolence. Poté, když si bude chtít ponechat úlovek, klikne na tlačítko přidat úlovek. Po kliknutí na toto tlačítko se zobrazí formulář, kde bude moct uživatel přidat druh, váhu a délku ryby. V případě, že uživatel nevyplní váhu, se do databáze doplní váha ze statisticky naměřených údajů jednotlivých druhů ryb. Další ošetřující funkcionalitou by mohlo být, že si chce uživatel nechat podměrečnou rybu nebo rybu, která je v období hájení. V tomto případě by uživatel mohl být upozorněn třeba nějakým vyskakujícím oknem a daným upozorněním.



Obrázek 13: Začít lov a přidat úlovek

6.5 WF rybářský řád

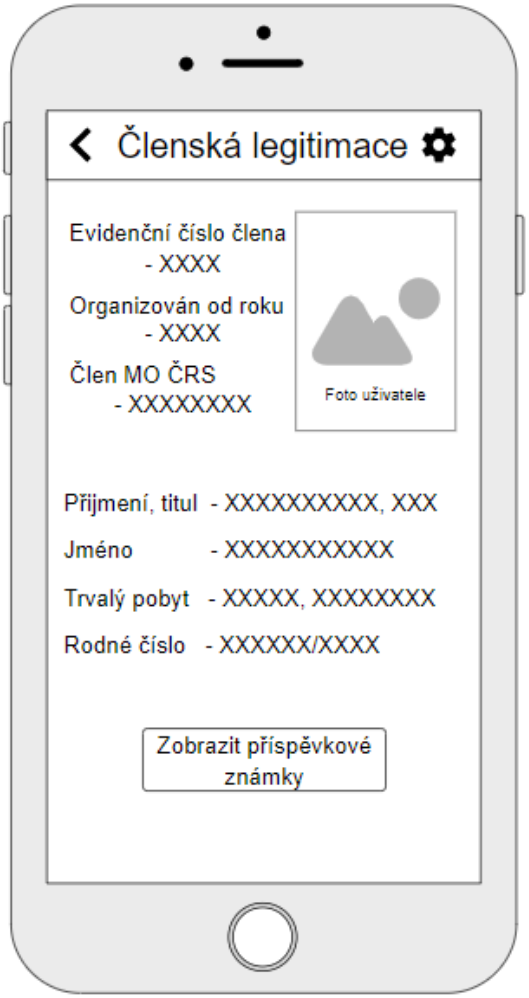
Rybářský řád je dokument, který má několik kapitol, ve kterých se zabývá vším od legislativy až po soupis revíru a jejich popisu. Celý tento dokument zobrazí po kliknutí na tlačítko rybářský řád. Na obrazovce dokumentu rybářský řád by bylo vhodné udělat vyhledávání pomocí klíčových slov. Toto vyhledávání by bylo možné realizovat pomocí formuláře.



Obrázek 14: Rybářský řád

6.6 WF členská legitimace

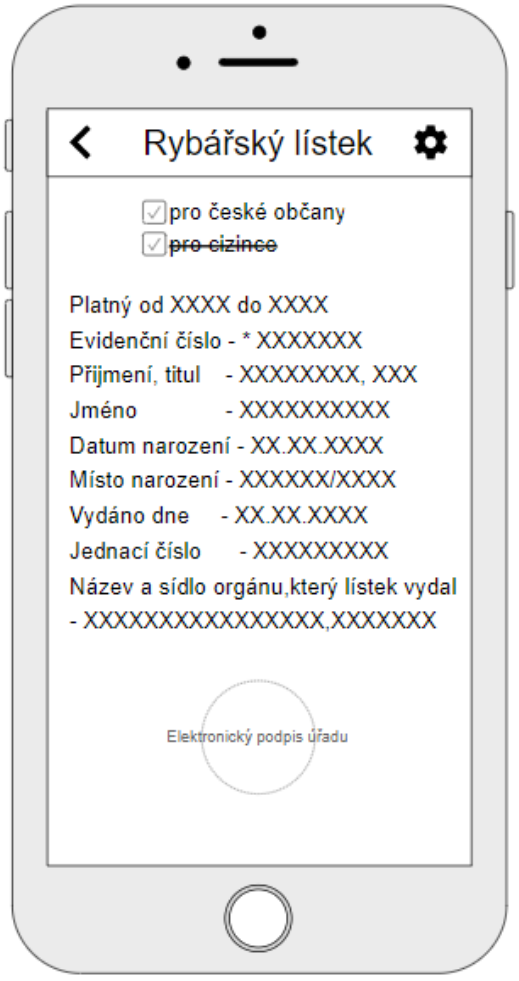
Členská legitimace je identifikační dokument, který uživateli slouží k prokázaní se rybářské stráži. V tomto dokumentu, stejně jako v jeho papírové podobě, nalezneme jméno uživatele, jeho trvalé bydliště, fotku a pod jakou organizaci rybářského svazu patří. Tento dokument uživatel nemůže upravovat, slouží pouze pro identifikaci před rybářskou stráží.



Obrázek 15: Členská legitimace

6.7 WF rybářský lístek

Rybářský lístek, který vydá městský úřad nebo obecní úřad za menší poplatek a za podmínky, že rybář složí zkoušku, slouží jednotlivým místním organizacím rybářského svazu jako doklad pro vydání povolenek k lovu a členské legitimace. Dále tento doklad, stejně jako členská legitimace, slouží při kontrole uživatele porybným. Na tomto dokumentu budou zobrazeny všechny informace, které jsou i na papírovém rybářském lístku. Mezi tyto informace patří jméno, datum narození, evidenční číslo, datum vydání dokladu, platnost dokladu, název úřadu, který tento dokument vydal, podpis a razítko úřadu.



Obrázek 16: Rybářský lístek

6.8 Závěr k WF

Tyto WF jsem navrhl podle papírových verzí aktuálních dokumentů. Jak si lze všimnout z těchto návrhů a z jejich popisů, některé informace, jako jméno, se duplikují. V budoucnu by se tento problém dal vyřešit například vytvořením jednoho dokumentu, ve kterém by byly všechny duplicitní a neduplicitní údaje.

7 Postup tvorby aplikace

Pro naprogramování této aplikace, jak již vyplývá ze zadání, jsem si vybral framework React Native. V následujících sekcích se budu zabývat tím, jak si připravit prostředí pro práci s tímto frameworkem. Při samotné tvorbě aplikace se budu řídit co nejvíce podle navrhnutých WF.

7.1 Příprava prostředí

V dokumentaci frameworku je detailně popsáno vše potřebné pro tvorbu nativních aplikací [13]. Při procházení jejich dokumentace máme na výběr pro tvorbu nativních aplikací mezi nástroji React Native Expo a React Native CLI. Rozdíl mezi nimi je pouze v možnosti rozběhnutí virtuálního stroje na možnost testování a debuggování. Pro tvorbu této bakalářské práce jsem zvolil variantu CLI, protože Expo umožňuje debuggování pouze, při připojení k internetu prostřednictvím stejnojmenné mobilní aplikace, ve které si naskenujeme QR kód, který se vygeneruje při žádosti na webových stránkách (https://www.doplnit.web).

Další nevýhodou oproti React Native CLI je to, že Expo má méně komponent. Expo je určené hlavně pro začátečníky, kteří si chtějí React Native pouze vyzkoušet, než v něm začnou dělat pokročilejší funkční aplikaci. Nicméně není nemožné v něm aplikaci vytvořit.

Dalším krokem pro tvorbu aplikace je vybrání operačního systému. Při volbě Windows podle dokumentace je doporučeno nainstalovat několik balíčků. Tyto balíčky jsou doporučeny naistalovat přes package manager od Chocolatey. Jak si tento package manager naistalovat lze zjistit například na oficiálních stránkám Chocolatey [8]. Po nainstalování tohoto package manageru pak můžeme jednoduše nainstalovat jednoduchým příkazem do konzole všechny potřebné balíčky.

```
choco install -y nodejs.install python2 jdk8
```

Obrázek 17: Instalace potřebných balíčků pomocí Chocolatey

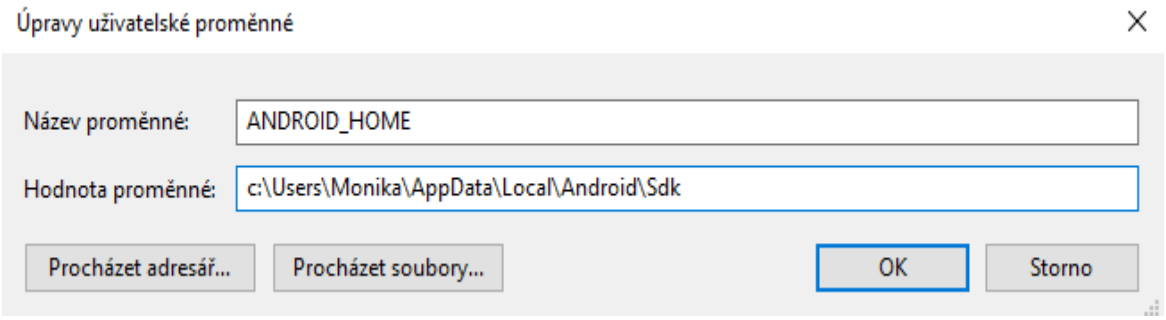
Při zadání takovéhoto příkazu do konzole stáhne Chocolatey nejaktuálnější balíčky, které jsou pro tvorbu aplikace s frameworkem React Native zapotřebí. Konkrétně je zapotřebí Node.js verze aspoň 8.3 nebo novější dále Python 2 a JDK verze 8 nebo novější.
Další věc, která se musí vyřešit je výběr nástroje na kterém se bude zobrazovat a debuggovat aplikace. Zde je opět možné si vybrat, buď se bude používat nějaké fyzické zařízení, nebo je možné zprovoznit v emulátoru virtuální stroj. Při volbě virtuálního stroje je podle dokumentace

doporučeno Android Studio, kde je zapotřebí dodatečné stáhnutí některých balíčků přes SDK manager, který je v Android Studiu vestaven.

