mistrovství

Ľuboslav Lacko

Android

KOMPLETNÍ PRŮVODCE VÝVOJÁŘE

- Návrh layoutu i aplikační logiky
- Senzory a jimi poskytovaná data
- Určování polohy a navigace
- Napojení aplikace na sociální sítě a cloudové služby
- Vývoj pro Android Wear a Android TV
- Testování a ladění, umístění aplikace do obchodu

computer press

Mistrovství Android

Vyšlo také v tištěné verzi

Objednat můžete na
www.computerpress.cz
www.albatrosmedia.cz

Ľuboslav Lacko
Mistrovství – Android – e-kniha

ALBATROS MEDIA a.s.

Ľuboslav Lacko

Mistrovství Android

Computer Press
Brno
2017

Obsah

Úvod

Android je v současnosti nejpopulárnější operační systém pro mobilní zařízení, jeho tržní podíl překračuje 80 %. Proto je tato platforma obrovskou výzvou pro vývojáře aplikací. Více než miliarda majitelů chytrých telefonů a tabletů s tímto operačním systémem má obrovský tržní potenciál nejen pro placené, ale i volně stáhnutelné aplikace, protože i v této kategorii mohou úspěšné aplikace svým tvůrcům vydělávat.

Na rozdíl od iOS, kde potřebujete na vývoj, aspoň v konečné fázi, počítač s operačním systémem OS X, na vývoj aplikace pro Android můžete použít vývojářský počítač jakékoliv platformy, tedy Windows, Linux i OS X. Všechny klíčové nástroje jsou zdarma, takže k vývoji mobilních aplikací pro Android nejsou zapotřebí žádné investice. Aby bylo možné spouštět aplikace na emulátoru, vývojářský počítač s operačním systémem Windows musí mít minimálně 4 GB RAM.

Problém nepředstavuje ani etapa testování, jelikož není důležité otestovat aplikace na „vlajkové lodi", tam určitě nebude mít žádné problémy s výkonem, ale spíše na levných, méně výkonných telefonech, které jsou samozřejmě mezi lidmi mnohem více rozšířeny. Díky stále rychlejším emulátorům dokážete provést celý vývoj a teoreticky i testování většiny aplikací, které nevyužívají speciální hardwarovou výbavu, aniž byste měli hardwarové zařízení k dispozici fyzicky, avšak v závěrečné fázi před publikováním do aplikačního obchodu doporučujeme důkladně otestovat aplikaci i na „železe", nejlépe na telefonu i tabletu.

Publikace předpokládá určité předběžné znalosti:

- Znalost programovacího jazyka Java; postačí i znalost C#, jelikož tento jazyk je odvozený od Javy a je mu velmi blízký
- Znalost základních principů objektově orientovaného programování
- Zkušenosti s používáním moderních integrovaných vývojových prostředí
- Znalost SQL (nejlépe SQLite, ale postačuje základní všeobecný přehled)
- Znalost základů XML, jelikož v tomto formátu se realizuje návrh uživatelského rozhraní

Zpětná vazba od čtenářů

Nakladatelství a vydavatelství Computer Press, které pro vás tuto knihu připravilo, stojí o zpětnou vazbu a bude na vaše podněty a dotazy reagovat. Můžete se obrátit na následující adresy:

Computer Press
Albatros Media a.s., pobočka Brno
IBC
Příkop 4
602 00 Brno

nebo

sefredaktor.pc@albatrosmedia.cz

Computer Press neposkytuje rady ani jakýkoli servis pro aplikace třetích stran. Pokud budete mít dotaz k programu, obraťte se prosím na jeho tvůrce.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nelze. Pokud v některé z našich knih nějakou najdete, ať už v textu nebo v kódu, budeme rádi, pokud nám ji oznámíte.

Veškerá existující errata zobrazíte na adrese *http://knihy.cpress.cz/K2301* po klepnutí na odkaz Soubory ke stažení. (Nejsou-li žádná errata zatím k dispozici, není odkaz Soubory ke stažení dostupný.)

KAPITOLA

1

Nástroje pro vývoj

Vstupním bodem k získání vývojářských nástrojů, návodů a příkladů je stránka *http://developer.android.com*. K vývoji aplikací budete potřebovat dva balíky softwaru:

- Java Development Kit (JDK)
- Android Studio

Ještě před rokem Google doporučoval používat vývojové prostředí Eclipse doplněné o Android Development Tools (ADT) a Android Software Development Kit, jelikož Android Studio bylo dlouho k dispozici pouze ve verzi Early Access Preview a instalace vývojářských nástrojů byla mnohem složitější.

Změna oficiálně doporučeného vývojového prostředí představuje v tomto případě jednoznačně krok vpřed. Změnily se některé technologie, například překladač využívající automatizační nástroj Gradle či IntelliJ IDEA Java IDE. Základní principy a filozofie zůstaly zachovány, změnilo se jen uživatelské rozhraní. I nadále budete využívat programovací jazyk Java a návrh uživatelského rozhraní se stále ukládá, případně i vytváří, v dokumentu XML. Díky tomu je migrace z Eclipse na Android Studio rychlá a bezproblémová. A ještě jedna důležitá informace: Projekty vytvořené v Eclipse můžete velmi snadno importovat do Android Studia bez nutnosti předchozího exportu.

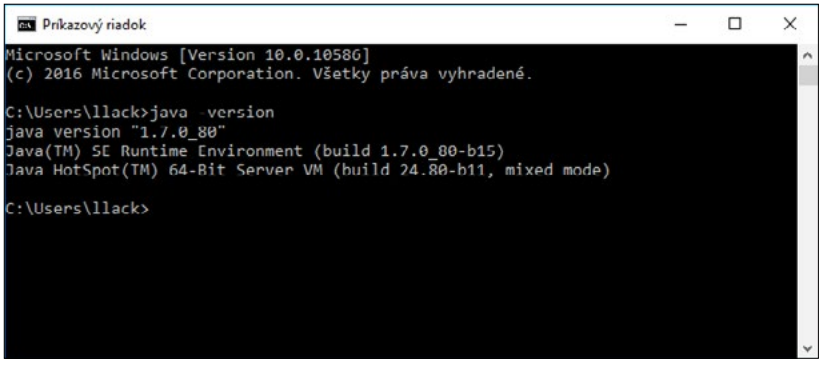
Instalace Javy

Java je objektově orientovaný programovací jazyk, jehož syntaxe je podobná C nebo C++. Java je interpretovaný jazyk, což znamená, že program se nepřekládá přímo do strojového kódu, ale do takzvaného bajtkódu Java. Ten je následně interpretován virtuálním strojem. Java je tak jazyk nezávislý na platformě.

O tom, zda máte, nebo nemáte nainstalovaný Java SE Development Kit a v jaké verzi, se přesvědčíte snadno. Stačí do konzolové aplikace zadat příkaz `java -version`.

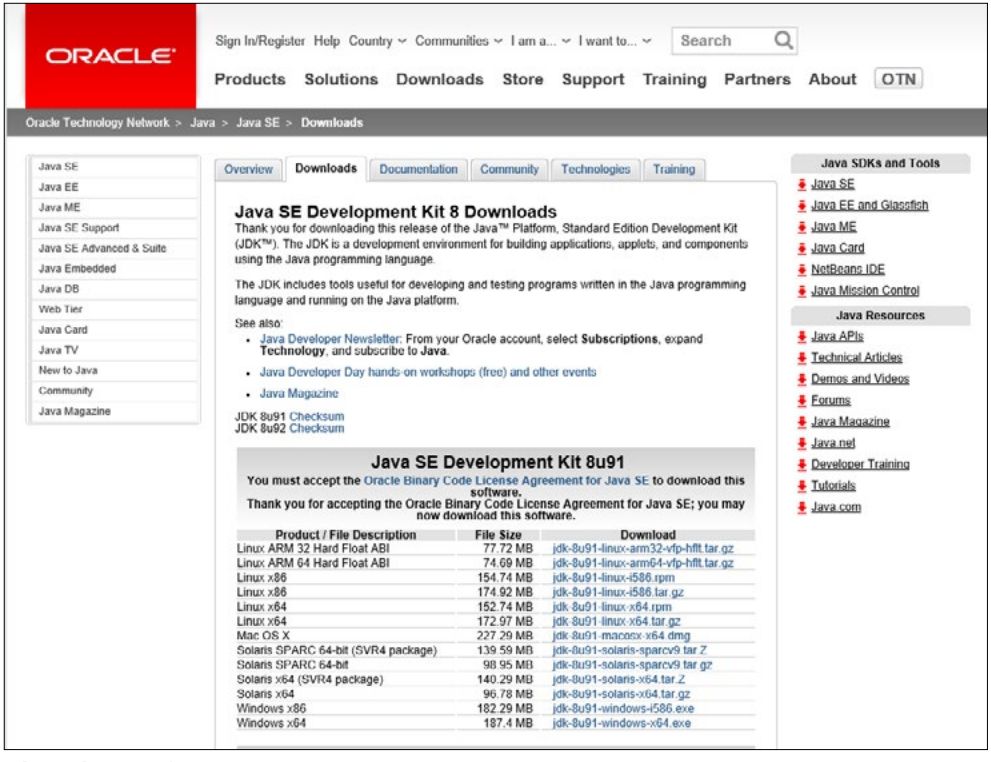
Témata kapitoly:

- **Instalace Javy**
- **Instalace vývojového prostředí Android Studio**
- **První spuštění a konfigurace Android Studia**
- **Průběžná aktualizace**
- **Vytvoření projektu aplikace**
- **Odemknutí zařízení pro vývoj**
- **Seznámení se s vývojovým prostředím**
- **Možnosti současného hardwaru**
- **Referenční zařízení**
- **Vytvoření emulátoru**
- **GitHub**

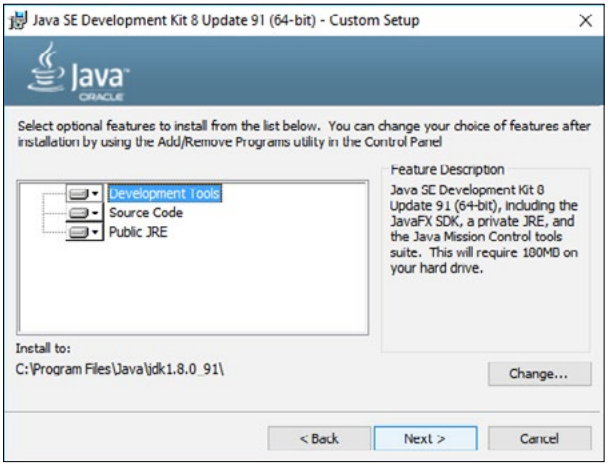


Obrázek 1.1: Identifikace nainstalované verze JDK

Platforma Java byla produktem společnosti Sun. Tuto společnost i se všemi produkty a řešeními získala společnost Oracle, která Javu momentálně spravuje. Z jejích stránek je možné Java SDK stáhnout a nainstalovat. Javu SE Development Kit ve verzi 8, případně novější, stáhnete pro platformy Windows a Linux, pro platformu macOS stáhnete Javu ze stránek společnosti Apple. V závislosti na verzi operačního systému stáhněte 32- nebo 64bitovou variantu JDK.

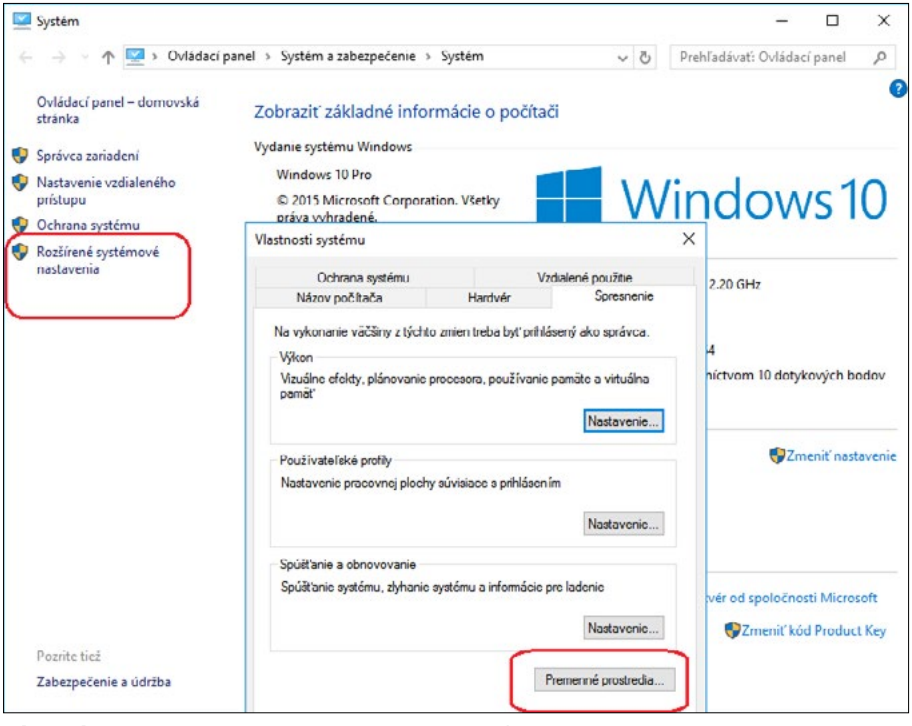


Obrázek 1.2: Vyberte správnou verzi JDK pro váš operační systém

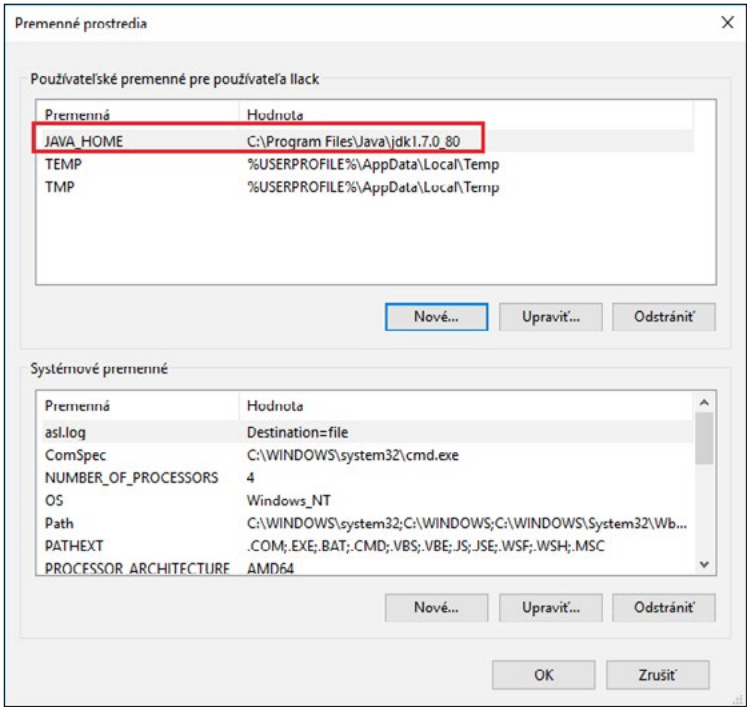


Obrázek 1.3: Instalace JDK

Pokud Android Studio nenajde vámi nainstalované JDK, je potřeba nastavit ještě uživatelskou proměnnou `JAVA_HOME`, aby aplikace používající JDK mohly správně pracovat. V systémových nastaveních Windows 10 klepněte na **Rozšířená systémová nastavení** a v zobrazeném dialogu klepněte na tlačítko **Proměnné prostředí**. Přidejte novou uživatelskou proměnnou `JAVA_HOME` a do ní textový řetězec adresáře, ve kterém máte nainstalované JDK.



Obrázek 1.4: Rozšířené systémové nastavení Windows 10



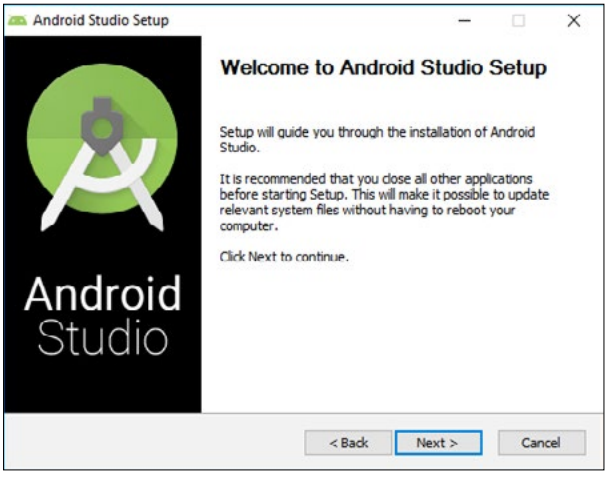
Obrázek 1.5: Nastavení uživatelské proměnné

Instalace vývojového prostředí Android Studio

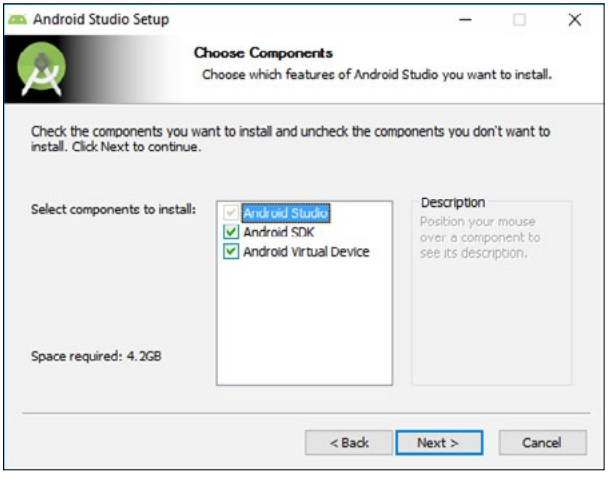
Android Studio jako oficiální nástroj na vývoj aplikací pro Android je k dispozici ke stažení na stránce *developer.android.com* pro platformy Windows, Linux a macOS. Dříve než zahájíte instalaci, nainstalujte balík Java SE Development Kit (viz předchozí podkapitolu). Instalace Android Studia je jednoduchá a proběhne doslova na jedno klepnutí. Stačí stáhnout instalační soubor a spustit ho.

V konfiguračním dialogu instalace doporučujeme ponechat všechny implicitně označené komponenty, tedy:

- Android Studio
- Android SDK
- Android Virtual Device



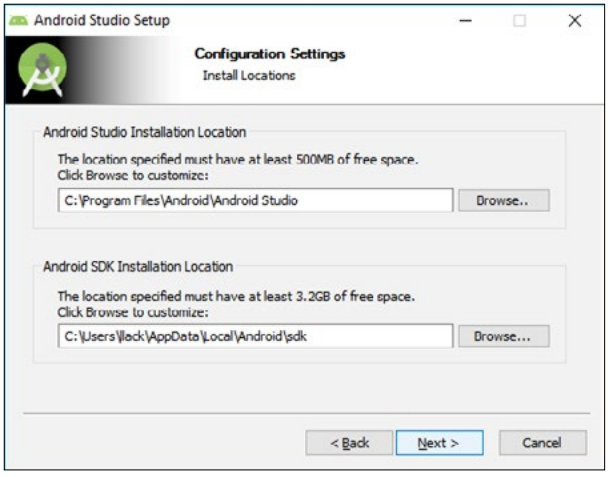
Obrázek 1.6: Instalace vývojového prostředí Android Studio



Obrázek 1.7: Android Studio – výběr komponent k instalaci

V následujícím dialogu se zobrazí složky, do kterých se nainstaluje Android Studio a Android SDK.