- Android SDK
- Android SDK platform
- Performence (Intel @ HAXM)
- Android virtual device

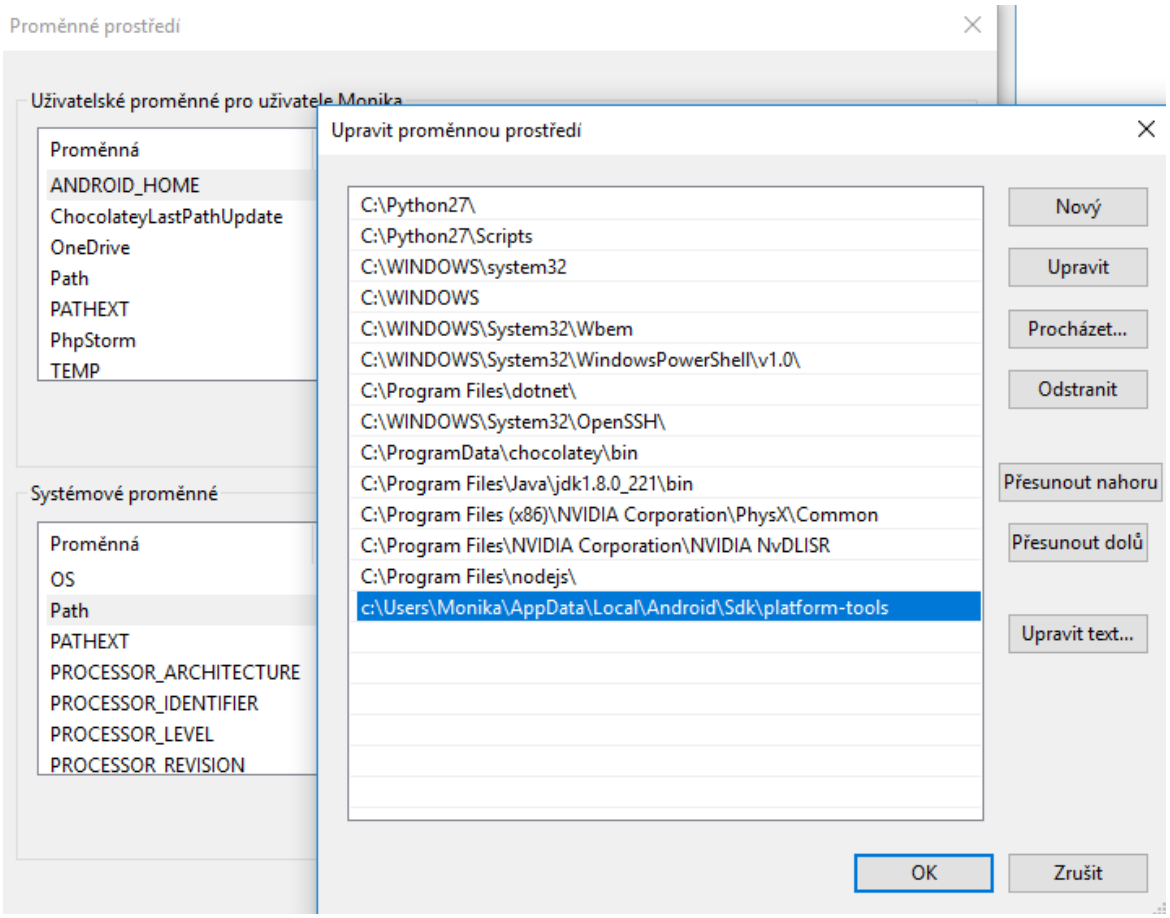
Dále bude zapotřebí stáhnout na virtuální stroj alespoň systém Android 9 (Pie). To je vše, co je nutné přidat do Android Studia.

Další potřebnou věcí pro rozběhnutí virtuálního stroje a možnost vidět živě úpravy, které provádíme v kódu je potřeba přidat jednu systémovou proměnnou a jednu uživatelskou proměnnou. Do uživatelské proměnné musíme přidat proměnnou s názvem ANDROID_HOME.



Obrázek 18: Uživatelská proměnná ANDROID_HOME

Do hodnoty proměnné se musí zadat cesta ke složce s SDK, kterou nalezneme ve složce, kde je nainstalované Android Studio. Druhá proměnná prostředí je již systémem vytvořena. Jedná se o proměnnou PATH. Do této proměnné se musí přidat nová cesta ke složce platform-tools.



Obrázek 19: Systémová proměnná PATH

Cestu ke složce platform-tools nalezneme opět ve složce, kde máme nainstalované Android Studio a ve složce SDK. Toto byl poslední krok pro přípravu prostředí pro tvorbu s frameworkem React Native na zařízení s operačním systémem Windows.
Teď už stačí pouze založit nový projekt a je možné začít s vývojem aplikace.

7.2 Založení nového projektu a spuštění virtuálního zařízení

V předchozí sekci bylo probráno, jak si připravit prostředí pro vytvoření projektu ve frameworku React Native a jak si zprovoznit virtuální stroj s operačním systémem Android pro testování a debuggování. Nyní se budeme zabývat, jak vytvořit nový projekt, a jak spustit virtuální stroj. Jelikož je React Native postaven na rozhraní příkazové řádky, tak založení nového projektu bude také přes příkazovou řádku. Veškeré nové komponenty, které neobsahuje náš projekt v základu, ale budou zapotřebí se také musí nainstalovat přes příkazovou řádku.

Nyní se založí nový projekt, jelikož je předmětem této bakalářské práce naprogramování mobilní aplikace s názvem rybářský lístek, tak se tento projekt také takhle pojmenuje. Příkaz, kterým se vytvoří nový projekt, je na následujícím obrázku.

```
C:\Users\Monika\Desktop\React native projects>npx react-native init RybarskyListek
```

Obrázek 20: Vytvoření React Native projektu

Jak je vidět na obrázku, projekt se zakládá ve složce React Native projects. Po spuštění příkazu se začne vytvářet projekt ve stejné složce. Po vytvoření tohoto projektu se může tento projekt otevřít více méně v jakémkoliv prostředí. Projekt můžeme otevřít buď tak, že ho najdeme jako složku v nějakém vývojovém prostředí, které podporuje React Native, nebo v případě používání Visual Studio Code se může projekt otevřít rovnou přes příkazovou řádku, tak že se před umístění souboru s projektem napíše příkaz code, který rovnou tuto složku otevře ve vývojovém prostředí Visual Studio Code.

```
C:\Users\Monika\Desktop\React native projects\RybarskyListek> code .
```

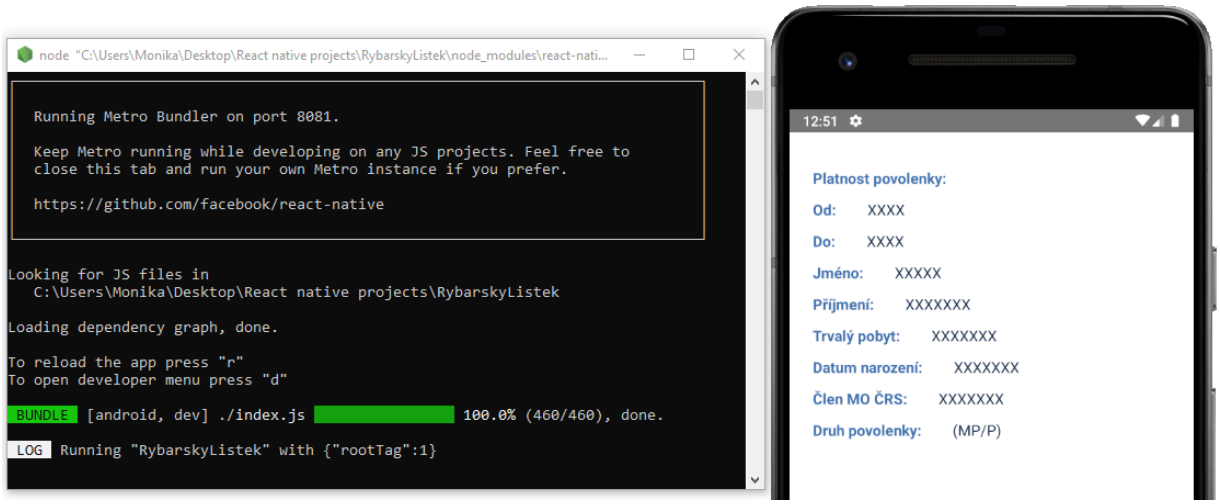
Obrázek 21: Otevření vytvořeného projektu ve Visual studio code

Teď je vytvořený projekt, ale aby byly vidět provedené změny živě a šly dobře debuggovat, tak je zapotřebí ještě spustit virtuální stroj. Zapnutí virtuálního stroje se dělá také přes příkazovou řádku následujícím příkazem.

```
React native projects\RybarskyListek>npx react-native run-android
```

Obrázek 22: Rozběhnutí virtuálního stroje

Opět se musí tento příkaz spustit ve složce s vytvořeným projektem. Po zadání tohoto příkazu se po chvilce spustí lokální server s pomocí Node.js, přes který probíhá komunikace z vytvořeného projektu k virtuálnímu stroji, a živě jsou vidět veškeré úpravy, které jsou v kódu provedeny. Příkazová řádka také slouží k debuggování případných chyb.



Obrázek 23: Status Node serveru a zapnutý emulátor

Pomocí těchto kroků je projekt vytvořený a připravený ke spuštění na emulátoru, který se automaticky aktualizuje při změnách v kódu. V následujících kapitolách je popsáno, jak byla aplikace naprogramována a jaké komponenty byly použity.

7.3 Přihlašovací obrazovka

Přihlašovací obrazovka je první obrazovka, která se po spuštění aplikace zobrazí. Tato obrazovka je tvořena z několika komponent. Základní komponenta, která je mnou vytvořena je (BaseScreen). Tato párová komponenta je složena ze základních komponent, jako je například (View) a slouží k základnímu definování stylů a chování obrazovky. V této komponentě je použita další komponenta (View), tato komponenta slouží jako container view. Její obdobou v HTML je tag . Uvnitř této komponenty (View) je další komponenta (Image) tato komponenta umožňuje zobrazovaní obrázků (na této obrazovce logo ČRS) a jejich stylování. Pro lepší možnosti pozicování je zabalena ve (View) komponentě. Dále jsou na přihlašovací obrazovce dvě komponenty, které se jmenují (TextField). Tyto komponenty slouží pro zadání přihlašovacích údajů. Komponenta (TextField) je opět složena z již existující komponenty (TextInput). Tato vlastní komponenta navíc obsahuje stylování tak, aby všechny vstupy napříč aplikací byly konzistentní. Poté je zde komponenta (Spacer), tato komponenta umožňuje odsazení dvou komponent, v tomto případě (AppButton) vertikálně od (View) kontejneru. Poslední komponentou, která je na přihlašovací obrazovce je (AppButton). Tato komponenta je taktéž vlastní z důvodu konzistence vzhledu.

```
render() {
  const { idValid, passValid } = this.state
  let validInput = idValid && passValid

  return(
   <BaseScreen>
    <View style={styles.container}>
     <Image
      source={require('../Assets/logo.png')}
      style={styles.logo} />
     <View style={styles.inputStack}>
      <TextField
       style={styles.input}
       placeholder='ID'
       validation={Validator.nonEmptyString}
       onValidationChange={valid => {
        this.setState({
         idValid: valid
        })
       }}
       onChangeText={(text) => {
        this.setState({ id: text })
       }} />

      <TextField
```

```
          style={styles.input}
          placeholder='Heslo'
          validation={Validator.nonEmptyString}
          onValidationChange={valid => {
            this.setState({
              passValid: valid
            })
          }}
          onChangeText={(text) => {
            this.setState({ pass: text })
          }} />

        <Spacer space={16} />
        <AppButton disabled={!validInput} title='Login' onPress={this.onLoginButtonPress} />
      </View>
    </View>
  </BaseScreen>
 )
}
```

Každou komponentu, která se na dané obrazovce využívá, je nutné naimportovat následujícím příkazem na začátku souboru.

```
import BaseScreen from '.. /Component/BaseScreen';
import TextField from '../Component/TextField';
import AppButton from '../Component/AppButton';
```

7.4 Hlavní menu

Hlavní menu se zobrazí po kliknutí na tlačítko přihlásit na přihlašovací stránce. Na obrazovce hlavního menu se zobrazí šest tlačítek, což je více než bylo v návrhu v logickém designu. Pro tuto odchylku od logického designu bylo rozhodnuto ze dvou důvodů.

Prvním důvodem je fakt, že pro navigaci mezi více obrazovkami se musí naimportovat balíček pro navigaci a jelikož v React Native se každá obrazovka musí zavolat funkcí, která danou obrazovku vykreslí, tak při použití dvou navigátorů, které by vyžadovaly logický design by vznikla složitá a zbytečná rekurze.

Druhým důvodem je, že nově navržené menu vypadá lépe a je stejně nebo i více přehledné.

Na kódu níže je vidět, jak je vyřešena obrazovka hlavního menu. Je použita vlastní komponenta (BaseScreen), která je nastylována pouze na hlavní rozmístění komponent. V této komponentě je opět komponenta (View), která vymezuje prostor pro obsažená tlačítka. Komponenta (MainMenuItem) je tlačítko s atributem item, který obsahuje název tlačítka.

Po kliknutí na tlačítko se aktivuje switch. Podle toho, jaké tlačítko bylo stisknuto switch odešle navigátoru název obrazovky, která se má vykreslit.

```
render() {
    return(
      <BaseScreen topMargin={0}>
        <View style={styles.container} >
          <MainMenuItem style={styles.item} item={new MenuItem(MenuItemType.startHunt)} onPre
ss={this.onItemPress} />
          <MainMenuItem style={styles.item} item={new MenuItem(MenuItemType.huntPolicy)} onPr
ess={this.onItemPress} />
          <MainMenuItem style={styles.item} item={new MenuItem(MenuItemType.legitimate)} onPre
ss={this.onItemPress} />
          <MainMenuItem style={styles.item} item={new MenuItem(MenuItemType.huntTicket)} onPr
ess={this.onItemPress} />
          <MainMenuItem style={styles.item} item={new MenuItem(MenuItemType.approvementDetai
l)} onPress={this.onItemPress} />
          <MainMenuItem style={styles.item} item={new MenuItem(MenuItemType.huntHistory)} onP
ress={this.onItemPress} />
        </View>
      </BaseScreen>
    )
  }
  private onItemPress(item: MenuItem) {
    const navigation = this.props.navigation
    switch (item.type) {
      case MenuItemType.startHunt:
        navigation.navigate(Router.ScreenName.HuntDetail)
        break
      case MenuItemType.huntPolicy:
        navigation.navigate(Router.ScreenName.Policy)
        break
      case MenuItemType.legitimate:
        navigation.navigate(Router.ScreenName.IdCard)
        break
      case MenuItemType.huntTicket:
        navigation.navigate(Router.ScreenName.Ticket)
        break
      case MenuItemType.approvementDetail:
        navigation.navigate(Router.ScreenName.Approvement)
        break
      case MenuItemType.huntHistory:
        navigation.navigate(Router.ScreenName.HuntHistory)
        break
    }
  }
}
```

MainMenuItem komponenta je vytvořena z existující komponenty TouchableOpacity. Tlačítku je také nastaven styl flex:1. Toto nastavení v React Native zajistí, kolik prostoru tlačítko bude zabírat vzhledem k ostatním komponentám. Reprezentuje poměr oproti jiným komponentám na obrazovce. Na většinu obrazovek uživateli stačí toto stylování a nemusí se vývoj komplikovat vytvářením vlastních balíčků pro odlišné druhy zařízení.

7.5 Začít lov a přidat úlovek

Na obrazovku Lov se uživatel dostane z hlavního menu po stisknutí tlačítka "Začít lov". Na této obrazovce uživatel nalezne formulář a dvě tlačítka. První vstup formuláře zobrazuje aktuální datum, druhý slouží pro zapsání revíru a třetí pro zapsání podrevíru. Aktuální datum je vyplněno automaticky, revír je povinné pole, které uživatel musí vyplnit dle platného číslování. Podrevír povinný není. Po vyplnění revíru se aktivuje tlačítko pro přidávání úlovků a na ukončení lovu. Při kliknutí na tlačítko Přidat úlovek se zobrazí další obrazovka, na které je další formulář a jedno tlačítko.

První vstup formuláře slouží pro zadání druhu ulovené ryby, druhý na délku ryby a třetí na její váhu. Po vyplnění těchto polí a stisknutí tlačítka "Přidat úlovek" se zavolá metoda onSaveButtonPress(). Tato metoda uloží zadané informace o úlovku a předá je v callbacku do předchozí obrazovky, na kterou se rovnou vrátí.

```
  private onSaveButtonPress() {
    const { kind, weight, length } = this.state
    let newCatch = new Catch(kind, Number(length), Number(weight))

    let hunt: Hunt = this.props.navigation.getParam('hunt')
    let callback: ((_: Hunt) => void) = this.props.navigation.getParam
('callback')
    hunt.catches.push(newCatch)
    callback(hunt)

    this.props.navigation.goBack()
  }
```

Tyto údaje z obrazovky Přidat úlovek se mi zobrazují u toho konkrétního lovu na obrazovce Lov pomocí metody renderCatches() do té doby, než kliknu na tlačítko ukončit lov. Toto zobrazování úlovku nebylo podle logického designu naplánováno, ale když jsem zkoušel zobrazování obsahu pole, tak mě napadlo, proč nezobrazovat rovnou tyto úlovky u toho daného

lovu přece jenom, tak uživatel dostane lepší přehled o tom, co ten daný den chytil. A nemusel by to zjišťovat až potom z historie lovů.

```
  private renderCatches() {
    if (this.state.hunt.catches.length > 0) {
      return (
        <View style={{flex: 1, alignSelf: 'stretch'}} >
          <Text style={styles.catchesTitle} >{`Úlovky (${this.state.hunt.catches.length})`}</Text>
          <CatchesList catches={this.state.hunt.catches} style={{ backgroundColor:Colors.secondary1 }
} />
          <Spacer space={16} />
        </View>
      )
    }
    return (
      <Spacer dynamic/>
    )
  }
```