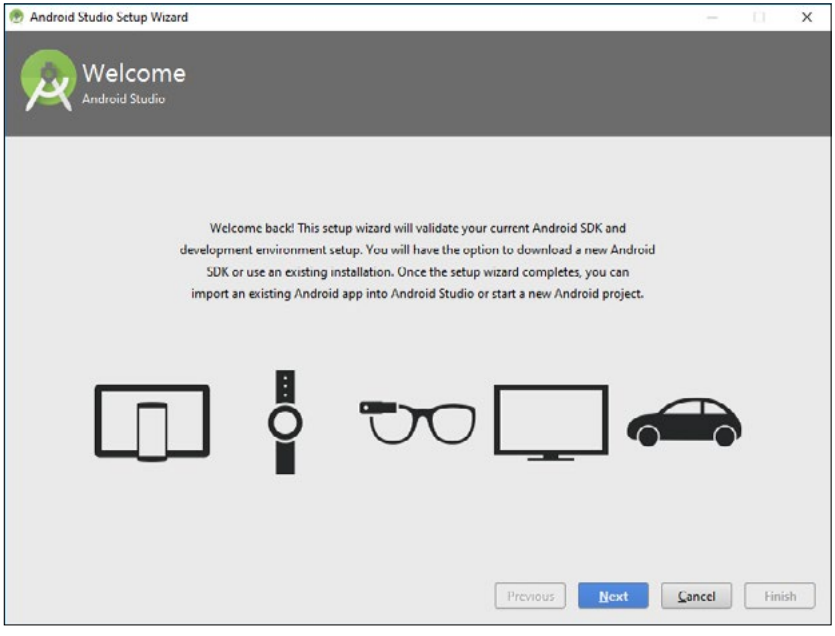
Během instalace se můžete setkat s oznámením firewallu, že Android Studio potřebuje využívat síťovou komunikaci, a firewall si od vás vyžádá povolení k přístupu.



Obrázek 1.8: Android Studio – implicitní adresáře k instalaci

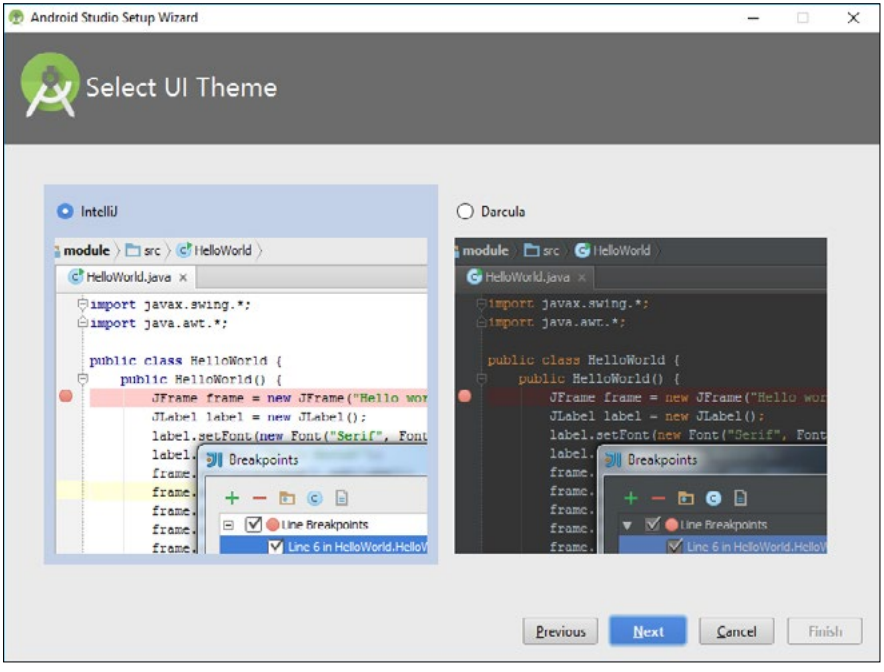
První spuštění a konfigurace Android Studia

V úvodním dialogu **Android Studio Setup Wizard** narazíte na oznámení, že bude následovat kontrola dostupnosti a aktuálnosti verzí Android SDK (Software Development Kit) a v případě potřeby se nainstalují potřebné aktualizace.



Obrázek 1.9: Android Studio Setup Wizard

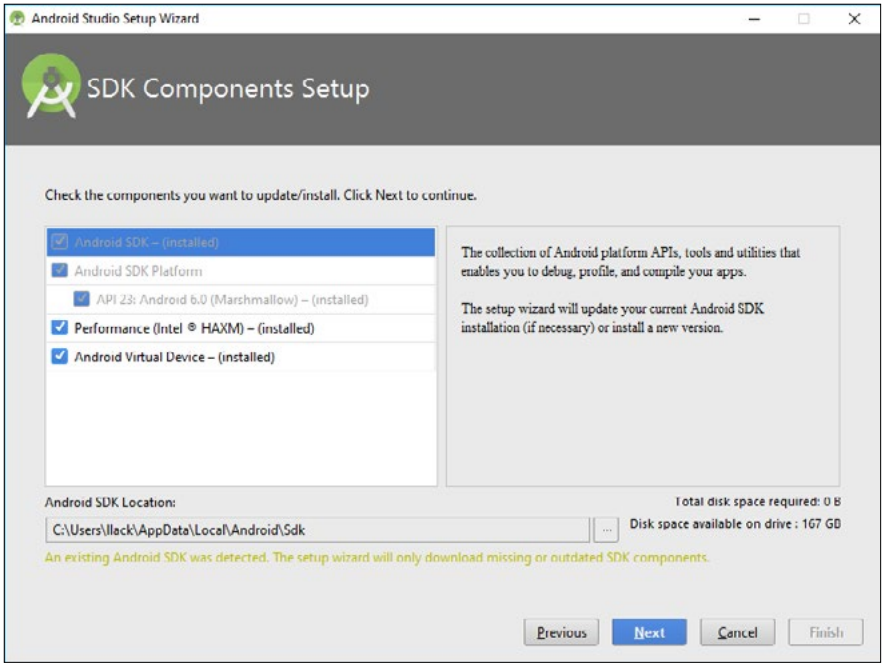
Můžete si zvolit typ instalace **Standard** nebo **Custom**. Pro běžné nasazení doporučujeme ponechat označenou volbu **Standard**.



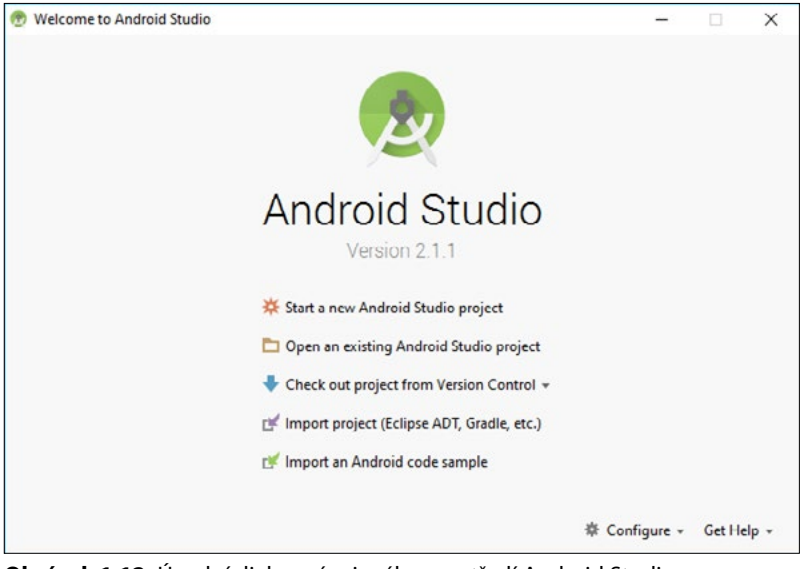
Obrázek 1.10: Volba Custom umožní vybrat barevné schéma uživatelského rozhraní Android Studia

Následuje volba komponent. Pokud už Android SDK máte nainstalované, například pokud jste vyvíjeli aplikace v Eclipse, doinstalují se jen aktualizace chybějící komponenty.

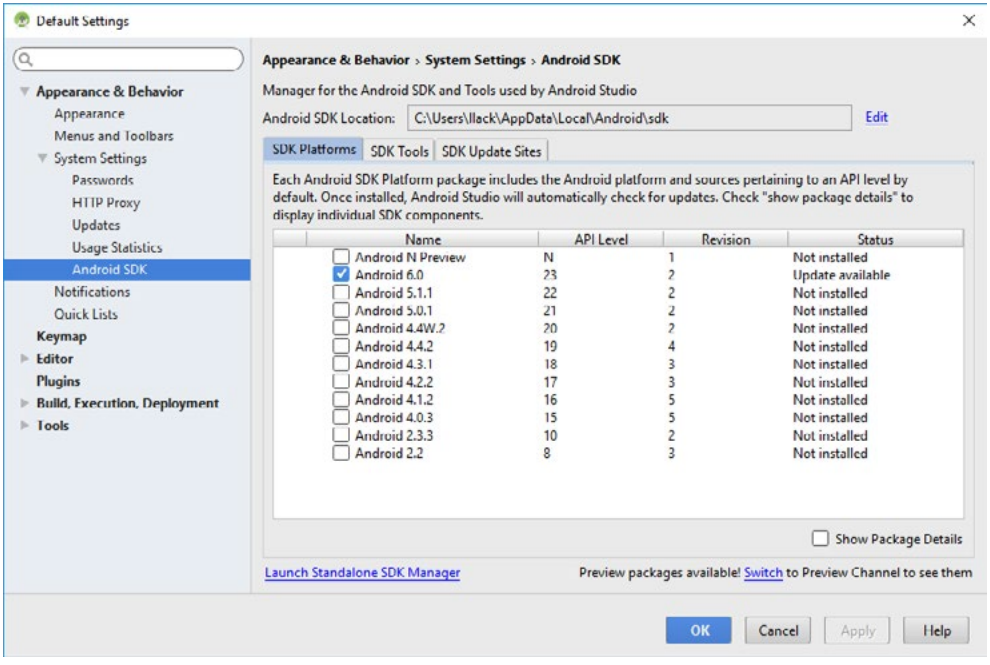
Po nainstalování a prvním spuštění doporučujeme zkontrolovat, pro které verze Androidu máte nainstalovanou podporu. V úvodním dialogu proto ve spodní části klepněte na položku **Configure** a z nové nabídky vyberte **SDK Manager**.



Obrázek 1.11: Volba Custom umožní vybrat barevné schéma uživatelského rozhraní Android Studia



Obrázek 1.12: Úvodní dialog vývojového prostředí Android Studio



Obrázek 1.13: SDK Manager

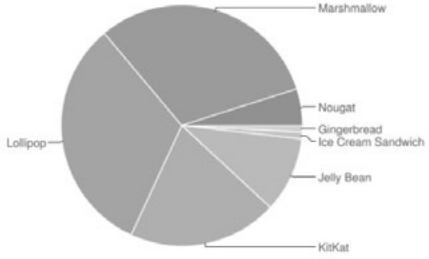
V okně **Android SDK Manager** si všimněte tří karet: **SDK Platforms**, **SDK Tools** a **SDK Update Sites**. Na kartě **SDK Platforms** je seznam SDK pro jednotlivé majoritní verze Androidu. Implicitně je nainstalované SDK pro Android 6.0 Marshmallow, jehož API má verzi 23. Jelikož v době psaní této knihy stále ještě dominovaly nižší verze, doporučujeme nainstalovat i je. V této publikaci budeme jako nejnižší verzi používat Android 5.0.1, tedy verzi API 21. Při rozhodování, zda vyvíjet aplikace, které budou kompatibilní i s verzí Android 4.4 KitKat, případně 4.1 JellyBean, vám poslouží graf procentuálního zastoupení jednotlivých verzí Androidu na *http://developer.android.com/about/dashboards/index.html*. Situaci v době psaní rukopisu dokumentuje obrázek.

Stále mají poměrně velké procentuální zastoupení i starší verze. Podle svého uvážení označte položky, které chcete instalovat, tedy verze Androidu, pro které chcete vyvíjet a testovat aplikace.

Vyvíjet aplikace tak, aby byla zachovaná kompatibilita se staršími verzemi až na výjimky, které jen potvrzují pravidlo, nemá smysl. Chytré telefony nemají dlouhou životnost, takže přístrojů s kdysi velmi rozšířenou verzí 2.2 je dnes již mizivé procento. Verze 3.0 HoneyComb byla určena jen pro tablety a přístrojů s touto verzí se v našich končinách prodalo méně než šafránu. Doporučujeme vydat se spíše cestou inovací. Každý rok je pro vývojáře v předstihu k dispozici předběžná podoba (preview) nové verze, aby si mohli nastudovat využití jejích funkcí. Tuto verzi je možné nainstalovat na novější Nexusy.

Myslíme si, že je velmi užitečné, abyste si SDK pro novou verzi Androidu nainstalovali, vytvořili si pro tyto platformy emulátory a jako vývojáři se s nimi v předstihu seznámili.

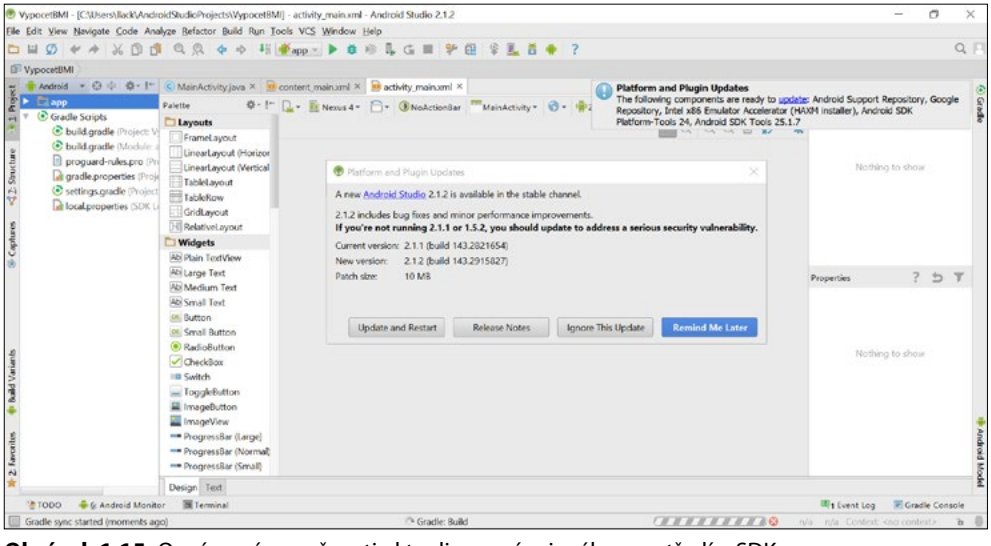
Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.9%
4.1.x	Jelly Bean	16	3.5%
4.2.x		17	5.1%
4.3		18	1.5%
4.4	KitKat	19	20.0%
5.0	Lollipop	21	9.0%
5.1		22	23.0%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	4.5%
7.1		25	0.4%



Obrázek 1.14: Procentuální zastoupení verzí Androidu (duben 2017)

Průběžná aktualizace

Pokud je k dispozici nová verze, opravný balíček, doplněné SDK, vylepšený emulátor a podobně, Android Studio vám to po spuštění oznámí a zobrazí odkazy ke stažení. Doporučujeme udržovat vývojové prostředí aktuální. Jedině tak máte jistotu, že vaše aplikace budou naplno využívat všechny možnosti operačního systému.



Obrázek 1.15: Oznámení o možnosti aktualizace vývojového prostředí a SDK

Nadpis této podkapitoly – *Průběžná aktualizace* – můžete chápat ve dvou rovinách. Udržujte aktuální nejen vývojové prostředí, ale i vaši aplikaci. Každá nová verze Androidu přinese nové a vylepšené funkce, ze kterých by vaše aplikace s vysokou pravděpodobností mohla profitovat. Proto doporučujeme tyto funkce nastudovat, a má-li to má význam, implementovat je formou aktualizace do vaší aplikace.

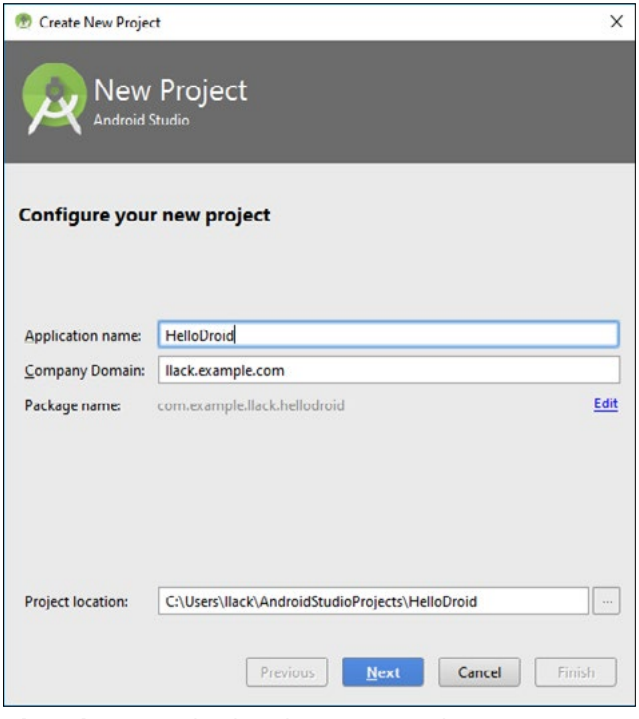
Vytvoření projektu aplikace

Možná se ptáte, proč se pouštíme do vytváření projektu mobilní aplikace dříve, než představíme aspoň v hrubých rysech architekturu operačního systému Android. Důvod je jednoduchý. Pokud se aplikace dá přeložit a spustit nejprve na emulátoru a následně na reálném zařízení, máte jistotu, že máte správně nainstalované a nakonfigurované vývojové prostředí, překladač jazyka Java, Android SDK, emulátor a propojení na reálné zařízení.

První projekt má v tomto případě i motivační význam, jelikož doslova na jedno klepnutí a bez jakéhokoliv programování vytvoříte aplikaci typu „Hello World", která vypíše na obrazovku text.

Začátečníci se v tomto příkladu nemusí snažit pochopit souvislosti. Návod na vytvoření cvičného projektu je uveden jako obrázkový postup krok za krokem.

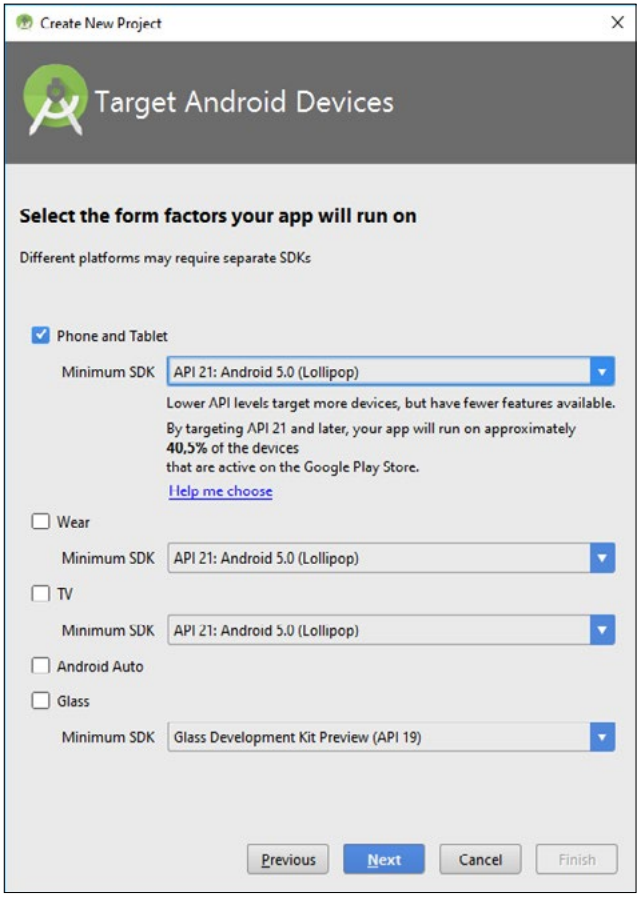
Spusťte vývojové prostředí Android Studio. V úvodním dialogu klepněte na položku **Start a new Android Studio project**. V úvodním dialogu průvodce vytvořením projektu je potřeba zadat název projektu a případně i firemní či vývojářovu doménu.