Tlačítko ukončit lov mi odešle vždy datum, revír, pod revír a v případě že si tedy ponecháváme úlovek tak taky zapsané úlovky, takže i druh, délku a váhu do lokálního uložiště v zařízení. Toto lokání uložiště se jmenuje (StorageManager)[20] a v případě této aplikace nahrazuje databázi rybářského svazu. Do tohoto uložiště ukládám pomocí metody push ().

```
private onEndHuntButtonPress() {
    this.setState({
      loading: true
    })
    StorageManager.shared.push(Constants.Storage.hunts, this.state.hunt).
then(() => {
      this.props.navigation.goBack()
    })
  }
```

7.6 Rybářský řád

Na obrazovku Rybářský řád se uživatel dostane z hlavního menu po kliknutí na tlačítko Rybářský řád. Tato obrazovka byla vyřešena pomocí komponenty (WebView), která má atribut source. V tomto atributu je odkaz na oficiální stránky Rybářského svazu. Jelikož má Rybářský svaz dobře optimalizované stránky pro zobrazování na mobilních zařízeních, byla zvolena varianta s odkazem na ně. Výhodou této varianty je, že web umožňuje snadnější orientaci v rybářském řádu pomocí odkazů na jednotlivé kapitoly. Dalším plusem je, že na tomto webu budou vždy nejaktuálnější informace týkající se rybářského práva, které by se jinak museli řešit aktualizacemi.

```
render() {
  return(
    <BaseScreen>
      <WebView
        source={{ uri: 'https://www.rybsvaz.cz/beta/index.php/legislativa/rybarskyrad' }}
      />
    </BaseScreen>
  )
}
```

7.7 Členská legitimace

Na obrazovku Členská legitimace se uživatel dostane z hlavního menu po kliknutí na tlačítko Členská legitimace. Na této obrazovce jsou zobrazovány informace o držiteli legitimace a o rybářském svazu, který legitimaci vydal. Na obrazovce je fotka uživatele a informace o něm a rybářském svazu. Tyto informace jsou zobrazovány pomocí metody renderList(), která zavolá třídu, ve které je předpis na zobrazování dvou složek title a value. Pro tento způsob zobrazovaní informací byla vytvořena třída, protože to ušetří prostor se stylováním. Tato třída je využita i na dalších obrazovkách.

```
render() {
  return(
    <BaseScreen>
      <View style={styles.container} >
        <Image style={styles.profileImage} source={require('../Assets/profileCar.jpg')}/>
        <Spacer space= {16}/>
        {this.renderList()}
      </View>
    </BaseScreen>
  )
}

renderList() {
  let items: KeyValueItemData[] = [
    new KeyValueItemData('Evidenční číslo člena', '5456132'),
    new KeyValueItemData('Organizován od roku', '2018'),
    new KeyValueItemData('Člen MO ČRS', 'Děčín'),
    new KeyValueItemData('Jméno', 'Evangelos'),
    new KeyValueItemData('Příjmení', 'Katsaros'),
    new KeyValueItemData('Trvalý pobyt', 'Děčín 2135'),
    new KeyValueItemData('Rodné číslo', '463546/12'),
  ]
  return(
    <KeyValueList items={items} />
  )
```

```
  }
}
```

Údaje jsou prozatím zobrazovány staticky. Kdyby byla možnost napojení na databáze rybářského svazu, tak by se v budoucích verzích aplikace tyto informace exportovaly z databáze podle ID uživatele.

7.8 Rybářský lístek

Na obrazovku Rybářský lístek se uživatel dostane z hlavního menu po kliknutí na tlačítko Rybářský lístek. Na této obrazovce jsou zobrazovány podobné údaje jako na obrazovce Členské legitimace s tím rozdílem, že zde není žádná profilová fotka uživatele, ale pouze text a elektronický podpis úřadu který tento doklad vydal. Opět pro zobrazení textových informací používám třídu KeyValueItemData, kterou opět zavolám pomocí funkce renderList(). Dále zde mám jednu (View) komponentu, která mi zobrazuje návrh elektronického podpisu. Tato komponenta obsahuje pouze styl, který nějakým grafickým znázorněním ukazuje možnou šablonu pro elektronický podpis.

```
render() {
  return(
    <BaseScreen>
      <View style={styles.container} >
        {this.renderList()}
        <View style={{flex: 1}} />
        <Text>Elektronický podpis úřadu</Text>
        <View style={styles.signature} />
      </View>
    </BaseScreen>
  )
}

renderList() {
  let items: KeyValueItemData[] = [
    new KeyValueItemData('Platný od', '2010'),
    new KeyValueItemData('do', '2025'),
    new KeyValueItemData('Příjmení', 'Katsaros'),
    new KeyValueItemData('Jméno', 'Evangelos'),
    new KeyValueItemData('Datum narození', '1996'),
    new KeyValueItemData('Místo narození', 'Soluň'),
    new KeyValueItemData('Vydáno dne', '25.8.2018'),
    new KeyValueItemData('Jednací číslo', '123456'),
    new KeyValueItemData('Název sýdla orgánu, který lístek vydal', 'Městský úřad děčín'),
  ]
```

```
    return(
      <KeyValueList items={items} />
    )
  }
}
```

Všechny údaje, které jsou v rybářském lístku jsou zaznamenávány také na městském úřadu na odboru životního prostředí. Tento úřad také vydává rybářské lístky a tato organizace není nijak propojena s rybářským svazem. Proto by bylo potřeba mít svolení od úřadu životního prostředí pro přístupu k databázi držitelů rybářského lísku, aby se tyto informace o uživateli opět natáhli z jejich databáze. Opět jsou tyto informace zobrazovány staticky.

7.9 Detail povolenky

Na obrazovku Detail povolenky se uživatel dostane z hlavního menu po kliknutí na tlačítko Detail povolenky. Tato obrazovka opět poskytuje pouze textové informace. Doklad vydává rybářský svaz. Informace na této obrazovce, stejně jako na obrazovkách Členské legitimace a Rybářského lísku jsou zobrazovány pomocí třídy KeyValueItemData, která je volána pomocí metody renderList().

```
  render() {
    return(
      <BaseScreen>
        <View style={styles.container} >
          {this.renderList()}
        </View>
      </BaseScreen>
    )
  }

  renderList() {
    let items: KeyValueItemData[] = [
      new KeyValueItemData('Vydání povolenky', '2018'),
      new KeyValueItemData('Platnost do', '31.12.2020'),
      new KeyValueItemData('Jméno', 'Evangelos'),
      new KeyValueItemData('Příjmení', 'Katsaros'),
      new KeyValueItemData('Trvalý pobyt', 'Děčín'),
      new KeyValueItemData('Datum narození', '.8.7.1996'),
      new KeyValueItemData('Člen MO ČRS', 'Děčín'),
      new KeyValueItemData('Druh povolenky', '(MP/P)'),
    ]
    return(
      <KeyValueList items={items} />
```

```
    )
  }
}
```

Tyto informace jsou uloženy v databázi rybářského svazu. Jelikož k ní není přístup, zobrazují se informace opět staticky. V budoucnu by bylo možné sjednocení obrazovek členské legitimace a detailu povolenky, protože oba tyto dokumenty vydává rybářský svaz a obsahují duplicitní údaje jako například jméno uživatele. Jediné, v čem se povolenka liší od členské legitimace je platnost dokladu, a typ povolenky, to značí, jestli je určena pro lov na vodách mimopstruhových či pstruhových.

7.10 Historie lovu

Na historii lovu se uživatel dostane z hlavního menu po kliknutí na tlačítko historie lovu. Na této obrazovce se budou zaznamenávat všechny uživatelovi absolvované lovy. Co se týče této obrazovky, tak byl zvolen lehce jiný postup oproti logickému designu.

Je to hlavně z důvodu čitelnosti dat, jelikož mřížka, která byla navržena nebyla ideální, proto je zde zvoleno vypisování do bloku. Tento blok je vykreslován pomocí metody renderItem() v komponentě (FlatList), která umožňuje to, aby byl obsah posouvatelný.

Informace jsou vykreslovány v této komponentě, protože je předpokládáno, že uživatel bude mít více docházek k vodě, než by se vešlo na jednu obrazovku. Data z absolvovaných lovů se vypisují pomocí proměnné hunts z lokální databáze, která byla vytvořena pomocí simple-store a do které byly předtím uloženy záznamy z obrazovky lovu. Tyto údaje se předají do proměnné hunts ve formě pole pomocí metody get. Následně se toto pole předá třídě HuntItem, která se vykreslí pomocí metody renderItem(), a která se volá z FlatList komponenty.

```
state = {
  hunts: []
}

async componentDidMount() {
  let hunts = await StorageManager.shared.get(Constants.Storage.hunts)
  this.setState({
    hunts: hunts
  })
}

render() {
  return(
    <BaseScreen>
```

```
            <FlatList
              style={styles.list}
              renderItem={this.renderItem}
              data={this.state.hunts}
            />
          </BaseScreen>
        )
    }

    private renderItem(item: any) {
        const hunt: Hunt = item.item

        return (
          <HuntItem hunt={hunt} style={styles.huntContainer} />
        )
    }
```

Třída HuntItem je tvořena z několika React Native komponent. Kromě stylu v ní dochází k přebírání hodnot z pole hunt. Jak je vidět níže, tak zde dochází k předávání konkrétních atributů z pole pomocí identifikátoru. Například datum je v simple-store uloženo jako date, ale toto datum je uloženo v nevhodném formátu, proto se musí nejdříve přeformátovat pomocí metody dayMontYearFromDate() do lépe čitelné podoby.

Zde se předává huntingGround jako revír a subhuntingGround jako podrevír. Dále je zde funkce, která vykresluje úlovky. Tato funkce předává parametry z pole hunt. Je vytvořena, protože formát pro vykreslování data, revíru a podrevíru byl nevyhovující. U vykreslování úlovků bylo zapotřebí, aby se každý úlovek vykresloval na jeden řádek, a aby byly u váhy a délky zobrazovány jednotky, protože jsou v poli uloženy jako číslo a bez jednotek by mohly být pro uživatele matoucí. Dalším důvodem, proč je vykreslování úlovků voláno funkcí renderDescription() je ten, že se zavolá víckrát v případě více úlovku během jednoho dne.

```
    render() {
        const hunt = this.props.hunt
        return(
          <View style={[styles.container, this.props.style]} >
            <View style={styles.header} >
              <KeyValueItem title='Datum' value={StringFormatter.dayMonthYearFromDate(hunt.date)} />
              <KeyValueItem title='Revír' value={hunt.huntingGround} />
              <KeyValueItem title='Podrevír' value={hunt.subhuntingGround} />
            </View>
            {this.renderDescription(hunt.catches)}
          </View>
        )
    }
```

```
private renderDescription(catches: Catch[]) {
   return (
      catches.map(item => {
         return (
            <View style={styles.descriptionContainer}>
               <Text style={{fontWeight: 'bold'}} >{item.kind}</Text>
               <Text>{item.weightInKg} kg</Text>
               <Text>{item.lenghtInCm} cm</Text>
            </View>
         )
      })
   )
}
```

8 Závěr

Na závěr bych chtěl zhodnotit můj návrh implementace aplikace Rybářský lístek pro rybářský svaz. Implementace vychází z mého vlastního návrhu a nebyla konzultována s žádnou osobou z rybářského svazu. Ale dle mého názoru je implementace aplikovatelná, v definovaném prostředí s navrženými rolemi.

Při porovnání WF s konečnou aplikaci si můžeme všimnout nejednoho rozdílu. Tento rozdíl vznikl, protože jsem narazil při programování na problém s navigátorem. Tedy s funkcionalitou, která vlastně odkazovala na jednotlivé obrazovky. Tento problém jsem vyřešil odstraněním, nebo spíše předěláním obrazovky hlavního menu. Toto nově naprogramované menu obsahuje jednotlivé funkce, které se dříve nacházely pod tlačítky navrženého čtyř segmentového menu. Další změnou, která vznikla při naprogramování aplikace je, že jsem odstranil základní uživatelské rozhraní, protože při testování aplikace jsem došek k závěru, že některá tlačítka (zpět domů, nebo shození aplikace) jsou přebytečná. To jsou nejzávažnější změny od původních WF.

V práci vznikly ještě dvě méně významné odchylky od původních WF. První vznikla u obrazovky začít lov, kde v naprogramované aplikaci uživatel nyní uvidí, co v daný den chytil, a ne až po ukončení lovu, kdy se úlovky zobrazují v historii lovu. Druhá změna vznikla při zobrazování historie lovu. Při návrhu této obrazovky jsem používal pouze papírové dokumenty, které jsou navrženy ve formě tabulky, což se v aplikaci ukázalo jako nepřehledné, proto jsem změnil design na okénka. Tato forma je daleko přehlednější a infomační hodnotu má stejnou jako navržený WF.

Tato aplikace by přinesla vysokou přidanou hodnotu zejména Rybářskému svazu a celé komunitě rybářů. Výhodou aplikace také je šetření životního prostředí a v neposlední řadě digitalizace dat.

V současné verzi aplikace poskytuje stejnou funkcionalitu jako poskytují nynější dokumenty. V dalších verzích by bylo možné vymyslet spoustu užitečných funkcionalit. První užitečnou funkcionalitou je například automatické doplnění revíru podle geolokace. Jelikož nynější mobilní zařízení obsahují GPS, do aplikace by se mohla přidat mapa revíru, podle které by aplikace dokázala předvyplnit revír, kterému se nachází uživatel nejblíže. Další funkcionalitu, kterou bych uznal za vhodné přidat, by bylo zaplacení hostovací povolenky. Hostovací povolenku často využívají rybáři v případě, kdy jedou mimo svůj svaz a chtějí si zalovit v jiné lokalitě. Nyní se to musí řešit tak, že rybář musí navštívit lokální rybářský svaz, kde v jeho otevírací době musí zaplatit poplatek za hostovací povolenku. Toto by se dalo v mé

aplikaci velice jednoduše vyřešit pomocí online platby. Tato funkcionalita by přinesla rybářskému svazu vyšší zisky.

Kromě toho, že by aplikace mohla rybářským svazům přinést větší zisk, tak by mohla usnadnit práci s přepisováním dat ze všech nynějších papírových povolenek. V současné době to funguje tak, že každý rok se musí do 15. ledna odevzdávat povolenka, ze které musí úředníci přepsat data, a poté zkompletovat statistiky o tom, kde bylo chyceno kolik ryb. Na základě těchto statistik následně probíhají zarybňovací plány revírů. V případě, že by rybáři používali tuto aplikaci, tak by se tato práce usnadnila, protože data by byla již v elektronické podobě.

Další výhodou při paralelním nasazení této aplikace je, že by měly rybářské svazy aktuální informace o jejich revírech. Mohly by díky tomu přizpůsobovat zarybňovací plán v průběhu roku. Nyní si mohou vytvořit statistiku pouze jednou za celý rok. Tyto informace by pomohly rybářskému svazu lépe hospodařit a případně ušetřit peníze.

Všechny výše zmiňovaná vylepšení aplikace jsou možná témata pro navazující akademické práce.

Vlastní zdrojový kód obsahuje přes 1600 řádků kódu. Tento zdrojový kód bude nahrán v informačním systému ČZU jako příloha bakalářské práce.

9 Seznam použitých zdrojů

9.1 Seznam literatury

1) FOWLER, Martin. *Destilované UML*. Praha: Grada, 2009. Knihovna programátora (Grada). ISBN 978-80-247-2062-3.
2) Rybářský řád a soupis mimo pstruhových revírů pro držitele celosvazových povolenek. Praha: Český rybářský svaz, 2007.
3) Rybářský řád a soupis pstruhových revírů pro držitele celosvazových povolenek. Praha: Český rybářský svaz, 2007.
4) ŠUSTA, Marek. *Průvodce systémovým myšlením*. Praha: Proverbs, c2015. ISBN 978-80-260-7602-5.

9.2 Internetové zdroje

5) BLAIR, Ian. *5 Ways You Will Benefit From Wireframing Your Mobile App* [online]. [cit. 2021-03-12]. Dostupné z: https://digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more
6) Button. *React Native* [online]. 2020 [cit. 2021-03-12]. Dostupné z: https://reactNative.dev/docs/button
7) Image. *React Native* [online]. 2020 [cit. 2021-03-12]. Dostupné z: https://reactNative.dev/docs/image
8) INSTALLING CHOCOLATEY. *Chocolatey* [online]. [cit. 2020-11-03]. Dostupné z: https://chocolatey.org/install
9) React Native SDK. *Facebook for developers* [online]. [cit. 2020-11-03]. Dostupné z: https://github.com/react-Native-community
10) Flatlist. *React Native* [online]. 2020 [cit. 2021-03-12]. Dostupné z: https://reactNative.dev/docs/flatlist
11) *Visual-paradigm* [online]. [cit. 2021-03-08]. Dostupné z: https://online.visual-paradigm.com/diagrams/tutorials/use-case-diagram-tutorial/
12) React Native comunity. *GitHub* [online]. [cit. 2021-03-13]. Dostupné z: https://github.com/react-Native-community