Obrázek 1.16: Android Studio Setup Wizard

Důležitý je výběr minimální verze SDK. Momentálně je nejvyšší komerčně dostupnou verzí Android 7.0 (Nougat) – verze API 24. Jako nejnižší verze je v aktuální verzi Android Studia implicitně nastavená verze 4.0.3 (IceCreamSandwich) – verze API 15. Můžete tak bez omezení používat nové prvky uživatelského rozhraní. V komentáři je uvedeno, jaké procento uživatelů využívá tuto a vyšší verze Androidu. Statistika se týká zařízení využívajících aplikační obchod Google Play. Doba životnosti mobilního telefonu je maximálně dva až tři roky, podobně je tomu i u tabletů, takže přístrojů se staršími verzemi rapidně ubývá. V našich projektech budeme používat jako nejnižší verzi Android 5.0 (Lollipop) – verze API 21. V době psaní publikace mělo tuto (a vyšší) verzi nainstalovano více než 68 procent zařízení a jejich podíl bude rychle stoupat.

Všimněte si, že můžete vytvářet projekty nejen pro telefony a tablety, ale i pro zařízení Android Wear, například hodinky, automobily, televizory a brýle Google Glass.

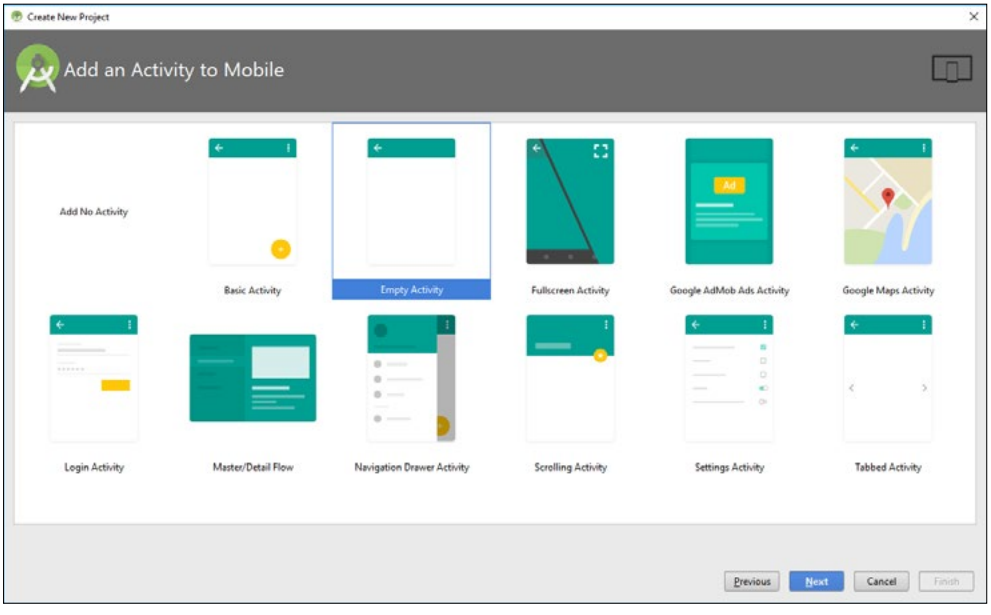


Obrázek 1.17: Výběr nejnižší podporované verze Androidu

Z projektu tedy můžete vytvořit aplikace pro dvě nebo více sesterských platforem. Vytvoření tandemu více projektů je logické. Uživatel bude mít s sebou nositelné zařízení zároveň s telefonem a tato zařízení budou mezi sebou komunikovat. Také může z telefonu nebo tabletu ovládat televizor.

Platforma Android Wear je příležitostí pro vývojáře na vytvoření užitečných informací pro nové typy zařízení. Návrh a vývoj aplikace pro zařízení, která nosíme na sobě, například hodinky, brýle a podobně, vzhledem ke specifikám těchto zařízení vyžaduje jinou filozofii než návrh aplikací pro mobilní telefony. Pokud si do tabletu nebo telefonu uživatel nainstaluje aplikaci využívající součinnost s hodinkami, část aplikace běžící v hodinkách se do nich nainstaluje automaticky.

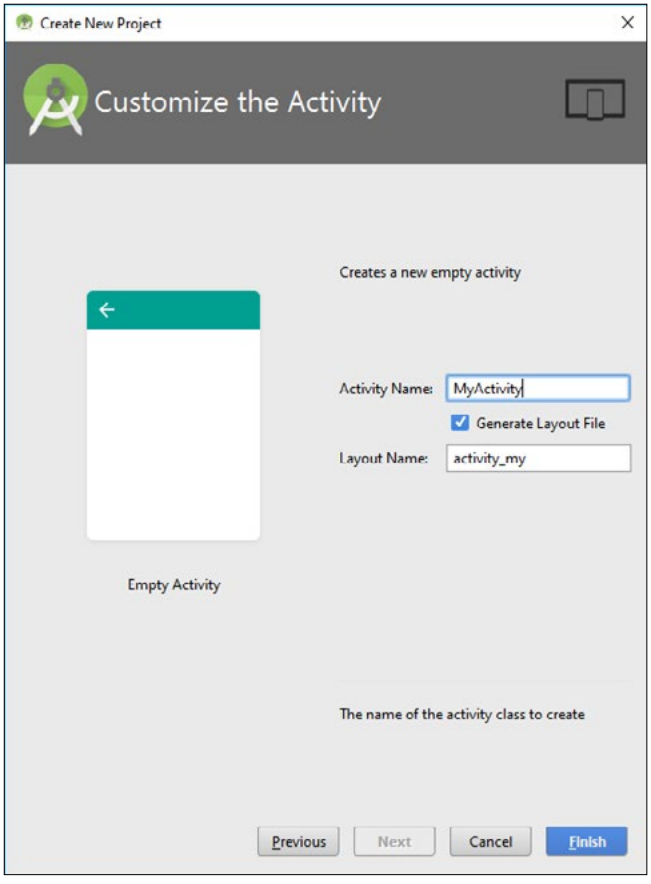
V dalším dialogu můžete zvolit několik typů pro hlavní aktivity aplikace včetně mapové či aktivity typu master-detail. Další postup se odvíjí od nastaveného typu hlavní aktivity. Například pokud jste zvolili typ master-detail, systém vás vyzve k definování názvu položky a skupiny položek. V našem příkladu se spokojíme s implicitně vybranou volbou **Empty Activity**.



Obrázek 1.18: Výběr typu hlavní aktivity

V následujícím dialogovém okně **Customize the Activity** můžete nakonfigurovat detaily pro vybraný typ aktivity. V poli **Activity Name** změňte název hlavní aktivity na `MyActivity`. Ponechejte označenou volbu **Generate Layout File** a klepnutím na tlačítko **Finish** vytvořte projekt.

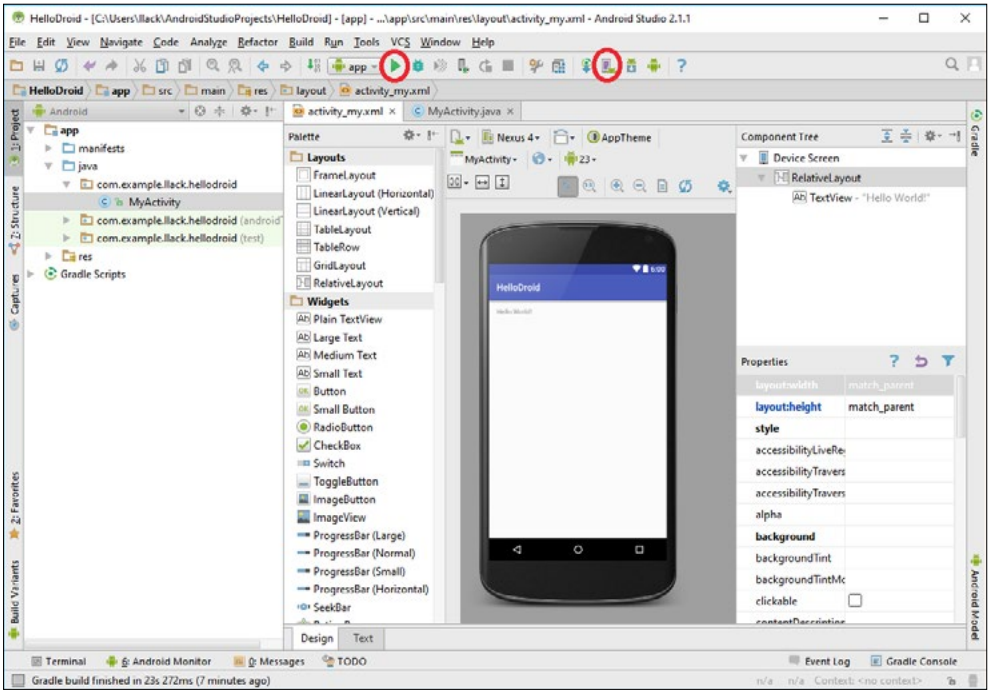
Než se vytvoří všechny potřebné soubory tvořící projekt, bude se několik sekund zobrazovat pouze šedé okno Android Studia s ukazatelem průběhu v dolní části a odhadem času, kolik sekund ještě příprava zabere. Pokud jste dosud nepovolili firevall pro Javu, zobrazí se vám dialog, ve kterém to můžete provést.



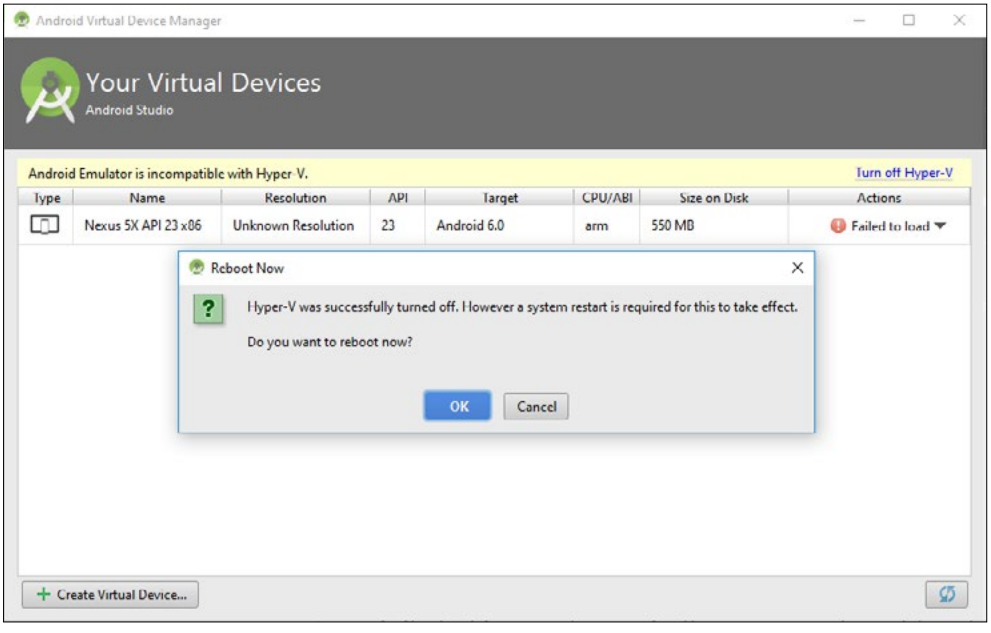
Obrázek 1.19: Vytvoření hlavní aktivity

Potom se zobrazí pracovní obrazovka včetně vizuálního návrhu uživatelského rozhraní aplikace. Zatím bude mít aplikace pouze jedinou funkcionalitu – zobrazí text „Hello World“. S anatomií projektu se seznámíte v dalších kapitolách. V této fázi se pokusíte projekt spustit, nejprve na emulátoru a následně na reálném zařízení.

Klepnutím na zelenou šipku můžete aplikaci spustit v emulátoru mobilního zařízení. V dialogu **Device Chooser** ponechejte označenou volbu **Launch Emulator**. V poli **Android virtual device** bude implicitně nastavený emulátor zařízení Nexus 4. Ponechejte tuto implicitní volbu. Pokud jste vytvořili více emulátorů, například emulátor zařízení typu chytrý telefon a emulátor zařízení typu tablet, vyberte nejvhodnější emulátor pro daný příklad. Některé emulátory nejsou kompatibilní s hypervizorem Hyper-V, který je součástí novějších verzí Windows. V tom případě budete muset klepnutím na odkaz hypervizor vypnout a vývojářský počítač restartovat.

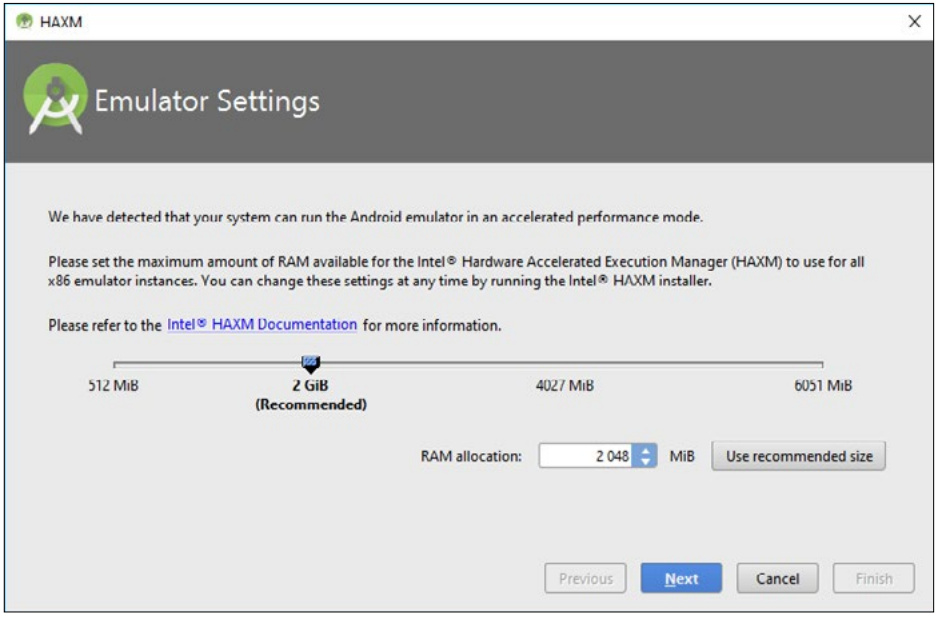


Obrázek 1.20: Nový projekt v Android Studiu. Označené jsou ikony na spuštění aplikace a ikona AVD manageru na konfiguraci emulátorů



Obrázek 1.21: Oznámení o nutnosti vypnutí Hyper-V

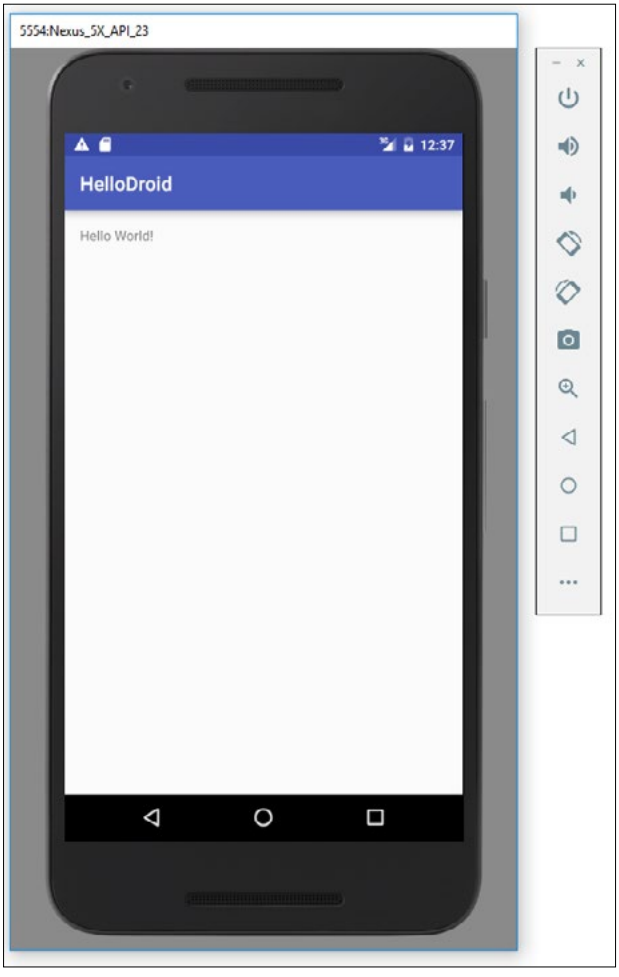
Po restartu vynuceném vypnutím hypervizoru Hyper-V bude potřeba nainstalovat Intel Hardware Accellerated Execution Manager (HAXM) a vyhradit mu adekvátní část paměti RAM. V našem případě měl počítač instalovaných 8 GB RAM a pro HAXM jsme vyhradili doporučenou velkost 2 GB.



Obrázek 1.22: Instalace Intel Hardware Accellerated Execution Manager (HAXM)

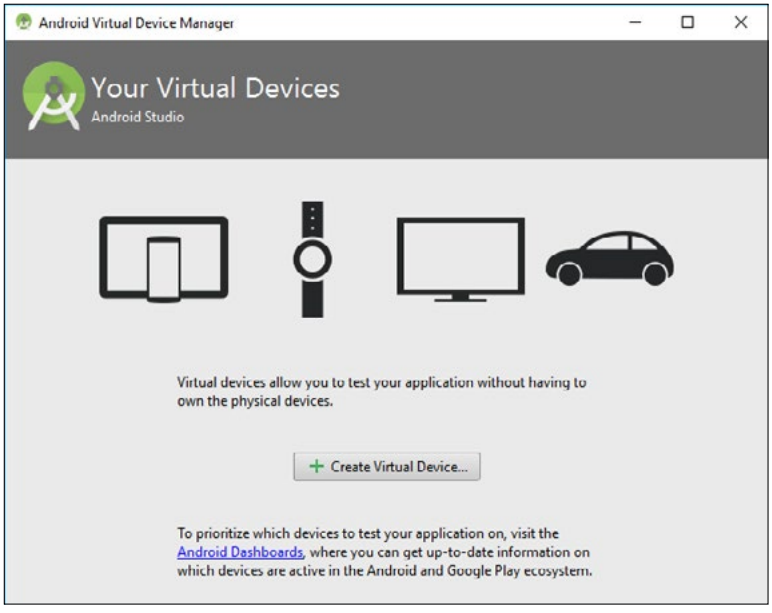
Tento úkon, tedy vypnutí Hyper-V a instalace HAXM, se samozřejmě provádí pouze jednou. Proto jsme zařadili projekt typu Hello World do kapitoly o instalaci, jelikož až po přeložení, linkovaní, spuštění na emulátoru a následně i spuštění projektu na reálném zařízení máte jistotu, že vývojové prostředí, SDK a emulátory jsou správně nainstalované a nakonfigurované.

Po náběhu emulátoru a jeho odemknutí se vaše aplikace automaticky spustí. V okně **Console** v dolní části pracovní plochy můžete sledovat průběh sestavení projektu a jeho zavedení do emulátoru.

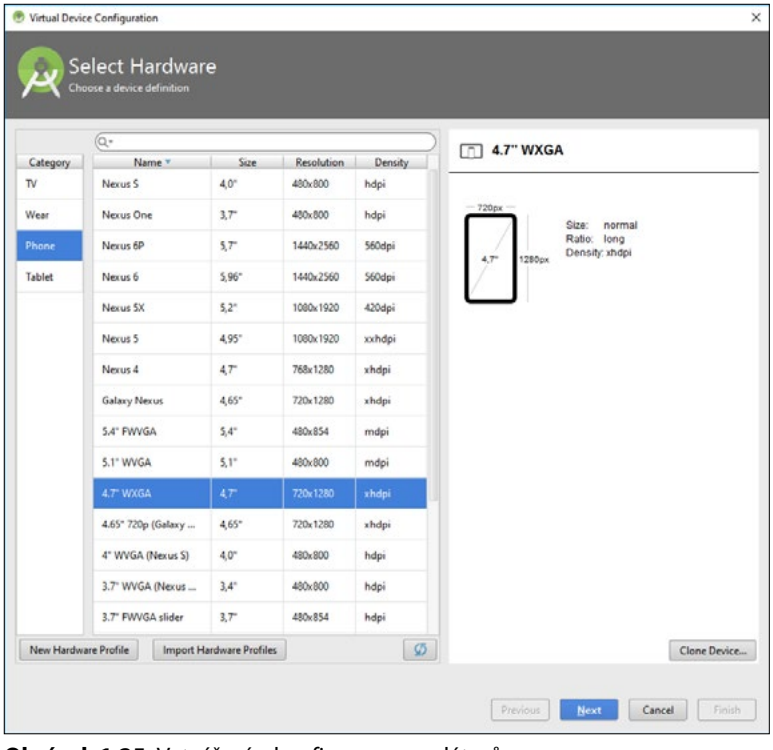


Obrázek 1.23: Spuštění aplikace v emulátoru

Pokud by se vám nepodařilo spustit implicitně nastavený emulátor, například pokud nemáte na vývojářském počítači dostatek operační paměti, můžete si nakonfigurovat jiný emulátor s menšími nároky na hardwarové zdroje. V Android Studiu klepněte na ikonu **AVD manager** (čtvrtá zleva). Zobrazí se **Android Virtual Device Manager**.



Obrázek 1.24: Android Virtual Device Manager

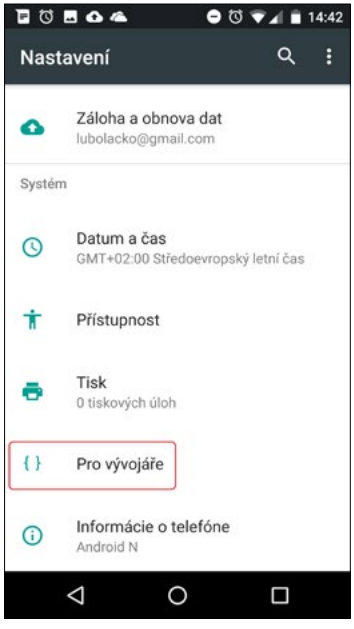


Obrázek 1.25: Vytváření a konfigurace emulátorů

Odemknutí zařízení pro vývoj

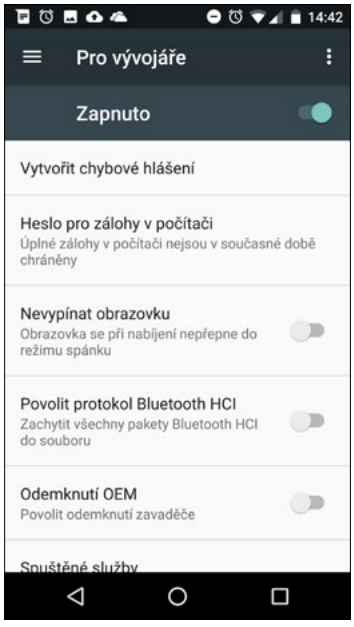
Abyste mohli vytvářet a spouštět aplikace na svém zařízení (bez ohledu na to, zda se jedná o telefon nebo tablet) musíte nejdříve aplikovat následující postup. V aplikaci **Nastavení** přejděte do nabídky **Informace o zařízení** a následně vyhledejte položku **Číslo sestavení**. U některých nadstaveb Androidu je tato položka vnořená na obrazovce **Informace o softwaru**. Další postup se vám možná bude zdát aprílový, ve skutečnosti však není. Klepejte postupně na položku **Číslo sestavení**, musíte klepnout až sedmkrát. Pokud například po šestém klepnutí přestanete, Android vás povzbudí, že se jedná o seriózní postup, aby se z vás stal vývojář, a zobrazí vám oznámení „Jste pouhé 1 klepnutí od toho, abyste se stali vývojářem", jež avizuje, kolikrát je potřeba ještě klepnout. Po posledním klepnutí se zobrazí oznámení „Nyní jste vývojářem" a v **Nastavení** přibude položka **Pro vývojáře**.

Obrázek 1.26: Postup odemknutí položky Pro vývojáře nalezení položky Číslo sestavení



Obrázek 1.27: Položka Nastavení Pro vývojáře po odemknutí

Proč takový neobvyklý postup? Na zařízeních s operačním systémem Android 4.2 a vyšším je položka **Pro vývojáře** implicitně skryta a musíte ji nejprve zobrazit. Vývojářský režim totiž umožňuje přímé zavedení aplikace z vývojářského počítače do zařízení s Androidem přes USB a to může být v případě aplikace z neznámého až podezřelého zdroje značně riskantní. V případě vývoje vlastní aplikace nic neriskujete, víte přesně, jakou aplikaci jste vytvořili a co bude dělat.

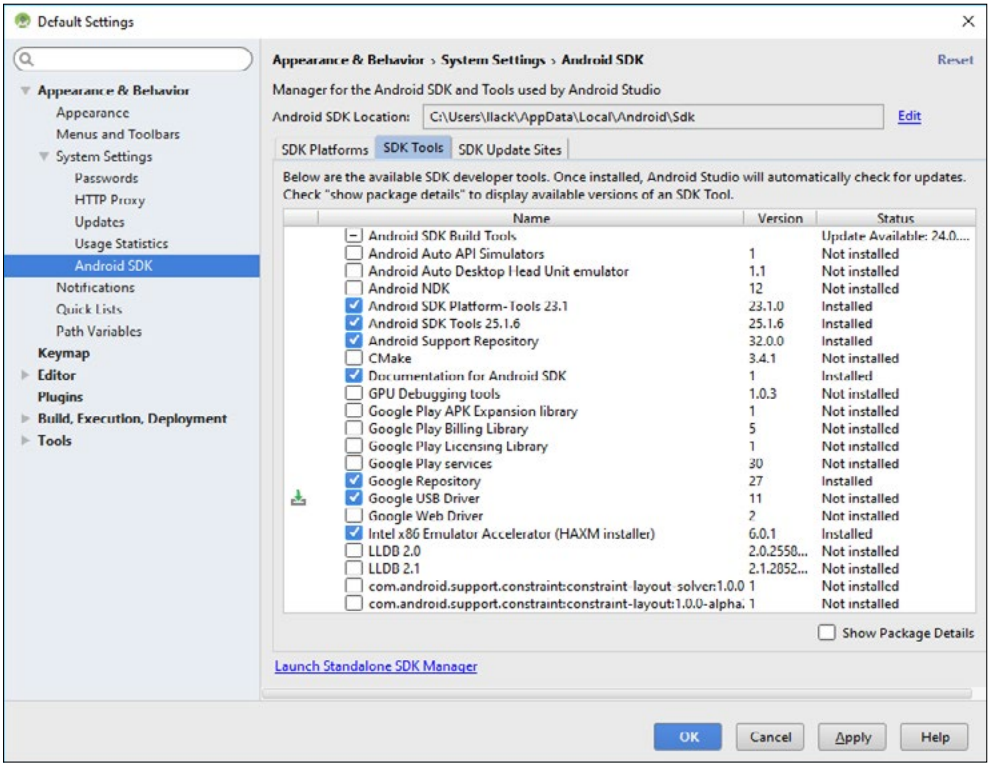


Obrázek 1.28: Seznam voleb, které můžete nastavit v nabídce Pro vývojáře

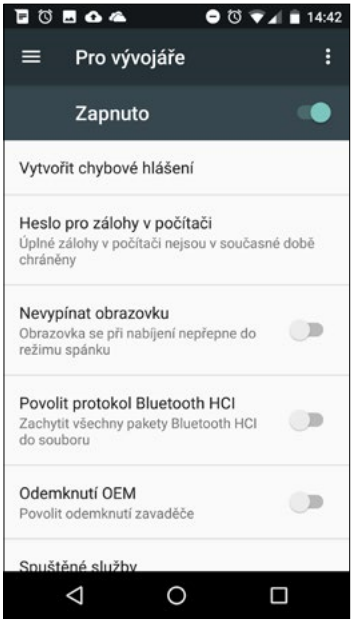
Na straně počítače potřebujete pro komunikaci se zařízením s Androidem USB ovladače pro ADB (Android Debug Bridge). Buď použijete ovladač Google USB Driver, který je součástí Android SDK, nebo ovladač dodaný výrobcem zařízení.

Spuštění aplikace na reálném zařízení

Aby bylo možné aplikaci spustit na reálném zařízení připojeném přes USB, je potřeba na vývojářském počítači nainstalovat USB ovladače pro ADB (Android Debug Bridge). Potom stačí připojit zařízení, které má povolené ladění přes USB. Pro zařízení Nexus, případně některé další, postačí Google USB Driver, který doinstalujete přes **SDK Manager** na kartě **SDK Tools**. Pro ostatní zařízení je potřeba ovladač doinstalovat ze stránky výrobce. Některá zařízení (Lenovo, Huawei apod.) mají možnost po připojení zařízení přes USB nastavit, aby se zařízení chovalo jako virtuální CD ROM, na kterém jsou ovladače.

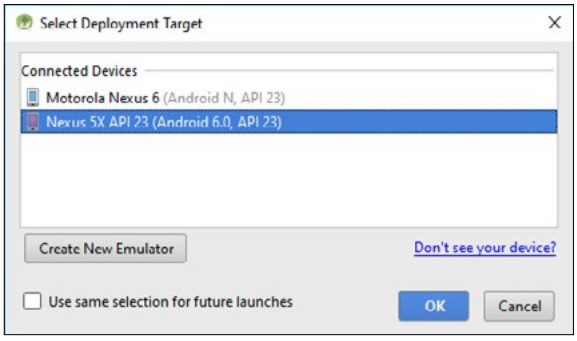


Obrázek 1.29: Instalace ovladače Google USB Driver přes SDK Manager



Obrázek 1.30: Zapnutí ladění přes USB na připojeném zařízení

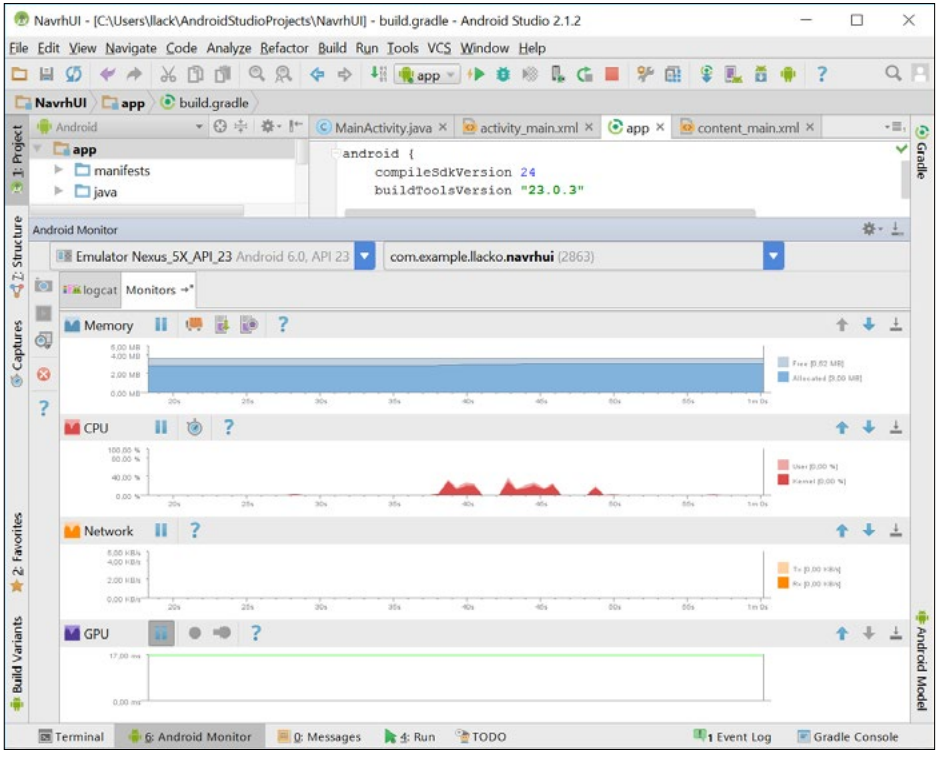
Po správném nakonfigurování USB ovladače pro ADB se při pokusu o spuštění aplikace zobrazí nabídka možností.



Obrázek 1.31: Výběr hardwarového zařízení ke spuštění aplikace

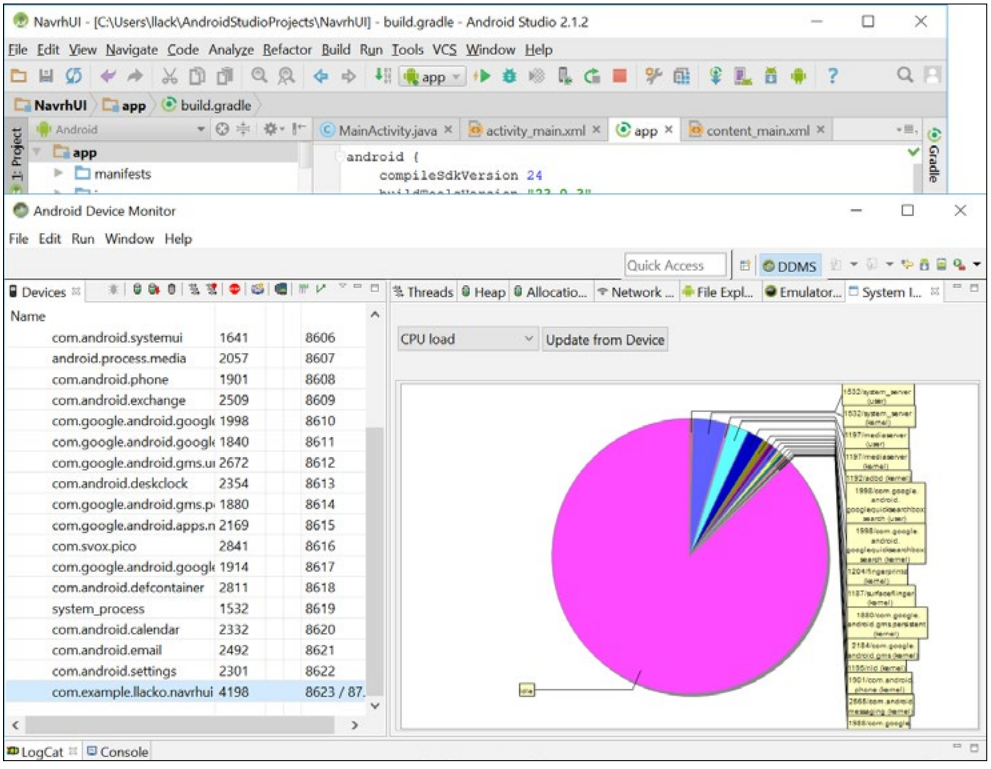
Monitorování spuštěné aplikace

Hodně zajímavých informací se dozvíte i za běhu aplikace. Po spuštění aplikace – ať už v emulátoru, nebo na reálném zařízení připojeném přes USB – se v dolní části plochy vývojového prostředí Android Studio zobrazí okno **Android Monitor**. Okno má dvě záložky: **LogCat**, kde se vypisují události, ať už implicitní, nebo události, které do protokolu zapisujete sami. Příklad najdete v části o životních cyklech aktivity. Na druhé záložce **Monitor** můžete například sledovat, jak vaše aplikace zatěžuje procesor, GPU, paměť, či síťové připojení.



Obrázek 1.32: Monitorování spuštěné aplikace v okně Android Monitor

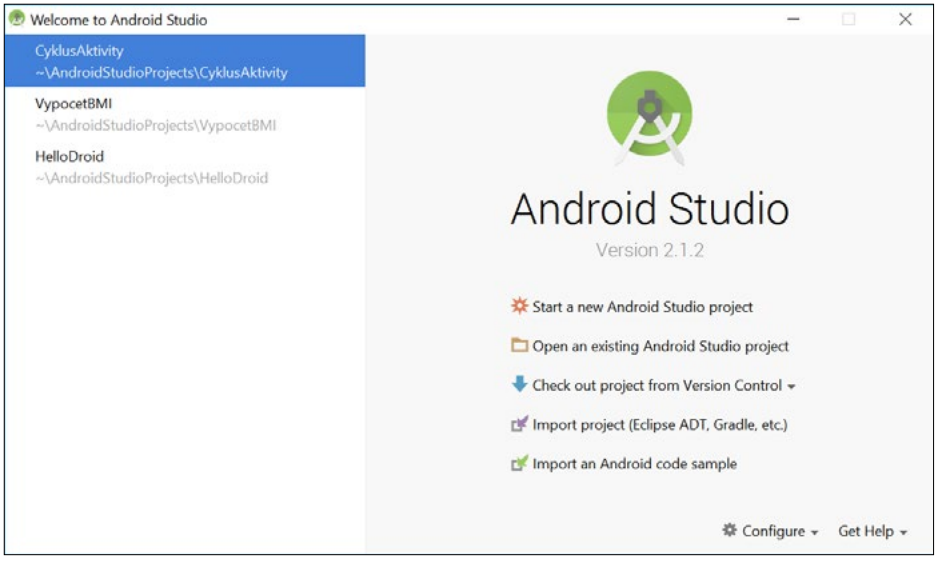
K monitorování spuštěné aplikace můžete využít i **Android Device Monitor**. Z Android Studia tento nástroj spustíte pomocí nabídky **Tools → Android → Android Device Monitor**.



Obrázek 1.33: Monitorování spuštěné aplikace pomocí nástroje Android Device Monitor

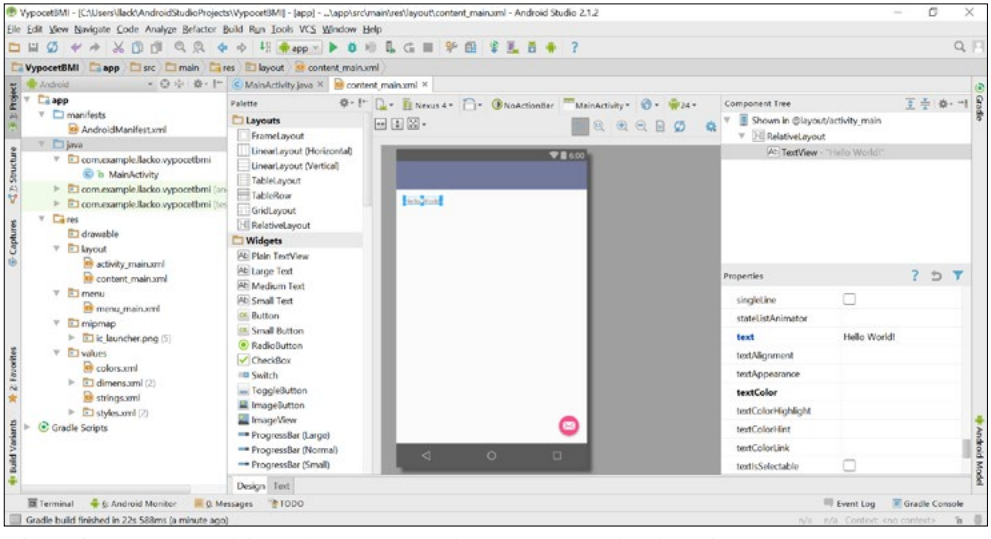
Seznámení se s vývojovým prostředím

Po spuštění vývojového prostředí Android Studio se zobrazí uvítací dialog s nabídkou nejčastěji prováděných činností. Můžete vytvořit nový projekt, případně otevřít existující, rozpracovaný projekt. Pro vývojáře, kteří dosud používali jiné vývojové prostředí, například Eclipse + ADT, je zajímavá možnost přímého importu projektů z dosud používaných vývojářských nástrojů do Android Studia. Také můžete importovat a testovat různé vzorové příklady. Zároveň tento dialog poskytuje informaci o aktuální verzi Android Studia a nabízí možnost konfigurace jeho součástí.