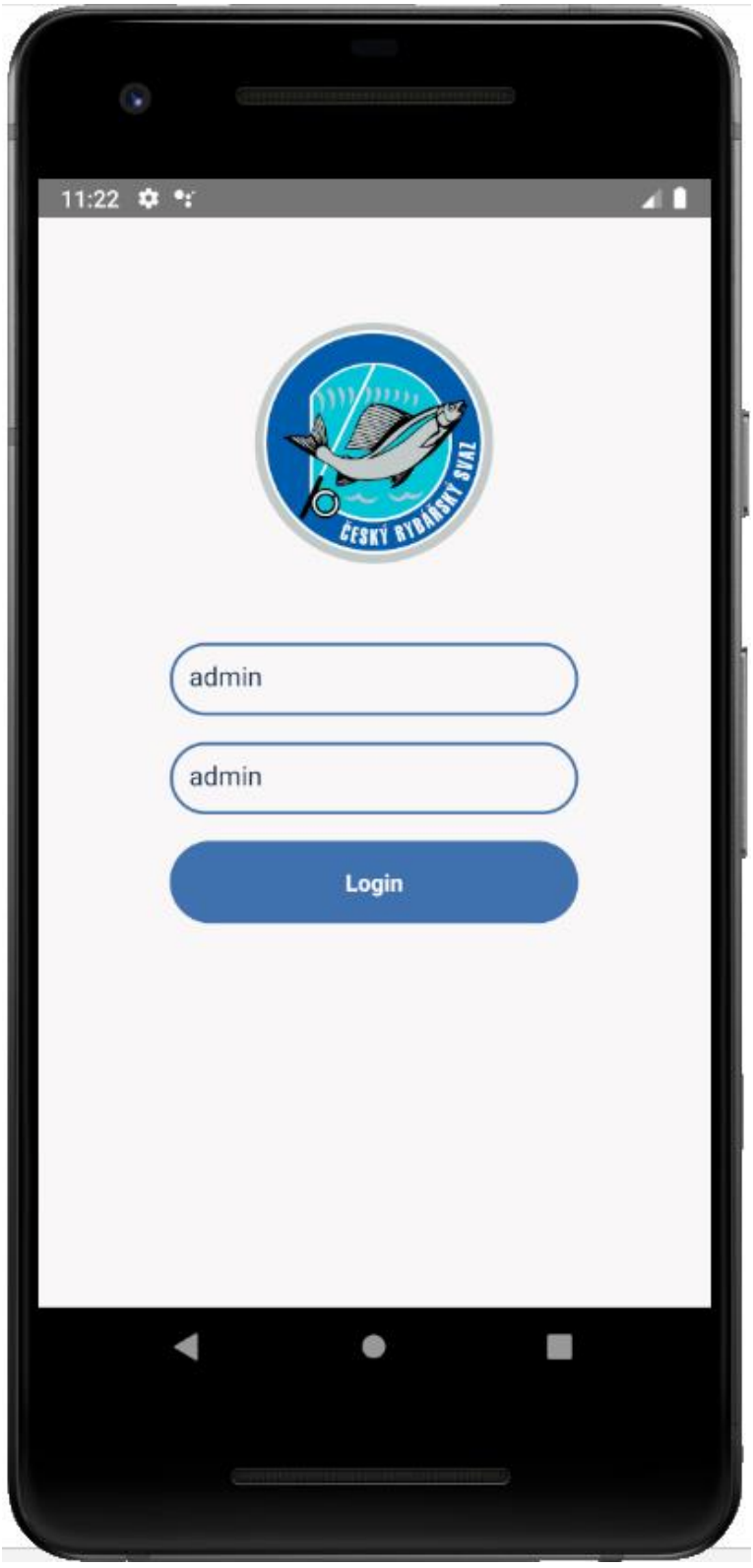
13) Setting up the development environment. *React Native* [online]. [cit. 2021-03-12]. Dostupné z: https://reactNative.dev/docs/environment-setup
14) TextInput. *React Native* [online]. 2020 [cit. 2021-03-12]. Dostupné z: https://reactNative.dev/docs/textinput
15) TouchableOpacity. *React Native* [online]. 2020 [cit. 2021-03-12]. Dostupné z: https://reactNative.dev/docs/touchableopacity
16) ZHANG, Elen. *Digital guardian* [online]. [cit. 2021-03-12]. Dostupné z: https://digitalguardian.com/blog/what-role-based-access-control-rbac-examples-benefits-and-more
17) View. *React Native* [online]. 2020 [cit. 2021-03-12]. Dostupné z: https://reactNative.dev/docs/view
18) Custom WebView. *React Native* [online]. 2020 [cit. 2021-03-12]. Dostupné z: https://reactNative.dev/docs/custom-webview-android
19) Who's using React Native? *React Native* [online]. [cit. 2021-03-14]. Dostupné z: https://reactNative.dev/showcase
20) StorageManager. *Developers* [online]. [cit. 2021-03-13]. Dostupné z: https://developer.android.com/reference/android/os/storage/StorageManager
21) Core Components and APIs. *React native* [online]. [cit. 2021-03-15]. Dostupné z: https://reactnative.dev/docs/components-and-apis

9.3 Obrázky

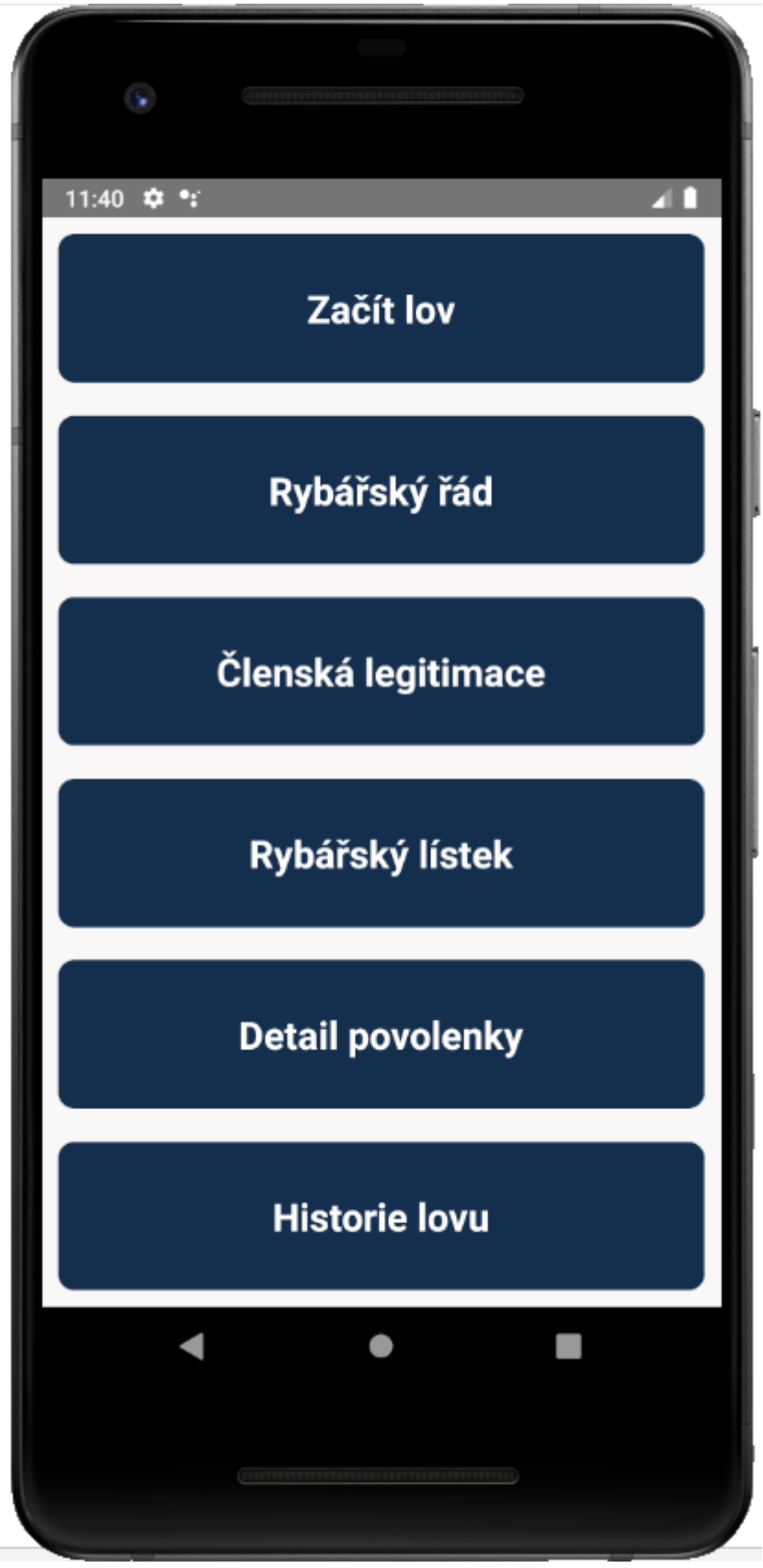
22) *Visual-paradigm* [online]. [cit. 2021-03-08]. Dostupné z: https://online.visual-paradigm.com/diagrams/tutorials/use-case-diagram-tutorial/