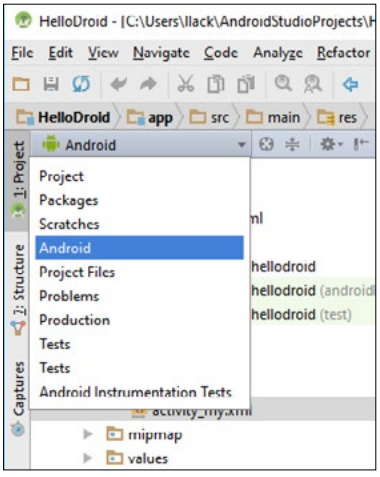
Obrázek 1.34: Úvodní dialog vývojového prostředí Android Studio

Rozmístění pracovních oken vývojového prostředí je na obrázku. Jejich velikost, přesněji poměry velikosti, lze podle potřeby měnit.



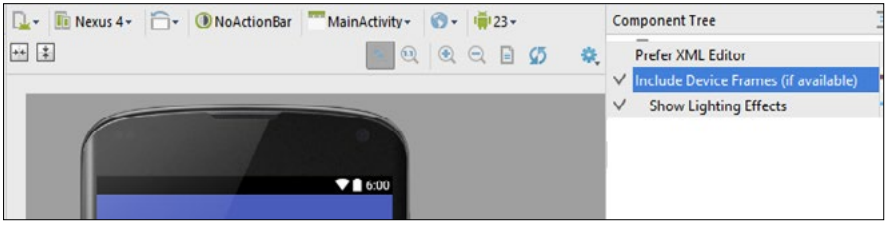
Obrázek 1.35: Uživatelské rozhraní vývojového prostředí Android Studio

V levém svislém podlouhlém okně se zobrazuje struktura projektu uspořádaná přehledně hierarchicky. Detailního zobrazení struktury projektu docílíte nastavením volby **Android**.



Obrázek 1.36: Možnosti zobrazení struktury projektu

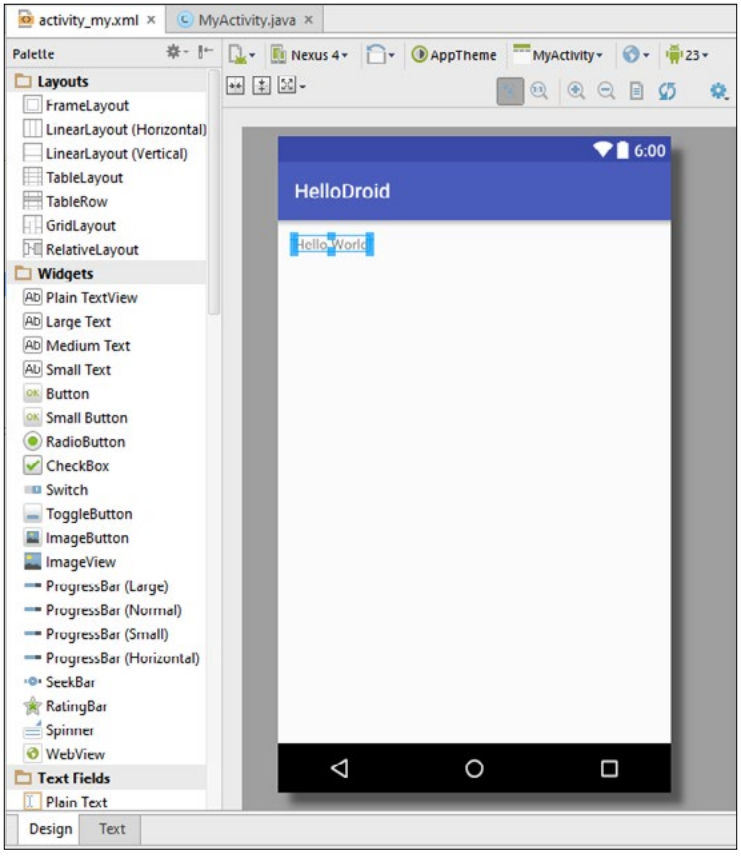
Dominantním při návrhu uživatelského rozhraní je náhledové okno. Po nainstalování se zobrazuje i s rámečkem znázorňujícím tělo telefonu. Na vývojářských noteboocích s menšími displeji je to tak trochu plýtvání místem. Tento rámeček lze ale naštěstí vypnout. Vpravo nahoře na liště okna najdete ikonu s ozubeným kolečkem, která zobrazí nabídku nastavení. V této nabídce zrušte zaškrtnutí políčka **Include Device Frames**. Potom se bude zobrazovat pouze displej, takže budete moct zobrazit větší uživatelské rozhraní.



Obrázek 1.37: Nastavení zobrazení – možnost vypnutí rámečku

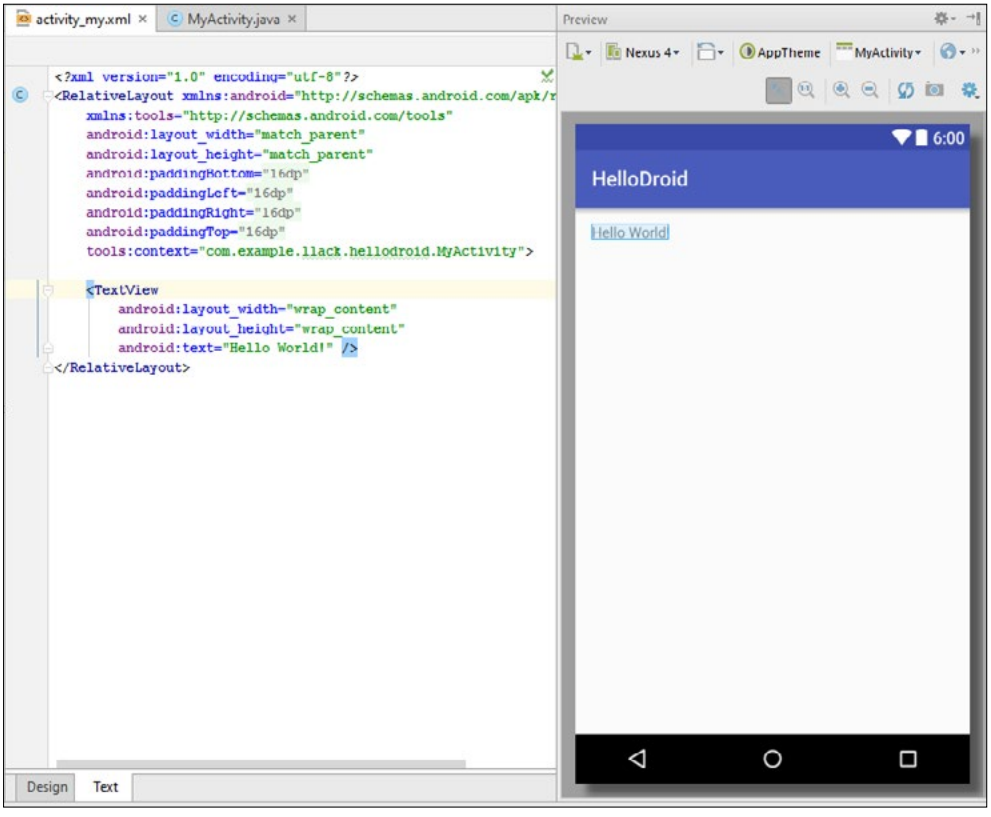
Uživatelské rozhraní můžete navrhnout ve dvou módech: **Design** a **Text**. Přepínají se pomocí karet ve spodní části okna.

V režimu **Design** je vlevo vedle okna s návrhem uživatelského rozhraní ve sloupcovém okně **Palette** zobrazena paleta komponent, uprostřed vizuální reprezentace uživatelského rozhraní a vpravo hierarchie komponent a vlastnosti vybrané komponenty. Komponenty z palety můžete přesouvat na plochu aplikace a vhodně rozmístit.



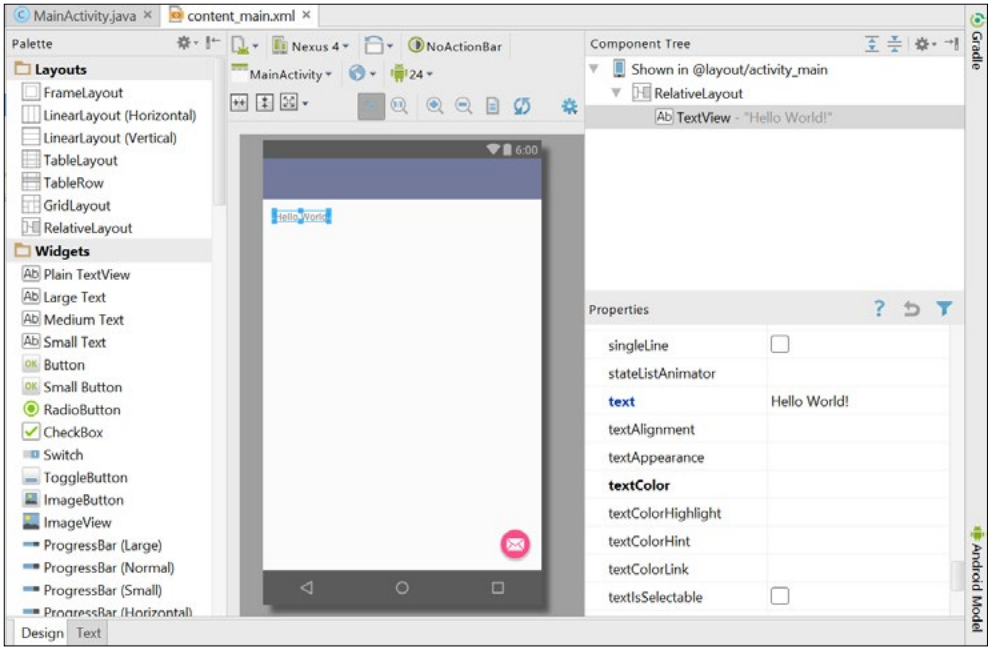
Obrázek 1.38: Návrh uživatelského rozhraní – mód Design

V režimu **Text** se vlevo zobrazuje kód XML definice návrhu uživatelského rozhraní a vpravo jeho vizuální reprezentace. Změny zrealizované v návrhovém kódu se automaticky průběžně promítají do vizuálního zobrazení.



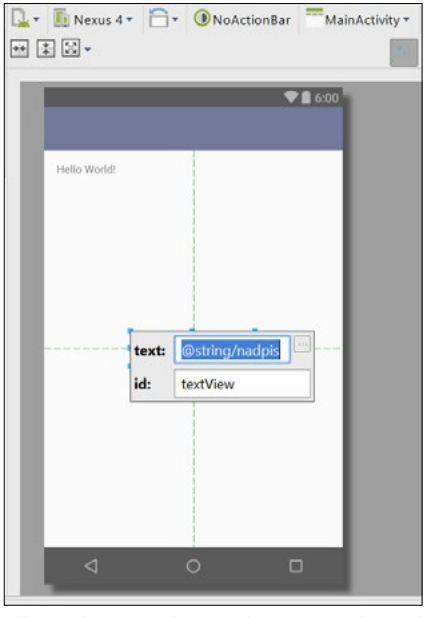
Obrázek 1.39: Návrh uživatelského rozhraní – mód Text

V pravé části pracovní obrazovky Android Studia najdete ještě dvě důležitá okna. V okně **Component Tree** je hierarchická struktura prvků uživatelského rozhraní. Toto okno poskytuje perfektní přehled, jak jsou prvky hierarchicky zapouzdřeny ve vizuálních kontejnerech, případně jak jsou kontejnery zapouzdřeny navzájem. V okně **Properties** se zobrazují vlastnosti vybraného prvku.



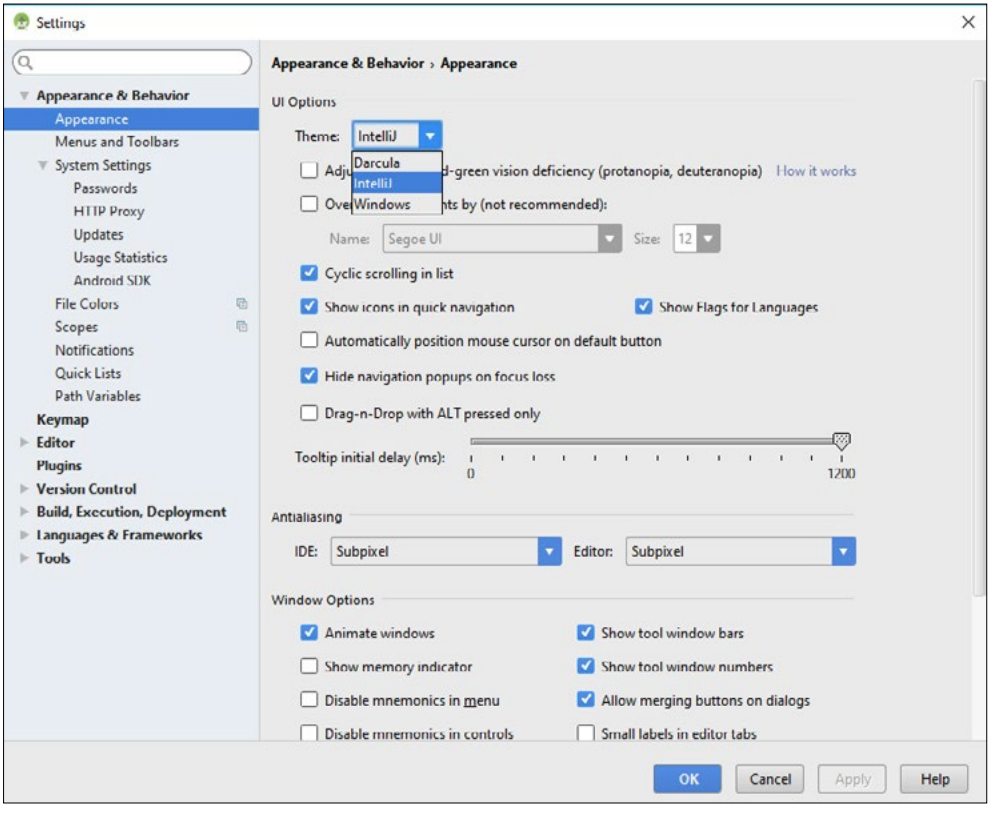
Obrázek 1.40: Vlastnosti vybraného prvku v okně Properties

Klíčovou vlastnost vybraného prvku – například v případě prvku `EditText` je to implicitně definovaný textový řetězec – zobrazíte poklepáním na prvek v návrhovém zobrazení.



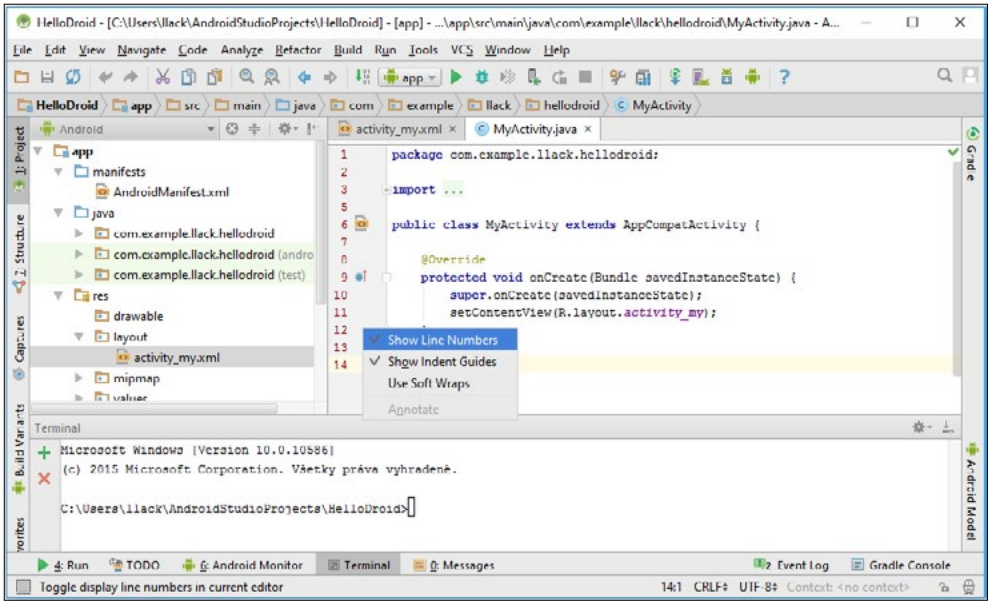
Obrázek 1.41: Klíčové vlastnosti vybraného prvku

Pokud vám více vyhovuje tmavý barevný design uživatelského rozhraní vývojového prostředí Android Studio, pomocí nabídky **File → Settings** zobrazte dialogové okno nastavení a na kartě **Appearance** nastavte položku **Theme** na hodnotu **Darcula**.



Obrázek 1.42: Změna barevného schématu vývojového prostředí Android Studio

U zobrazení zdrojového kódu v programovacím jazyce Java můžete pomocí místní nabídky zapnout zobrazování čísel řádků. Pomůže vám to například při identifikaci a lokalizaci případné chyby.

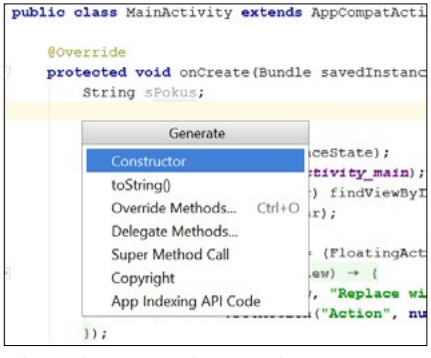


Obrázek 1.43: Zobrazení zdrojového kódu v programovacím jazyce Java

Pro některé často opakované úkony nabízí prostředí klávesové zkratky. Jejich seznam získáte v nabídce **Help** pod položkou **Default Keymap Reference**. Zobrazí se seznam zkratek ve formátu PDF, který si případně můžete vytisknout a umístit na vhodné místo jako tahák.

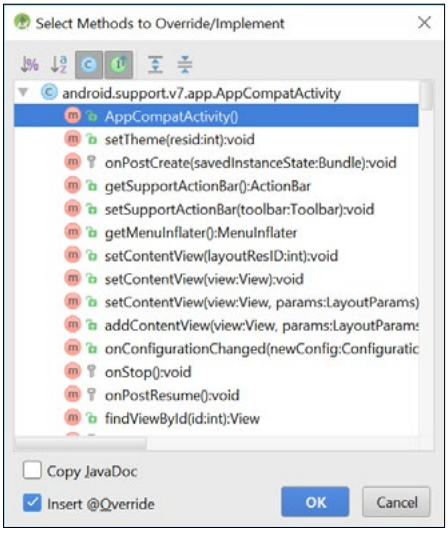
Editor kódu v jazyce Java

Vývojové prostředí Android Studio se snaží co nejvíce zjednodušit a zefektivnit psaní kódu. Pomocí místní nabídky a položky **Generate** nebo klávesové zkratky **Alt+Insert** se zobrazí automatické generování bloků kódu.



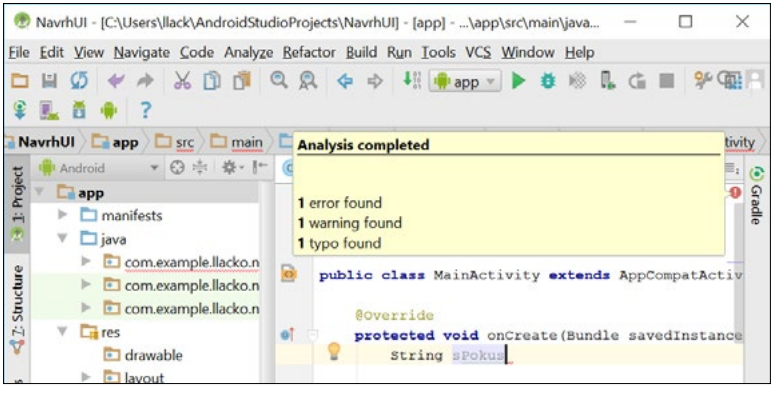
Obrázek 1.44: Dialogové okno s automatickým generováním bloků kódu

Takovýmto způsobem můžete například jednoduše generovat těla kódu metod `Override`.



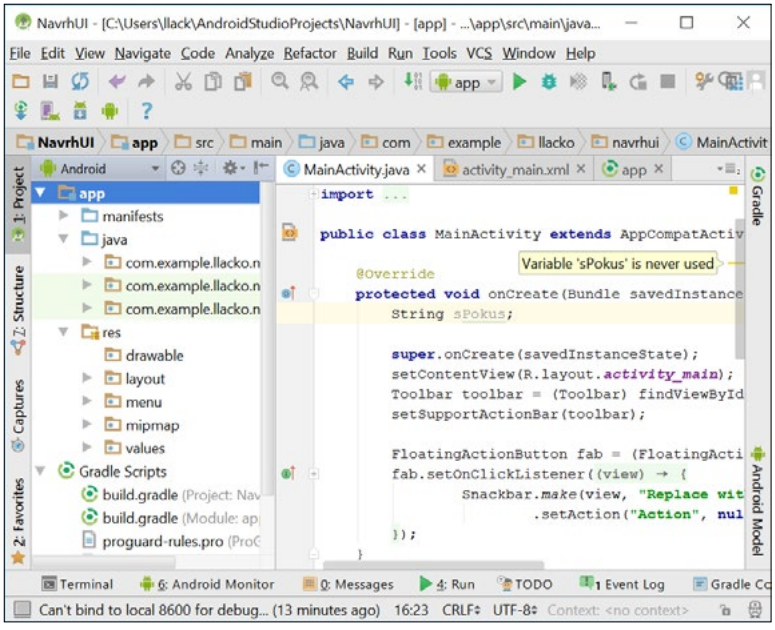
Obrázek 1.45: Nabídka metod Override

Vývojové prostředí neustále kontroluje syntaktickou správnost vašeho kódu. Pokud narazí na problém (v našem případě), chybějící středník za příkazovým řádkem, zobrazí se na liště vpravo červená ikonka s vykřičníkem. Klepnutím na ikonu otevřete nápovědu k problému.



Obrázek 1.46: Výsledek průběžné analýzy kódu

Chyby a upozornění, například na nepoužitou proměnnou, se signalizují na liště vpravo i červenými a žlutými čárkami, které jsou rozmístěny proporcionálně s řádky kódu, na nichž se tyto chyby nacházejí. Klepnutím na čárku se zobrazí podrobnosti.



Obrázek 1.47: Upozornění, v tomto případě na nepoužitou proměnnou

Java pro migranty

Nejjednodušší situaci, co se týká přechodu na programovací jazyk Java, budou mít programátoři v C#. Při zběžném pohledu na segment kódu jsou tyto jazyky téměř k nerozeznání.

Programovací jazyk Java má striktní kontrolu typů a rozsahů polí. Nezná pole znaků, text uzavřený do uvozovek se automaticky konvertuje na objekt typu `String`. Proměnné primitivních typů se automaticky inicializují na `0`, respektive příslušný ekvivalent, a reference objektů se automaticky inicializují na hodnotu `null`.

Java podporuje vícevláknový (multithread) kód s automatickým uzamykáním a dá se využít i tzv. souběžné programování. Pokud něco nevyjde podle představ vývojáře, přichází ke slovu robustní systém obsluhy výjimek. Java neumožňuje používat globální datové struktury ani funkce. Všechny programové struktury musí být zahrnuty v třídách. Namísto oboru jmen (namespace) používá balíčky (packages).

Pozitivní zprávou je, že Java nezná ukazatele, známý to postrach začátečníků v programovacím jazyce C, a nedefinuje automatické standardní (default) ani kopírovací (copy) konstruktory. Namísto destruktorů využívá Java k rušení objektů mechanizmus nazývaný garbage collector, který pomocí metody `finalize()` uvolňuje používané zdroje. Objekt se automaticky zruší v případě, že už dále nebude využíván a současně existuje více takovýchto nepoužívaných objektů.

Co se týká objektově orientovaného programování, všechny třídy jsou organizovány v jednokořenové (single-root) struktuře začínající třídou `Object`. Třídy se definují, ne deklarují. Bez specifikace konkrétního přístupu pro data a metody třídy jsou data a metody automaticky přístupné všem třídám definovaným v rámci balíčku. Využívají se i tzv. vnořené (nested) třídy, které efektivně a pro programátora jednoduše spolupracují s členy nadřazených tříd. Pokud defi-

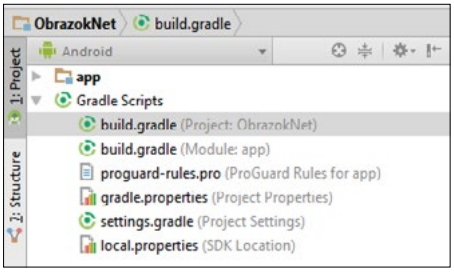
nujete přístupovou úroveň `protected`, můžete k takto definovaným členům třídy přistupovat z libovolného místa v rámci balíčku, avšak mimo něj pouze z podtříd dané třídy. Rodičovská třída je definovaná pomocí klíčového slova `extends` a metody rodičovské třídy voláme pomocí klíčového slova `super`. Toto volání se musí provést na začátku konstruktoru.

Java s výhodou používá přetěžování (override) metod. Název metody a seznam jejích parametrů se nazývá signatura. K vytvoření abstraktní třídy (rozhraní) slouží klíčové slovo `interface`. Následně k definování třídy, která implementuje abstraktní metody rozhraní, slouží klíčové slovo `implements`. K vytvoření metody, která nemůže být přetížena, slouží klíčové slovo `final`.

Gradle

Dosud jsme se nezajímali o to, jak Android Studio zkompiluje či interpretuje zdrojové a návrhové kódy, slinkuje je dohromady se zdroji, vytvoří aplikační balíček, zavede ho do emulátoru nebo reálného zařízení a spustí výslednou aplikaci. Už z předchozí věty je zřejmé, že se jedná o velké množství na sebe navazujících úloh, přičemž následující může (někdy nemusí) být zahájena až po úspěšném ukončení předchozí. Automatizaci úloh při sestavení, zavedení a spuštění aplikace v Android Studiu má na starosti moderní nástroj na automatizaci Gradle, integrovaný přímo v Android Studiu.

Gradle je nástroj na automatizaci procesů, přesněji jazyk na automatizaci. Je to jazyk typu Domain Specific Language (DSL), který umožňuje popsat posloupnost úloh, které chceme zautomatizovat. Je založen na Apache Groovy. Samotný Gradle toho moc nedokáže, jeho skutečná síla spočívá v pluginech. Na sestavování projektů aplikací pro Android se využívá Android plugin for Gradle.



Obrázek 1.48: Struktura zobrazení složky Gradle v Android Studiu

Android Studio umožňuje vytvářet současně více podprojektů, ze kterých se skládá komplexní aplikace. Každý takovýto podprojekt má vlastní *build.gradle* skript. Za názvem *build.gradle* je ještě v závorce uvedeno, že je to build soubor pro celý projekt ve tvaru **(Project: <project name>)**. Skript na nejvyšší úrovni obsahuje globální konfiguraci pro všechny podprojekty (moduly).

Příklad skriptu:

```
// Top-level build file where you can add configuration options
// common to all sub-projects/modules.

buildscript {
  repositories {
    jcenter()
  }
  dependencies {
```

```
    classpath 'com.android.tools.build:gradle:2.1.2'

    // NOTE: Do not place your application dependencies here; they belong
    // in the individual module build.gradle files
  }
}

allprojects {
  repositories {
    jcenter()
  }
}

task clean(type: Delete) {
  delete rootProject.buildDir
}
```

Soubor *settings.gradle* deklaruje jednotlivé podprojekty. Názvy modulů odpovídají názvům adresářů, ve kterých se nacházejí. V jednoduché aplikaci je v tomto souboru jeden řádek:

```
include ':app'
```

Tento soubor, označený **build.gradle (Module: app)**, obsahuje skript:

```
apply plugin: 'com.android.application'

android {
  compileSdkVersion 24
  buildToolsVersion "23.0.3"

  defaultConfig {
    applicationId "sk.pcrevue.llacko.obrazoknet"
    minSdkVersion 23
    targetSdkVersion 24
    versionCode 1
    versionName "1.0"
  }
  buildTypes {
    release {
      minifyEnabled false
      proguardFiles getDefaultProguardFile('proguard-android.txt'),
        'proguard-rules.pro'
    }
  }
}

dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  testCompile 'junit:junit:4.12'
  compile 'com.android.support:appcompat-v7:24.0.0'
}
```

V prvním řádku skriptu je deklarovaný plugin `com.android.application`. V sekci `android` jsou údaje o verzích SDK a verzích nástrojů. Následuje sekce `defaultConfig`, která obsahuje ID aplikace, minimální a cílovou verzi SDK.

```
apply plugin: 'com.android.application'

android {
  compileSdkVersion 24
  buildToolsVersion "23.0.3"
```

Sekce `buildTypes` obsahuje instrukce, jak spustit ProGuard se souborem APK. A to nejdůležitější nakonec: Sekce `dependencies` definuje vzájemné závislosti modulů, které se budou kompilovat a sestavovat.

Graddle můžete spustit i z příkazového řádku Windows. Příkazem `CD "cesta"`, například

```
CD C:\BT android\ObrazokNet
```

přejdete do adresáře, ve kterém máte projekt vytvořený v Android Studiu a zadáte příkaz

```
> gradlew.bat assembleRelease
```

V okně konzole můžete sledovat průběh sestavení. Při prvním spuštění se budou navíc stahovat a rozbalovat potřebné moduly.

```
C:\BT android\ObrazokNet>gradlew.bat assembleRelease
Incremental java compilation is an incubating feature.
:app:preBuild UP-TO-DATE
:app:preReleaseBuild UP-TO-DATE
:app:checkReleaseManifest
:app:preDebugBuild UP-TO-DATE
:app:prepareComAndroidSupportAnimatedVectorDrawable2400Library UP-TO-DATE
:app:prepareComAndroidSupportAppcompatV72400Library UP-TO-DATE
:app:prepareComAndroidSupportSupportV42400Library UP-TO-DATE
:app:prepareComAndroidSupportSupportVectorDrawable2400Library UP-TO-DATE
:app:prepareReleaseDependencies
:app:compileReleaseAidl UP-TO-DATE
:app:compileReleaseRenderscript UP-TO-DATE
:app:generateReleaseBuildConfig UP-TO-DATE
:app:mergeReleaseShaders UP-TO-DATE
:app:compileReleaseShaders UP-TO-DATE
:app:generateReleaseAssets UP-TO-DATE
:app:mergeReleaseAssets UP-TO-DATE
:app:generateReleaseResValues UP-TO-DATE
:app:generateReleaseResources UP-TO-DATE
:app:mergeReleaseResources UP-TO-DATE
:app:processReleaseManifest UP-TO-DATE
:app:processReleaseResources UP-TO-DATE
:app:generateReleaseSources UP-TO-DATE
:app:incrementalReleaseJavaCompilationSafeguard UP-TO-DATE
:app:compileReleaseJavaWithJavac UP-TO-DATE
:app:compileReleaseNdk UP-TO-DATE
:app:compileReleaseSources UP-TO-DATE
```

```
:app:lintVitalRelease
:app:prePackageMarkerForRelease
:app:transformClassesWithDexForRelease UP-TO-DATE
:app:mergeReleaseJniLibFolders UP-TO-DATE
:app:transformNative_libsWithMergeJniLibsForRelease UP-TO-DATE
:app:processReleaseJavaRes UP-TO-DATE
:app:transformResourcesWithMergeJavaResForRelease UP-TO-DATE
:app:packageRelease UP-TO-DATE
:app:assembleRelease

BUILD SUCCESSFUL
Total time: 14.428 secs
C:\BT android\ObrazokNet>
```

Možnosti současného hardwaru

Hodně uživatelů ani netuší, jaký obrovský výpočetní a často i grafický výkon nosí v kapsách. Zkuste se přenést o 20 let zpět, kdy se začaly do počítačů osazovat procesory Intel Pentium. Byly taktované na 60 MHz, ty nejdražší dosahovaly tehdy až neuvěřitelných 75 MHz. Nejvýkonnější stroje měly k dispozici 64 MB paměti RAM.

A nyní to porovnejte s moderním smartphonem střední třídy. Například Huawei P8 Lite – telefon v cenové hladině 5 000 Kč, tedy žádná vlajková loď ani horká novinka, v době psaní knihy byl už rok na trhu – má osmijádrový procesor s taktem 1,2 GHz, 2 GB paměti RAM a 16 GB úložného prostoru. Dokázali jste si v éře prvních Pentií vůbec představit počítač, natož pak levný mobilní telefon, který bude mít dvacetkrát výkonnější procesor (reálně mnohem víc, protože první Pentium mělo jen jedno jádro a procesory mobilů mají 4 až 8) a třicetkrát více paměti? Stejně neslavně dopadne špičkový počítač nedávné minulosti v porovnání s inteligentními hodinkami. Například hodinky Samsung Gear S2 mají dvoujádrový procesor taktovaný na 1,2 GHz, 512 MB RAM a úložný prostor s kapacitou 4 GB.

Proč tyto věci zmiňujeme? Velký výkon mobilních zařízení je totiž zároveň i velkou výzvou pro vývojáře aplikací, aby tento výkon dokázali využít a poskytnout uživateli své aplikace co nejvyšší přidanou hodnotu. Představte si, že byste před 20 roky, ve zlaté éře PC, kdy pro ně nebyl dostatek kvalitních aplikací a her, položili IT expertovi otázku: Co by dokázalo zařízení, které by bylo stokrát výkonnější a mělo stonásobně více operační paměti a úložného prostoru a navíc by se vešlo do kapsy a minimálně den by vydrželo fungovat na baterii? Odpovědi, či spíše předpovědi, by určitě předstihly možnosti 99,99 % současných mobilních aplikací. Je to tedy obrovská výzva pro vývojáře.

Referenční zařízení

Možná jste už slyšeli o zařízeních s označením Nexus. Produktová rodina Nexus má mezi zařízeními s Androidem výsadní postavení referenčního etalonu. Navrhuje je Google ve spolupráci s některými renomovanými dodavateli telefonů, kteří je následně vyrábí. Nové modely se tradičně představují na konferenci Google I/O spolu s novým operačním systémem. V září 2015 byl představen tandem Nexus 5X od LG a větší Nexus 6P, který vyrábělo Huawei.

Jejich pozice etalonu má svá specifika a zaběhlá pravidla, která se týkají i designu. Dobrou analogií jsou etalony pro základní fyzikální veličiny. Jsou velmi kvalitní, ale bez jakýchkoliv designo-

vých okras. A stejné jsou i Nexusy. Strohost jejich designu má i jiný důvod. Google s nimi nechce konkurovat modelům ostatních výrobců. U Nexusů je zároveň pravidlem, že disponují nejmodernějšími technologiemi, aby na nich vývojáři mohli testovat aplikace, které budou tyto technologie využívat. Nové modely Nexusů mají například porty USB typu C a snímače otisků prstů.

Abychom byli konkrétní, Nexus 5X, který vyrábí LG, má šestijádrový procesor disponující čtyřmi jádry Cortex-A53 pro běžný úsporný provoz a dvěma jádry Cortex-A57 pro zvládnutí špičkové zátěže. K dispozici má 2 GB RAM. Jako grafický čip je použit Adreno 418. Fotoaparát umožňuje natáčet video s rozlišením 4 K.

Jednu z nejatraktivnějších vlastností Nexusů jsme si nechali na konec. Po ohlášení nové preview verze Androidu jsou to vždy právě Nexusy, pro které je tato verze k dispozici. Takže jejich majitelé se mohou těšit, že budou mít nový Android s kódovým označením „N“, případně později s označením „O“, jako první. Řadu telefonů Nexus jako referenčních zařízení pro vývoj a testování aplikací Google ukončil a nahradil novou produktovou řadou špičkových zařízení Google Pixel.