10 Přílohy

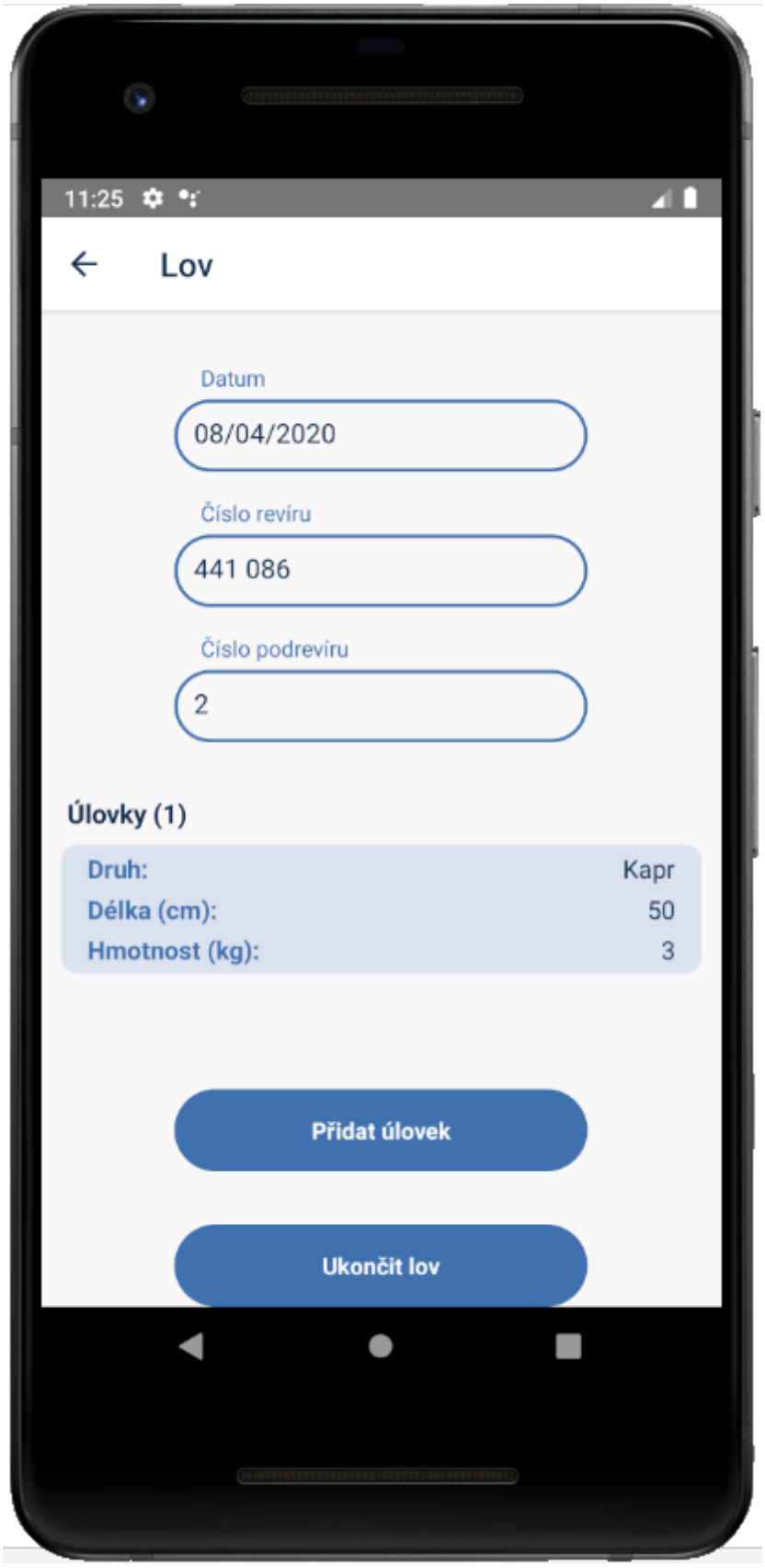
10.1 Přihlašovací obrazovka



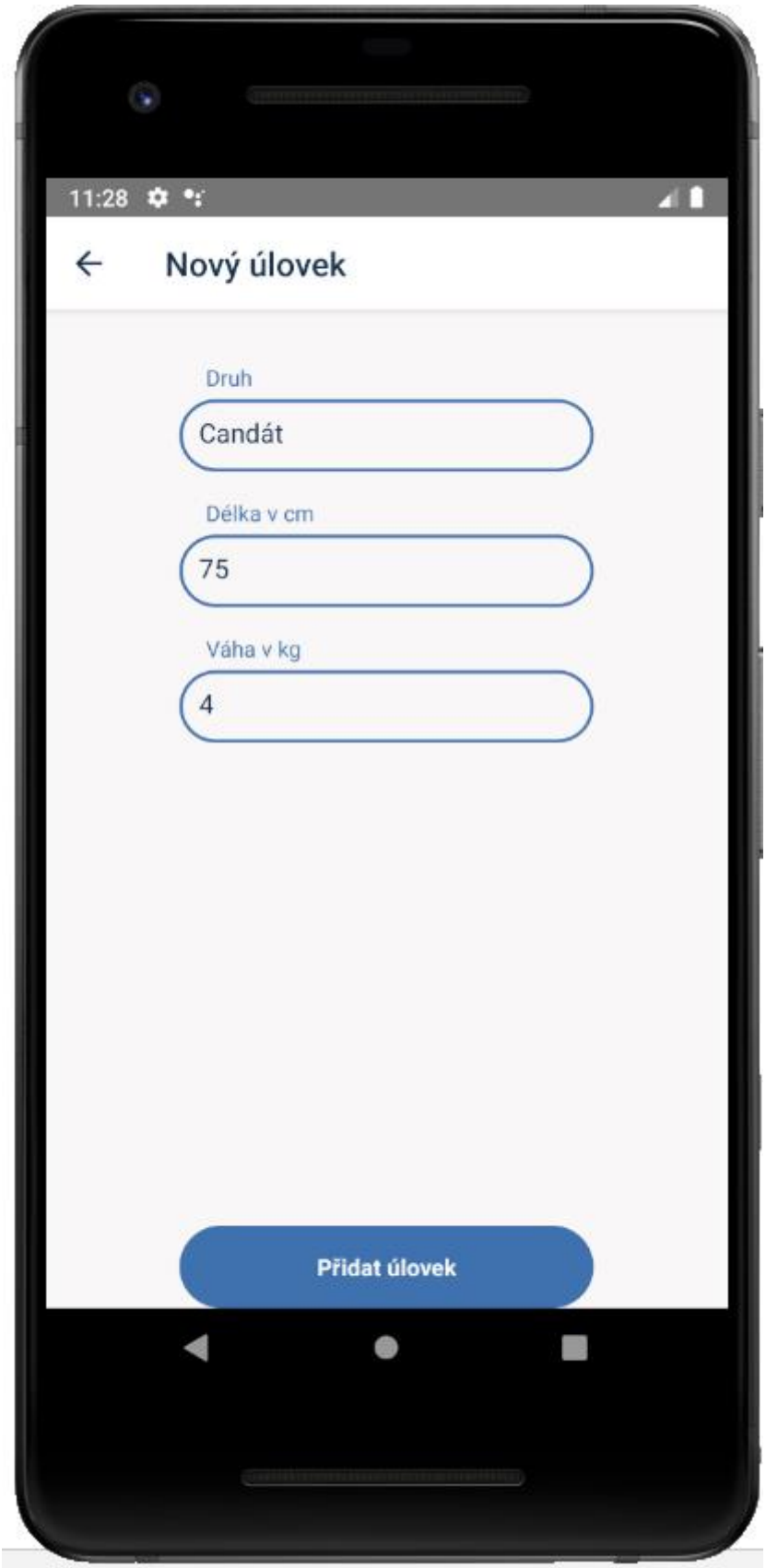
10.2 Obrazovka hlavního menu



10.3 Obrazovka lovu

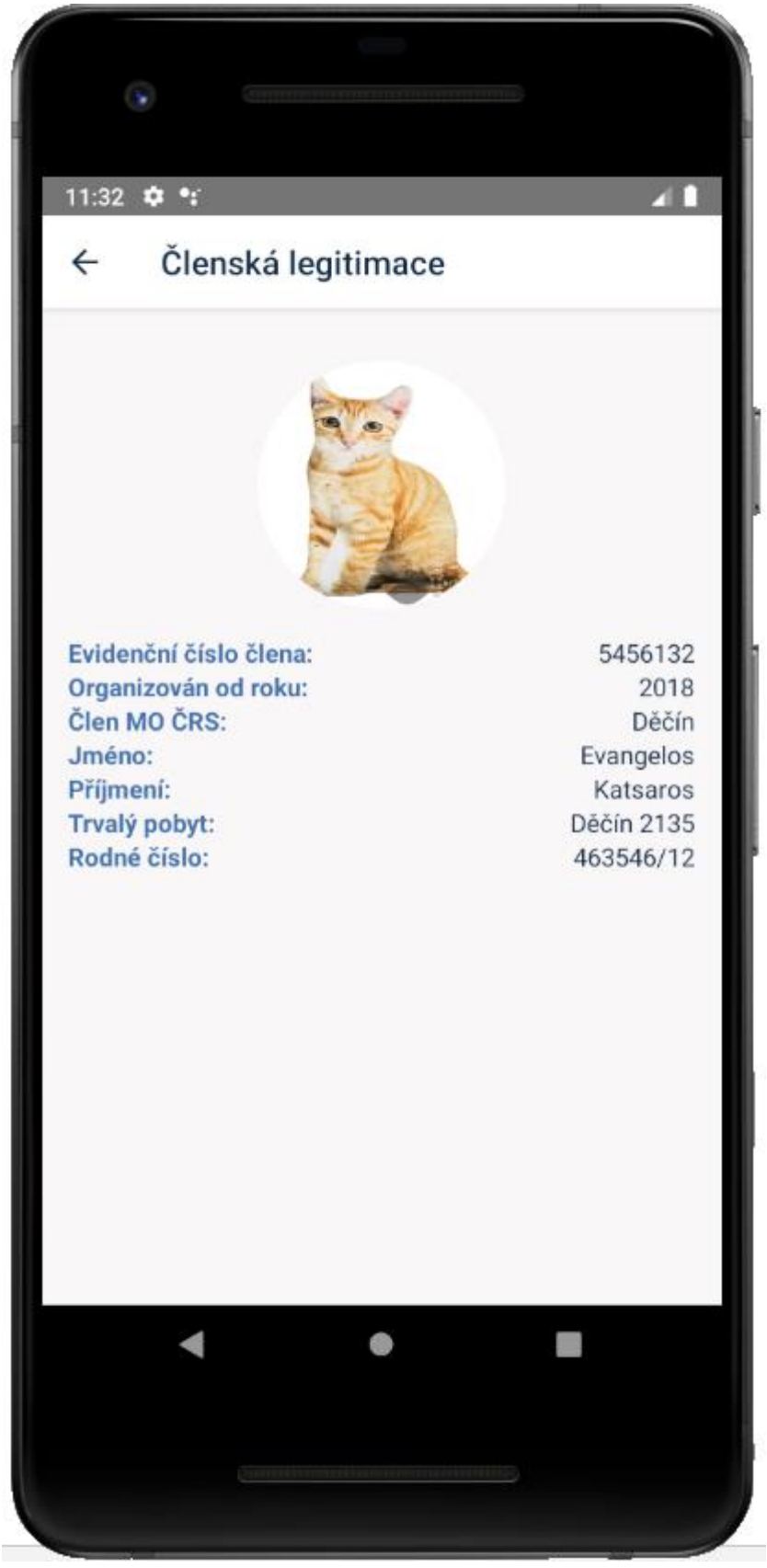


10.3.1 **Obrazovka přidání úlovku**