Technické údaje Nexusu 5X:

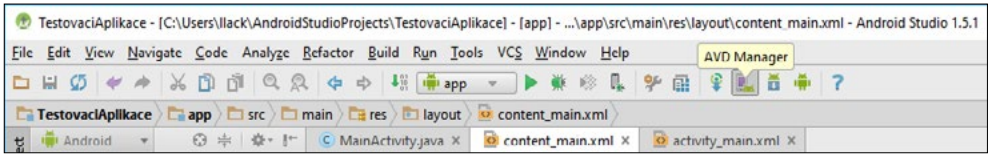
Procesor a grafika:	Šestijádrový Qualcomm Snapdragon 808, 1,8 GHz, GPU Adreno 418
Paměť:	2 GB RAM, úložný prostor 16 GB nebo 32 GB
Displej:	5,2“ IPS s rozlišením 1 920 × 1 080, 423 ppi
Fotoaparát:	Zadní: 12,3 MP, Přední: 5 MP, oba se světelností f/2.0
Konektivita:	LTE, Wi-Fi 802.11ac, USB-C, Bluetooth 4.2, NFC
Baterie:	2700 mAh, nevyměnitelná
Operační systém:	Android 6.0 Marshmallow
Rozměry a hmotnost:	147 × 72,6 × 7,9 mm, 136 g

Obrázek 1.49: Referenční telefon NEXUS 5X

Vytvoření emulátoru

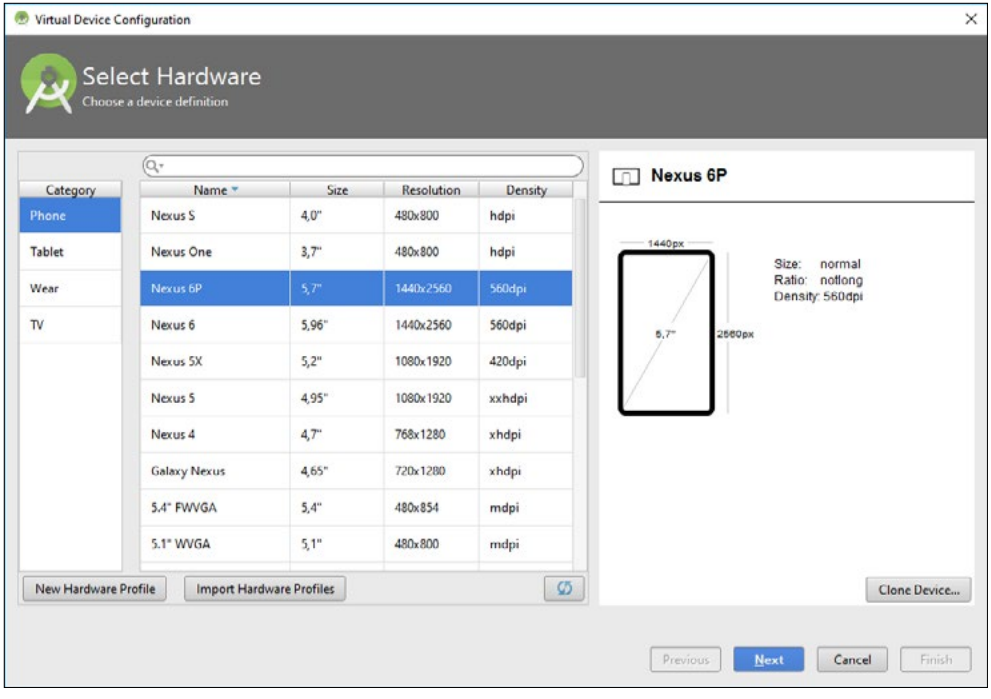
Zdálo by se, že pokud máte k dispozici jeden moderní chytrý telefon a jeden tablet s Androidem, dokážete pokrýt návrh, testování a ladění aplikací. Opak je pravdou. Výhodou a zároveň nevýhodou Androidu je variabilita různých zařízení s různými verzemi systému a různým rozlišením displeje. Neodmyslitelnou pomůckou vývojáře je proto emulátor, na kterém je možné otestovat aplikaci ve více verzích operačního systému Android a na obrazovkách s různým rozlišením.

V aplikaci Android Studio na nástrojové liště vyberte ikonu **AVD Manager**. Zkratka AVD znamená Android Virtual Devices. Zobrazí se dialogové okno aplikace **Android Virtual Device Manager** se seznamem virtuálních zařízení.



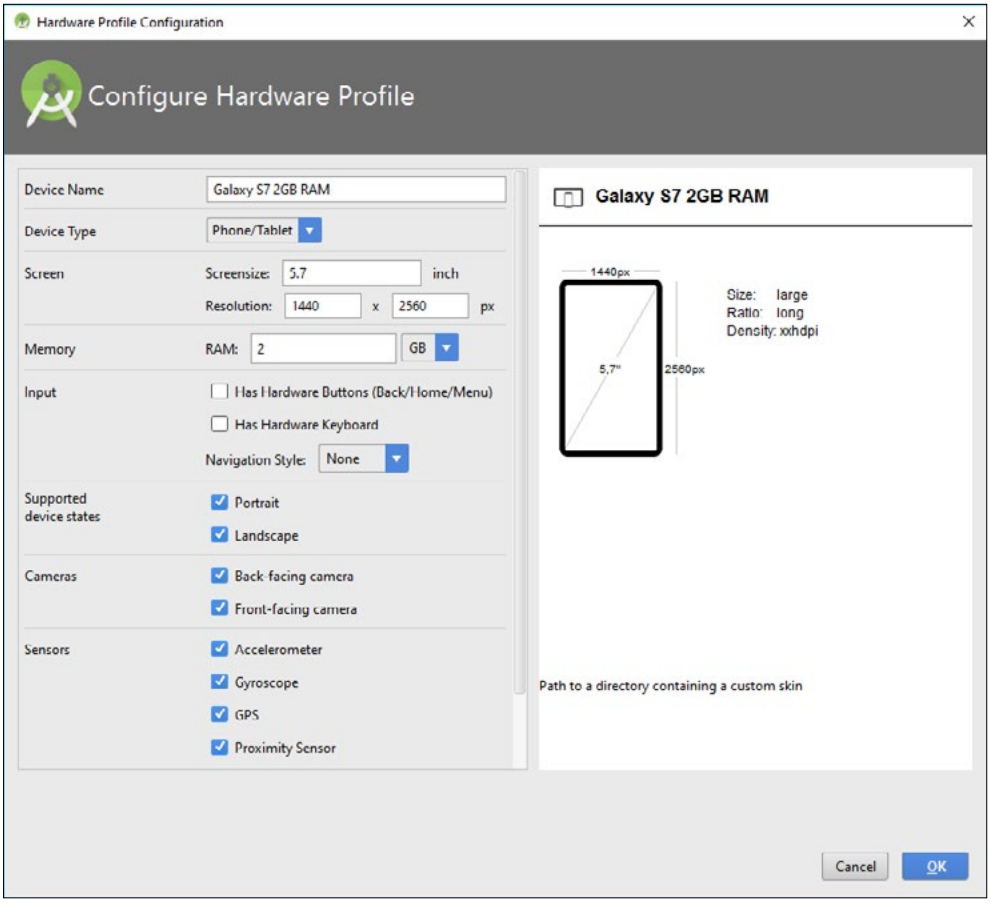
Obrázek 1.50: Ikona AVD Managera v aplikaci Android Studio

Tlačítkem **Create Virtual Device** vytvořte nový emulátor požadovaného zařízení. V dialogovém okně na vytvoření emulátoru jsou předdefinovaná zařízení rozdělena do kategorií. K dispozici jsou kategorie **Phone**, **Tablet**, **Wear** a **TV**. Buď si vyberete některé z předdefinovaných zařízení a tehdy doporučujeme vhodný Nexus s čistým Androidem, nebo pokud máte speciální požadavky, například potřebujete více či méně paměti RAM, vytvořte emulátor požadovaného zařízení klonováním.



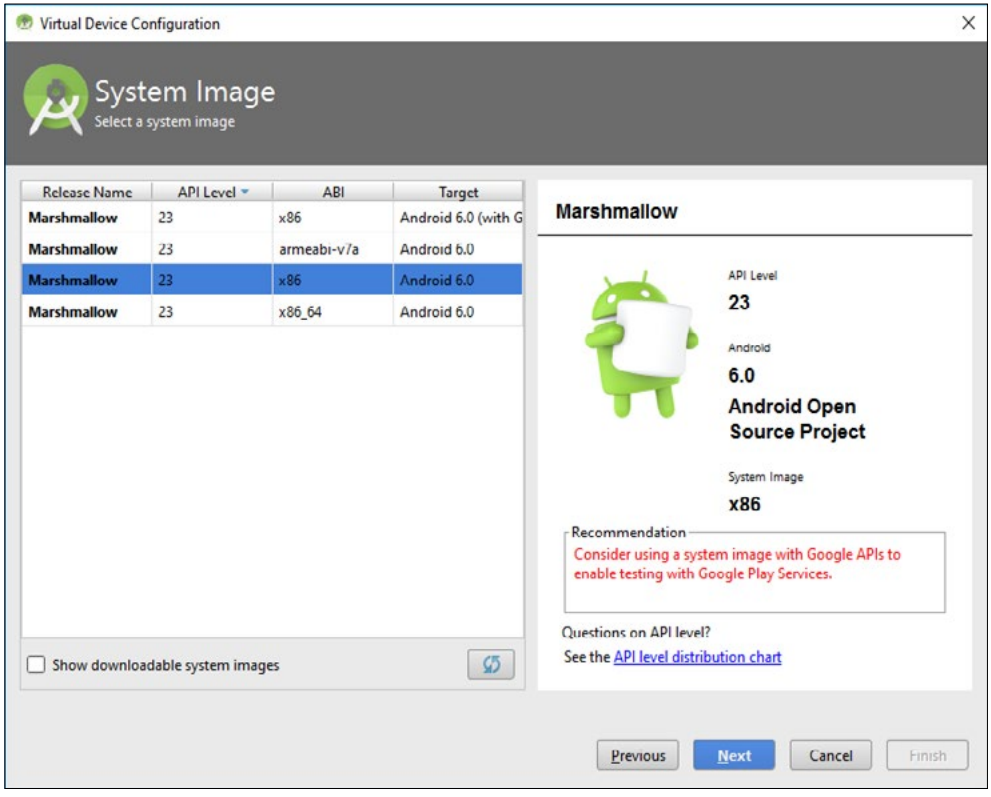
Obrázek 1.51: Vytvoření emulátoru

Například potřebujete emulátor Samgungu Galaxy S7 Edge se 4 GB RAM a displejem QHD s tím, že vám budou stačit 2 GB, abyste příliš neukrojili z paměti vývojářského počítače. Vyberte nejvhodnější podobné zařízení, například Nexus 6P, a klepněte na tlačítko **Clone Device**. Upravte velikost paměti.



Obrázek 1.52: Vytvoření emulátoru klonováním

Následuje výběr verze operačního systému. AVD Manager vám nabídne verzi, která nejrychleji poběží na vašem vývojářském počítači. V našem případě jsme ponechali implicitně vybranou volbu x86, přestože originální zařízení má procesor Exynos nebo Qualcom Snapdragon s architekturou ARM.

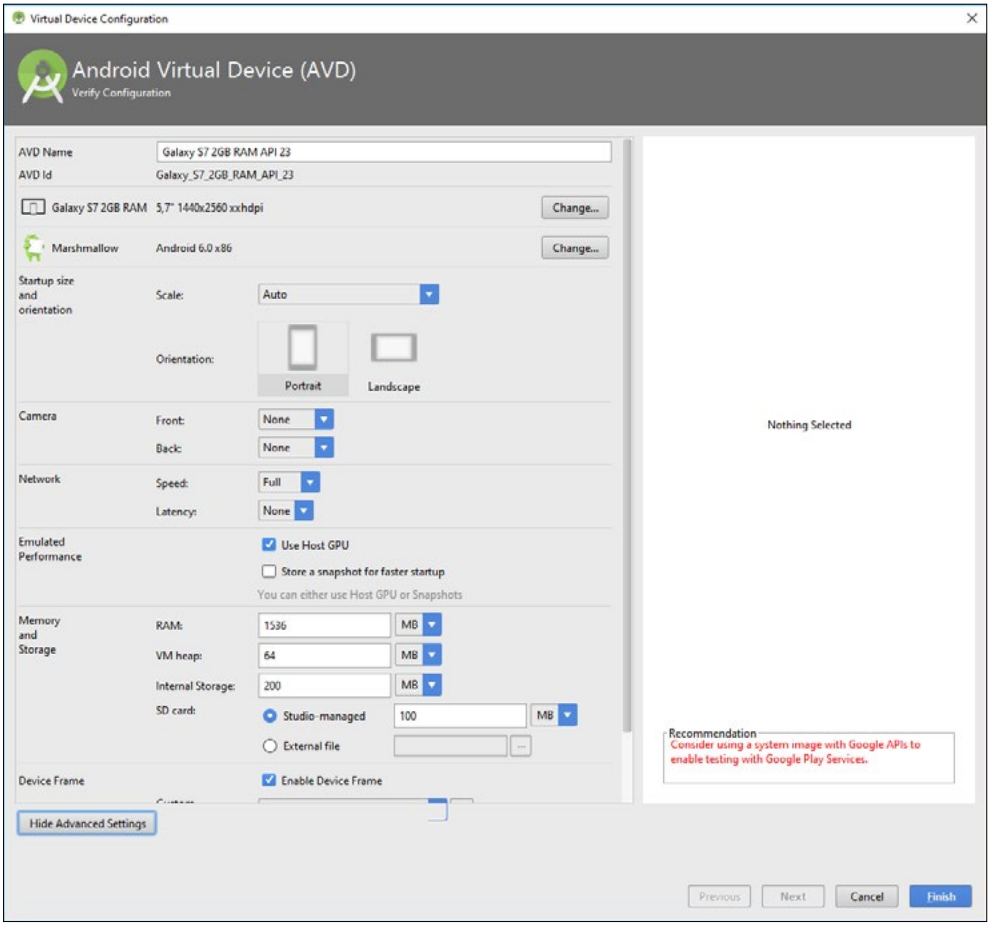


Obrázek 1.53: Vytvoření emulátoru – výběr verze operačního systému

Po potvrzení zadaných parametrů se emulátor přidá do seznamu. Pro účely této publikace doporučujeme vytvořit dva emulátory, oba pro verzi Android 6.0 Marshmallow (API Level 23) nebo 5.0 Lollipop: jeden emulátor telefonu s úhlopříčkou 5 palců a rozlišením 768 × 1280, druhý emulátor tabletu s úhlopříčkou 7 až 10 palců a rozlišením full HD 1 920 × 1 080 pixelů nebo vyšším.

Pokud označíte možnost **Store a snapshot for faster startup**, druhé a další spuštění emulátoru proběhne velmi rychle, protože AVD po zavření uloží svůj aktuální stav. První spuštění emulátoru trvá trochu déle, u dalších spuštění je už doba náběhu přiměřená. Při vytváření emulátoru nezapomeňte nakonfigurovat dostatečnou kapacitu paměti RAM, úložného prostoru a případně i SD karty, pokud ji bude aplikace využívat.

Z důvodu kompatibility vyberte nejnižší předpokládanou verzi systému. Takovéto aplikace budou na vyšších verzích fungovat bez problémů, opačně to ale neplatí.



Obrázek 1.54: Vytvoření emulátoru – nastavení parametrů

Při používání emulátoru může nastat problém s jeho zobrazením. Základní mód většiny telefonů a tabletů s Androidem je totiž „na výšku“, naproti tomu monitory vývojářských počítačů jsou orientované „na šířku“. Jak zobrazit tablet s rozlišením 1 920 × 1 080, případně vyšším, na monitoru s vertikálním rozlišením 768 pixelů? Nemusíte se bát, s implicitním nastavením parametru **Startup size and orientation** se emulátor zobrazí v takovém měřítku, aby maximálně využil plochu vaší obrazovky.

GitHub

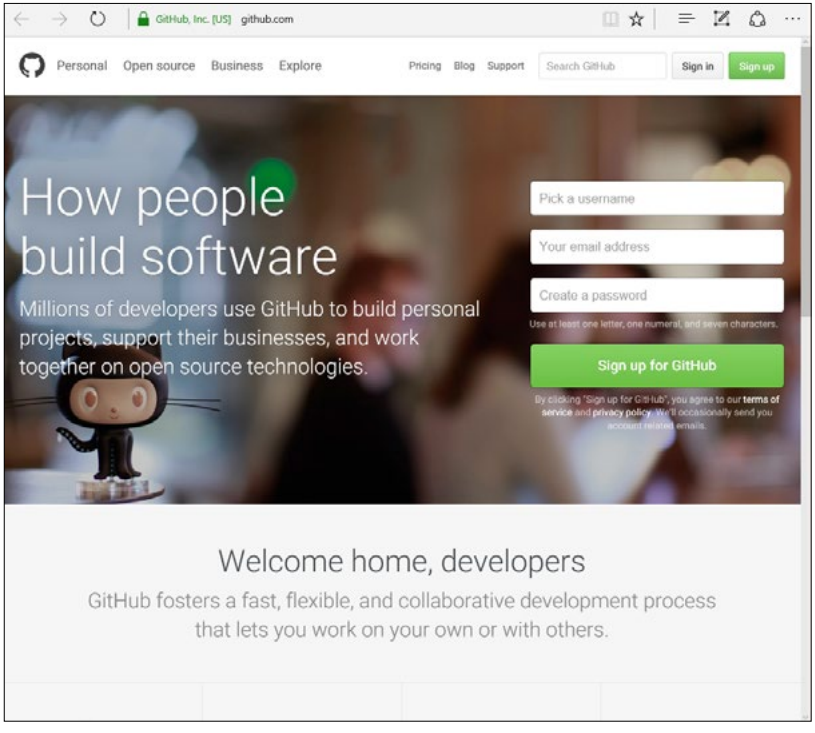
Správa verzí zdrojového kódu aplikací je důležitá především u týmového vývoje, ale díky řadě výhod ji využívají i samostatní vývojáři. Předchází tak mnohým problémům souvisejícím se ztrátou zdrojových a jiných souborů, případně jejich přepsáním. I samostatní vývojáři často pracují na projektu z více míst a počítačů, například v práci, doma, případně u zákazníka. Potřebují pracovat kontinuálně a úpravy rychle realizované u zákazníka se do produkční větve aplikace zapracují až po jejich důkladném vyzkoušení na testovací verzi.

Dobrý systém na správu verzí umožňuje přehledně oddělit aktuálně distribuovanou funkční verzi aplikace od upravovaných verzí a navíc vytvářet pokusné verze, které v konečném důsledku mohou, ale nemusí znamenat na jedné straně slepou větev a na druhé straně revoluční upgrade.

Git je systém na správu verzí, který byl původně navržen pro správu verzí open-source operačního systému Linux. V současnosti je nejrozšířenějším systémem na správu verzí zdrojového kódu aplikací a je populární nejen u hobby vývojářů. GitHub je online služba, která vám umožní vaše data v Git ukládat na online server, abyste je měli vždy přístupná. Navíc umožňuje kompletní správu vašich repozitářů přes webové rozhraní. Služba je k dispozici v základní verzi zdarma, to v případě, že vám nevadí, že vaše kódy budou veřejně přístupné a kdokoliv si je může vyhledat, prohlédnout a případně stáhnout jako celek nebo zkopírovat bloky kódu. Pokud chcete mít repozitář, který není veřejně přístupný, musíte za tuto možnost zaplatit.

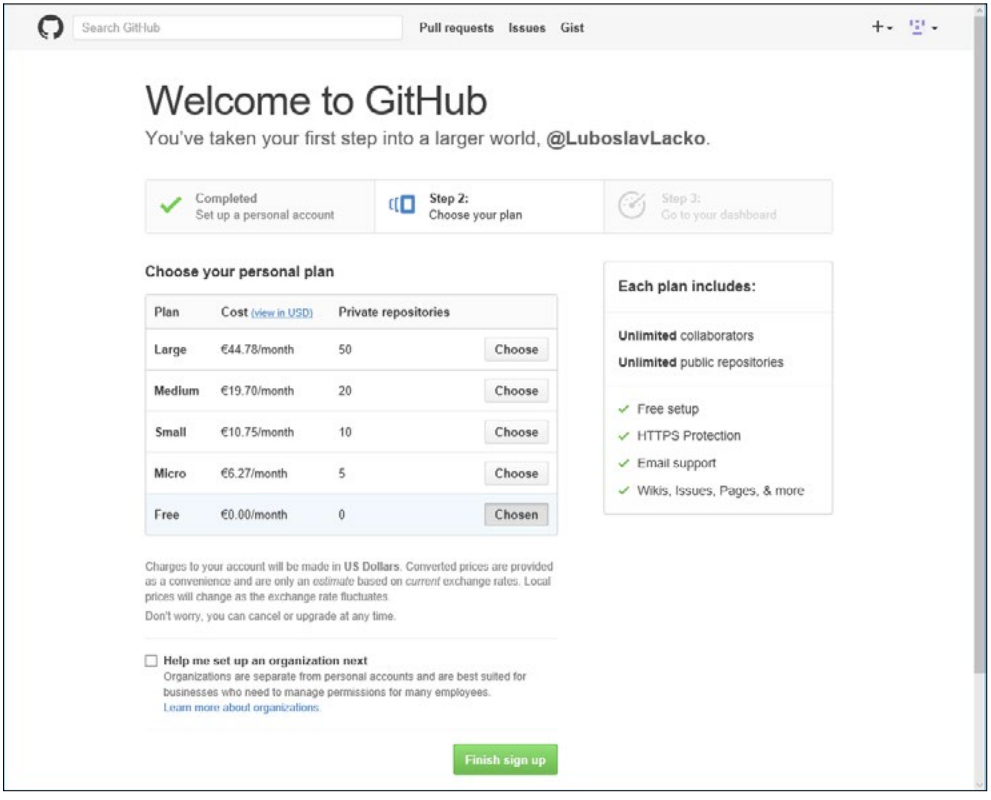
Vytvoření účtu

Vstupním bodem služby je stránka *www.github.com*. Na vytvoření účtu stačí zadat přihlašovací jméno, e-mail a heslo.