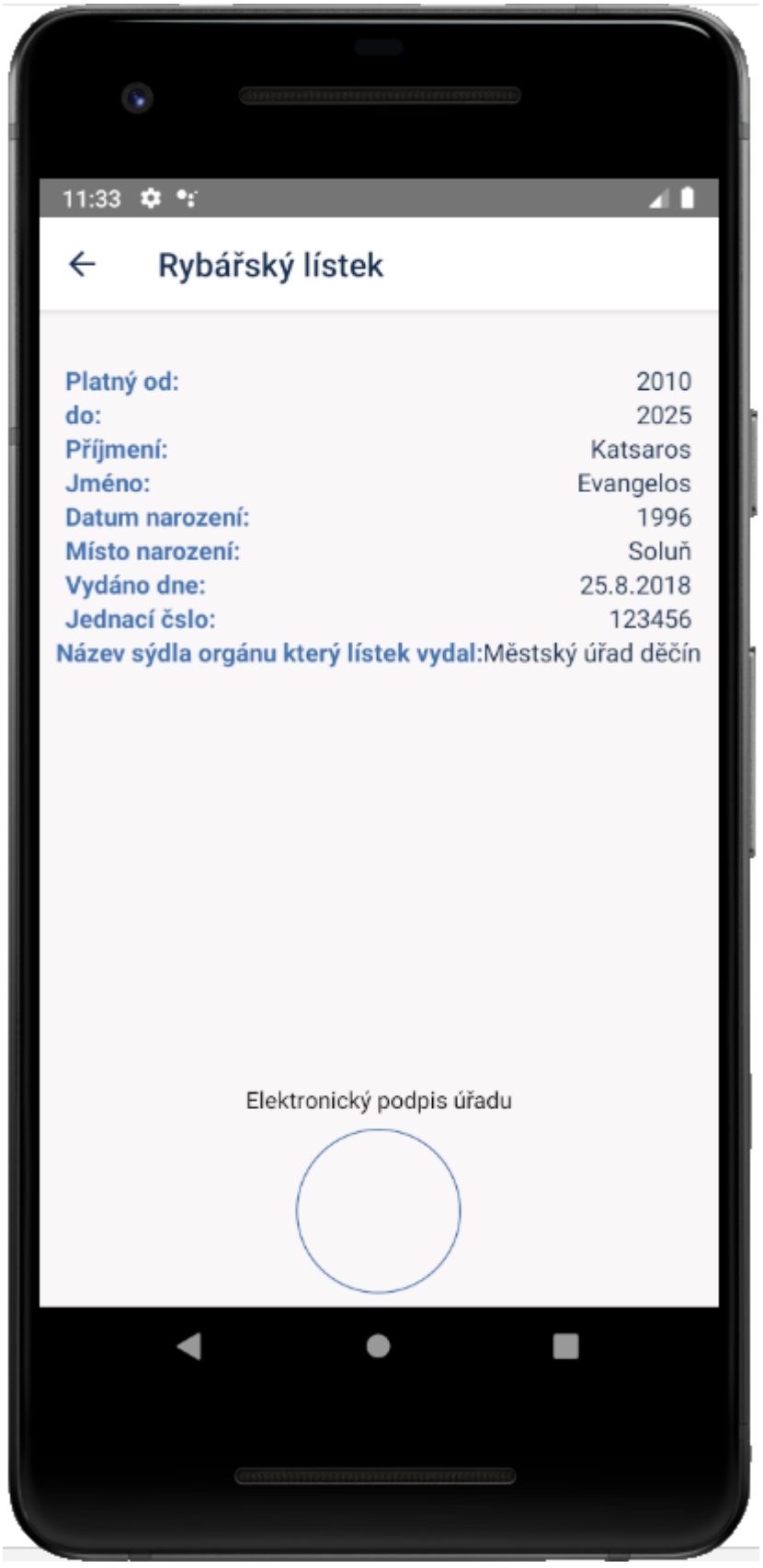


10.4 Obrazovka rybářského řádu

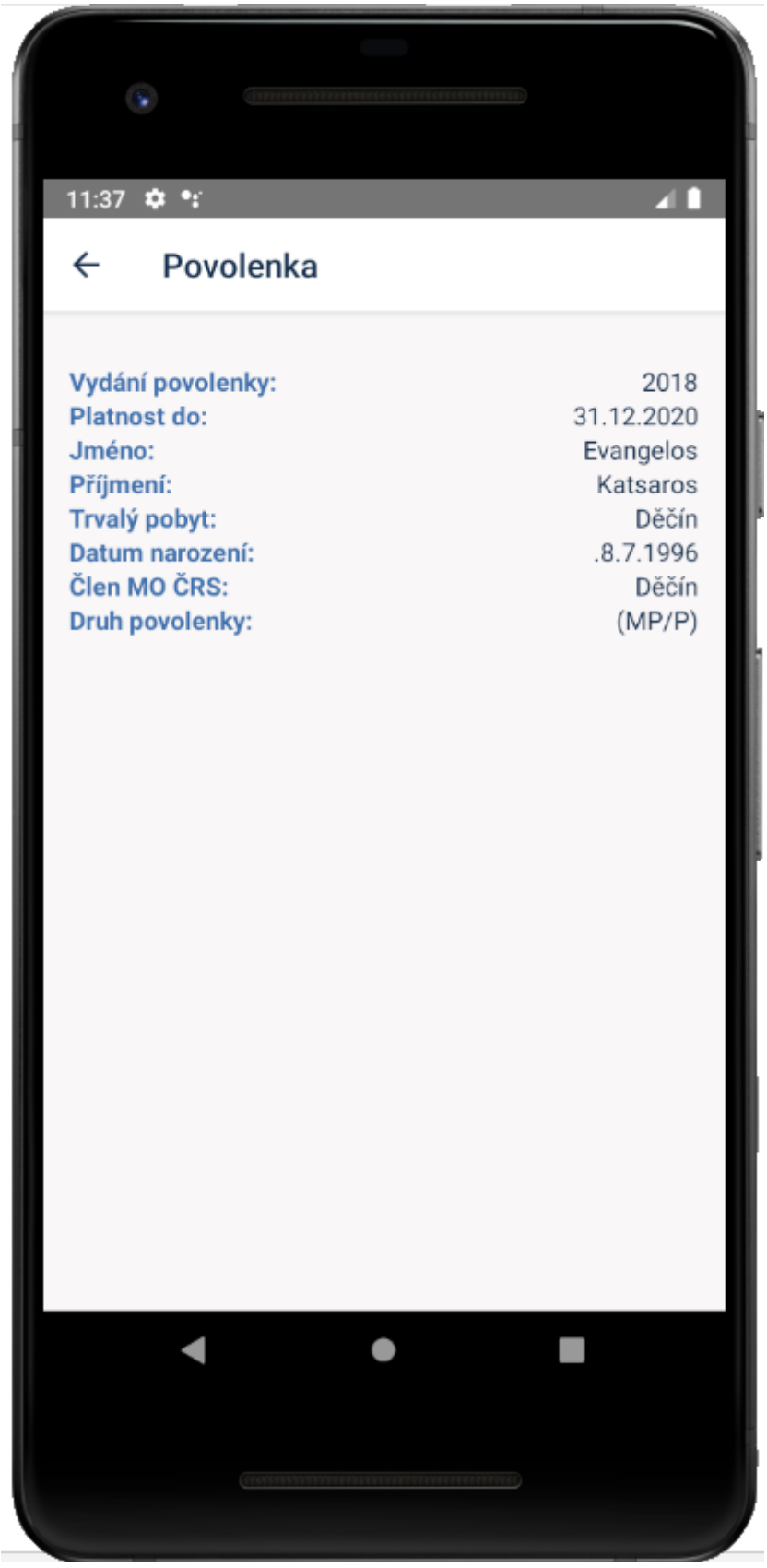
10.5 Obrazovka členské legitimace



10.6 Obrazovka rybářského lístku



10.7 Obrazovka povolenky k lovu



10.8 Obrazovka historie lovu