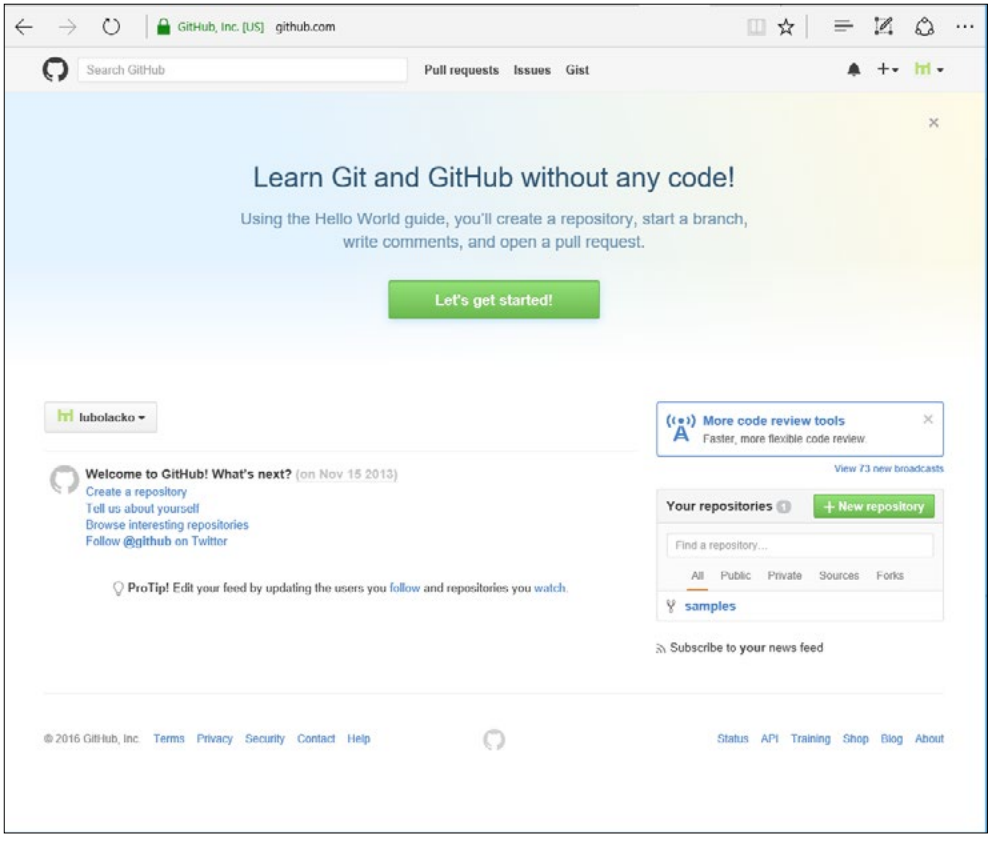
Obrázek 1.55: Vytvoření účtu ve službě github.com

Jak už bylo zmíněno, základní varianta s veřejnými repozitáři je zdarma. Cena placených variant se odvíjí od počtu privátních repozitářů. Všimněte si, že pro každý plán poskytování služby, včetně bezplatného, můžete vytvářet neomezený počet repozitářů a není omezen ani počet participujících na jednotlivých projektech.



Obrázek 1.56: Cenové varianty služby

Plné vytvoření účtu vyžaduje potvrzení odkazu v ověřovacím e-mailu. V opačném případě nebude váš účet plně funkční a webový portál služby vás bude vyzývat k ověření.



Obrázek 1.57: Úvodní stránka zobrazená po vytvoření služby

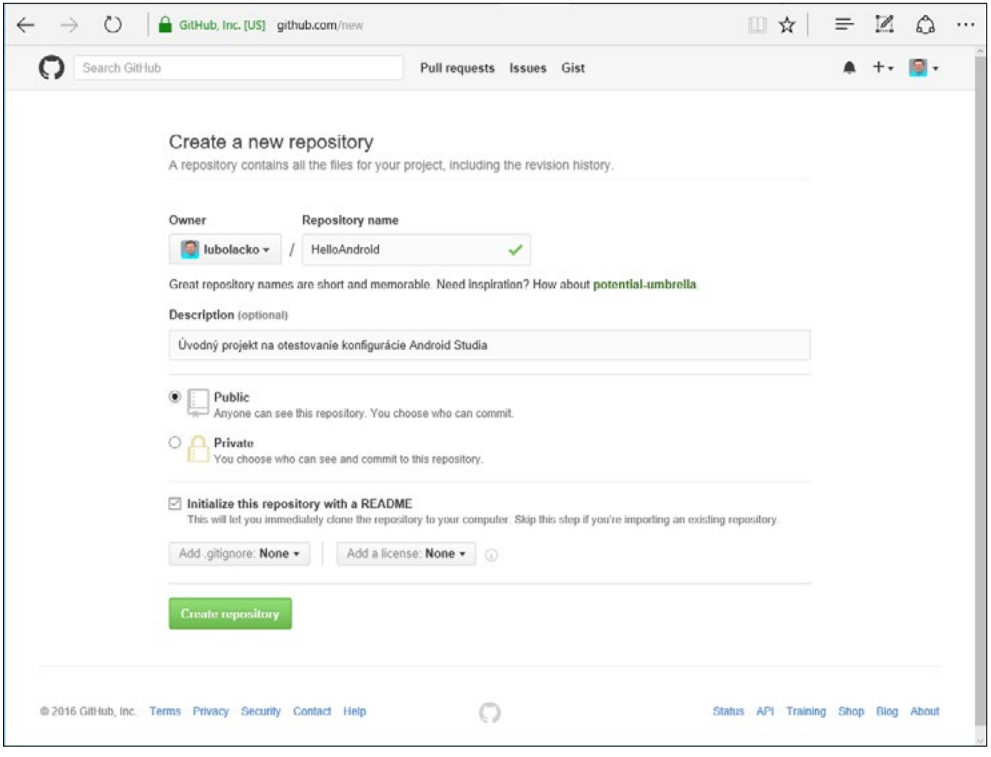
Doporučujeme v profilu aktualizovat fotku vašeho avatara, případně zveřejnit jméno či e-mailovou adresu. Pro komunitu vývojářů je přece důležitá především vzájemná spolupráce a sdílení poznatků.

Vytvoření repozitáře

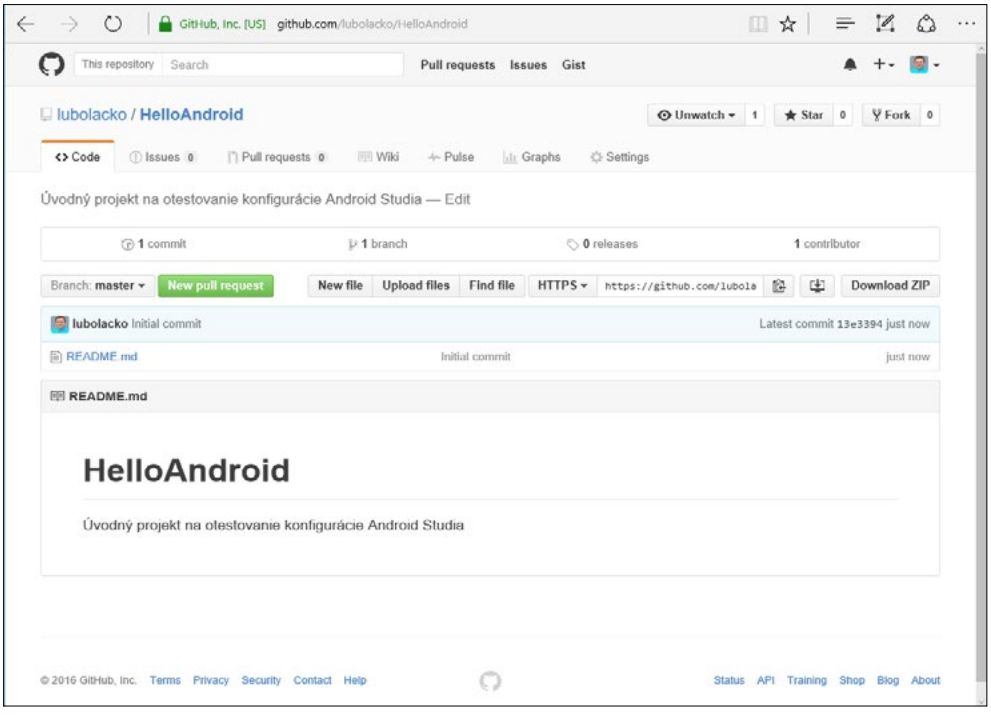
Pokud jste server GitHub dosud nevyužívali, doporučujeme absolvovat krátký, přibližně 10minutový tutoriál, který nabízí hlavní stránka projektu po prvním přihlášení. Nejprve je potřeba vytvořit nový repozitář. Repozitář si nejjednodušeji představíte jako složku, ve které máte umístěný jeden projekt, to znamená všechny soubory, které ho tvoří, včetně různých verzí.

Repozitář vytvoříte klepnutím na ikonu se symbolem„+“ v pravé horní části obrazovky vedle svého avatara. Potvrďte volbu **New repository** a do pole **Repository name** zadejte název vašeho repozitáře. Doporučujeme dodržet radu pod zadávacím polem, abyste vybírali krátké a výstižné názvy. Také doporučujeme do pole **Description** napsat krátký a výstižný jednořádkový popis repozitáře. Pokud používáte free verzi služby, může být váš repozitář jen typu **Public**, tj. veřejně dostupný. Volba **Private** se zpřístupní jen pro placené účty.

Všimněte si volby **Initialize this repository with a README**. Pokud ji zaškrtnete, vytvoří se v novém repozitáři soubor *README.md*. Při seznamování se s GitHubem aktivací této volby de facto zařídíte, že repozitář bude obsahovat aspoň jeden soubor a bude s ním možné pracovat. Zároveň se pro takovýto nyní už plnohodnotný repozitář, i když jen s jedním souborem, vytvoří klon na vašem počítači. Pokud volbu neoznačíte, musíte si první soubory nahrát do repozitáře sami. Tlačítkem **Create repository** vytvoříte nový repozitář.



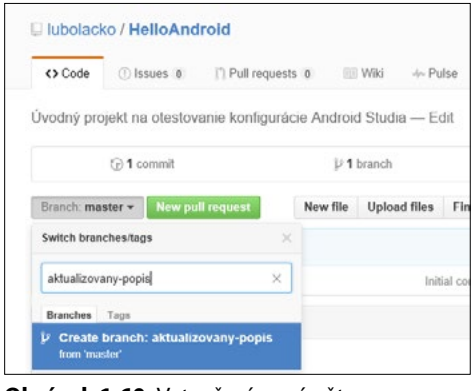
Obrázek 1.58: Vytvoření repozitáře



Obrázek 1.59: Nově vytvořený repozitář. Zatím obsahuje jen soubor README.md

Vytvoření větve (branch)

Každá linie vašeho projektu bude tvořit jednu paralelní větev. Například jedna větev bude aktuální verze aplikace pro Android, která je už publikovaná v aplikačním obchodě, a další větev bude verze, na které právě pracujete, vylepšujete ji, a když ji důkladně otestujete, může se stát finální verzí. Tehdy ji zpravidla sloučíte se svou hlavní (master) větví. Větev Master branch obsahuje produkční verzi projektu, která je v aplikačním obchodě nebo nasazená v ostrém provozu. Tato větev se ve většině případů stává základem na vytváření nových, paralelních větví.



Obrázek 1.60: Vytvoření nové větve

Postup vytvoření nové větve v repozitáři je jednoduchý. Klepněte na prvek s označením **Branch: master**. Zobrazí se okno k zadání nové větve. Větev vhodně a výstižně pojmenujte.

Po tomto úkonu máte v repozitáři dvě větve, v našem případě „master" a „aktualizovany-popis". Zatím jsou stejné. V reálném projektu můžete provádět změny v nové větvi, aniž by se tyto změny promítly do větve master.

Úprava souborů

Git spravovaným souborům přiděluje tři základní stavy:

- **Zapsané (committed)** – data bezpečně uložená ve vaší lokální databázi
- **Změněné (modified)** – v souboru byly provedeny změny, avšak soubor ještě nebyl zapsán do databáze
- **Připravené k zapsání (staged)** – změněný soubor jste v jeho aktuální verzi určili k tomu, aby byl zapsán v další revizi (tzv. Commit)

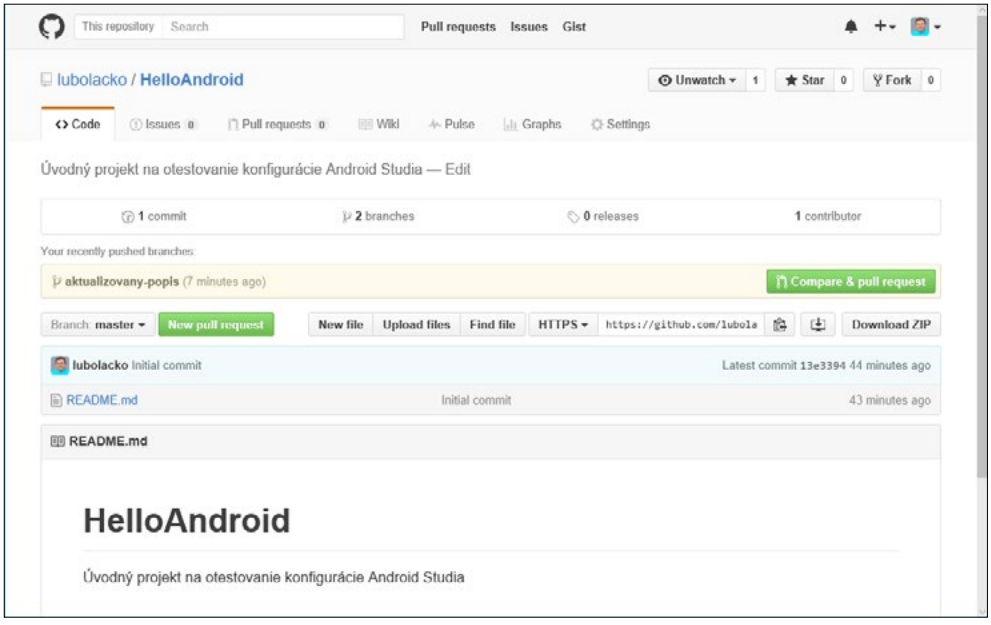
Změny si vyzkoušejte na editování textového souboru *README.md*. Klepněte na název souboru a následně na ikonu tužky v pravém horním rohu. Otevře se okno, ve kterém můžete editovat obsah souboru. Změny potvrdíte tlačítkem **Commit changes**. Operace **Commit** umožňuje uložení provedených úprav, během kterého se vytvoří skupiny změn v příslušných souborech. Během operace **Commit** je potřeba napsat komentář s popisem provedených změn. Později vám tento komentář může značně pomoct, napište proto do něj všechno potřebné o provedených úpravách.

Na synchronizaci zdrojových kódů jsou určeny operace:

- **Push** – odeslání všech naposled provedených operací commit na server.
- **Pull** – aktualizace vašich lokálních souborů ze serveru GitHub. Aktualizuje se jen větev, pro kterou tuto operaci požadujete.

Klepněte na tlačítko **New pull request**. Vyberte větev, v našem případě „aktualizovany-popis", a porovnejte ji s hlavní větví. Pokud souhlasíte se změnami, klepněte na tlačítko **Create pull request**.

Velmi zajímavou funkcí je **Fork** (rozvětvení). Použijete ji tehdy, pokud do svého účtu potřebujete přidat repozitář jiného vývojáře, nejčastěji proto, abyste jeho kód modifikovali a vytvořili na jeho bázi vlastní projekt. Funguje to jednoduše: Otevřete cizí repozitář a klepněte na tlačítko **Fork**. Na funkci Fork se můžete podívat ze dvou úhlů pohledu. Buď vy přeberete repozitář jiného vývojáře, nebo jiný vývojář převzal váš repozitář. Zda a kolikrát se tak stalo, zjistíte podle číselného údaje vedle tlačítka **Fork**. Klepnutím na tlačítko **Network** zjistíte, kdo vytvořil původní repozitář a jaké úpravy se v něm následně prováděly.



Obrázek 1.61: Aktualizovaný popis

Pokud chcete uploadovat soubory z vývojářského počítače na server GitHub, musíte si nainstalovat aplikaci, kterou stáhnete z *https://desktop.github.com*. Nainstalují se dvě aplikace: *GitHub* a konzolová aplikace *GitShell*.

GitHub je de facto sociální síť, která úplně změnila způsob, jakým pracují komunity vývojářů. V současnosti je to největší online úložný prostor ke spolupráci na tvorbě softwarových a jiných projektů, kde se pracuje s textem.

KAPITOLA

2

Anatomie Androidu

Témata kapitoly:

- Multiplatformní operační systém
- Historie verzí
- Starší verze
- Android 5.0 Lollipop
- Android 6.0 Marshmallow
- Android 7.0 Nougat
- Stručně o architektuře Androidu

Android je platforma open-source na bázi Linuxu určená hlavně pro mobilní zařízení, tedy chytré telefony, tablety a stále více se prosazuje i v chytrých hodinkách, televizorech, autech a podobně.

Multiplatformní operační systém

Systém Android vyvíjí organizace Open Handset Alliance, jejíž součástí jsou desítky firem včetně těch nejznámějších v mobilní branži – Google, HTC, Intel, NVIDIA, Qualcomm, Samsung atd. Jde o jeden z mála operačních systémů podporujících více platforem, můžete ho vidět v zařízeních nejrůznějších značek. To však přináší jednu značnou nevýhodu – chybí optimalizace systému na konkrétní platformu, což je silná zbraň Apple iOS. Android je však multiplatformní s možností přizpůsobení a vytvoření nadstaveb (Samsung TouchWiz, HTC Sense a mnohé další). Na druhé straně když Google vydá aktualizaci systému, ta není hned dostupná pro všechna zařízení a uživatel si musí počkat na konkrétní aktualizaci od svého výrobce.

Největší výhodou a zároveň nevýhodou platformy je její otevřenost a možnost úprav, ať už ze strany výrobců, nebo uživatelů. Úpravy se netýkají jen konfigurace či widgetů, ale i firmwaru. Pro Android je k dispozici nejvíce aplikací, mnohé jsou však pochybné kvality, jelikož proces jejich schvalování není tak přísný jako u iOS či Windows. Tablety a telefony s Androidem dodává hodně firem. Je to záruka dynamičtějšího vývoje nových zařízení, například v porovnání s Apple, kde je celý vývoj hardwaru v režii jedné firmy, a zároveň představuje problém, protože aplikace běží na na přístrojích s různým rozlišením displeje a různým výkonem procesoru a grafiky. V praxi to znamená různý komfort uživatelského ovládání. Na rozdíl od ostatních platforem nevydává aktualizace operačního systému centrálně Google, ale výrobci zařízení, takže se u různých zařízení můžete setkat s různými verzemi.