Podstawowe pojęcia i struktura systemu Android. ConstraintLayout

Aplikacje mobilne

1. Przegląd systemów dla urządzeń mobilnych

Aplikacje mobilne

NIEROZWIJANE

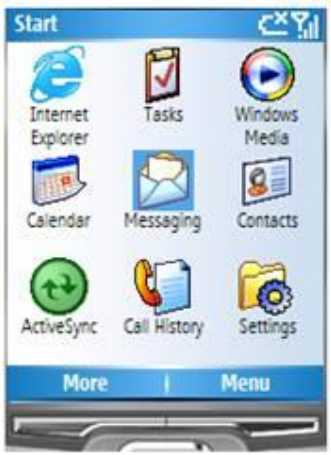
symbian OS

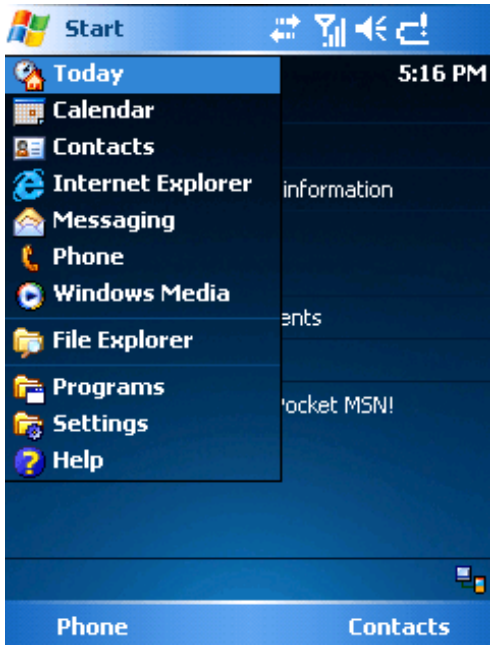
- ✓ Wywodzi się z systemu EPOC dla palmtopów firmy PSION
- ✓ Psion, Nokia, Ericssoni Motorola założyli w 1999 r. firmę Symbian
- ✓ Pierwszy Symbian 6.0 w 2000 roku (numerek odziedziczył po EPOCu)
- ✓ W 2008 Nokia przejęła większość udziałów i przekształciła firmę w fundacje Symbian
- ✓ Podstawowe języki programowania:
  - do 2010 r: Symbian C++
  - od 2010: standardowy C++ wraz z Qt
  - można używać też innych języków (Python, Java ME, .NET…)

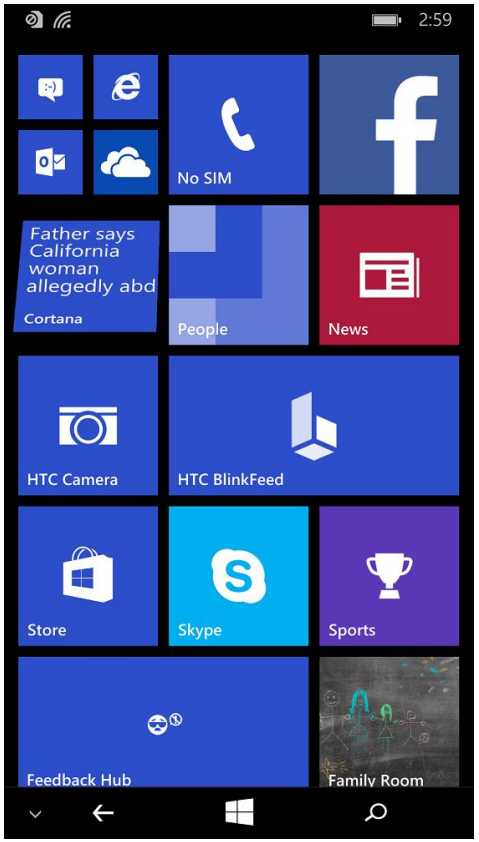
Aplikacje mobilne

NIEROZWIJANE

Windows CE Windows Mobile Windows Phone







- ✓ Protoplasci: Windows CE (1996),
- ✓ PocketPC 2000-2002
- ✓ Windows Mobile 2003,
- ✓ Windows Mobile 5 (2005), 6 (2007), 6.5 (2010)
- ✓ Wersja dla smartfonów: Windows Phone7 (2010) –niekompatybilny z wcześniejszymi używa interfejsu METRO
- ✓ Języki programowania: C#, VB Sliverlight XNA Wgrany pakiet Offic

Aplikacje mobilne

iOS

- ✓ Wywodzi się z systemu Mac Os X, do 2010 r. iPhone OS Oryginalnie powstał dla iPhone'a, później rozszerzony do obsługi iPoda, iPadai AppleTV
- ✓ Zamknięty system, brak możliwości instalacji aplikacji spoza AppStore
- ✓ Dość kosztowna możliwość dystrybucji aplikacji (99 USD rocznie)
- ✓ Język programowania: Objective-C Wymagany komputer Mac z procesorem Intela
- ✓ Prosty, czytelny, bardzo sprawnie działający interfejs użytkownika

Aplikacje mobilne

Aplikacje mobilne

Android

- ✓ W lipcu 2005 roku Google kupuje firmę Android Inc., produkującą oprogramowanie dla urządzeń mobilnych
- ✓ Premiera systemu i pierwsze smartfony: połowa 2008
- ✓ Oparty na Linuksie, z własną maszyną wirtualną Dalvik / ART
- ✓ Język programowania: Java, z możliwością wstawek w C++
- ✓ System otwarty, umożliwia instalowanie aplikacji spoza Marketu
- ✓ Dostępne źródła systemu

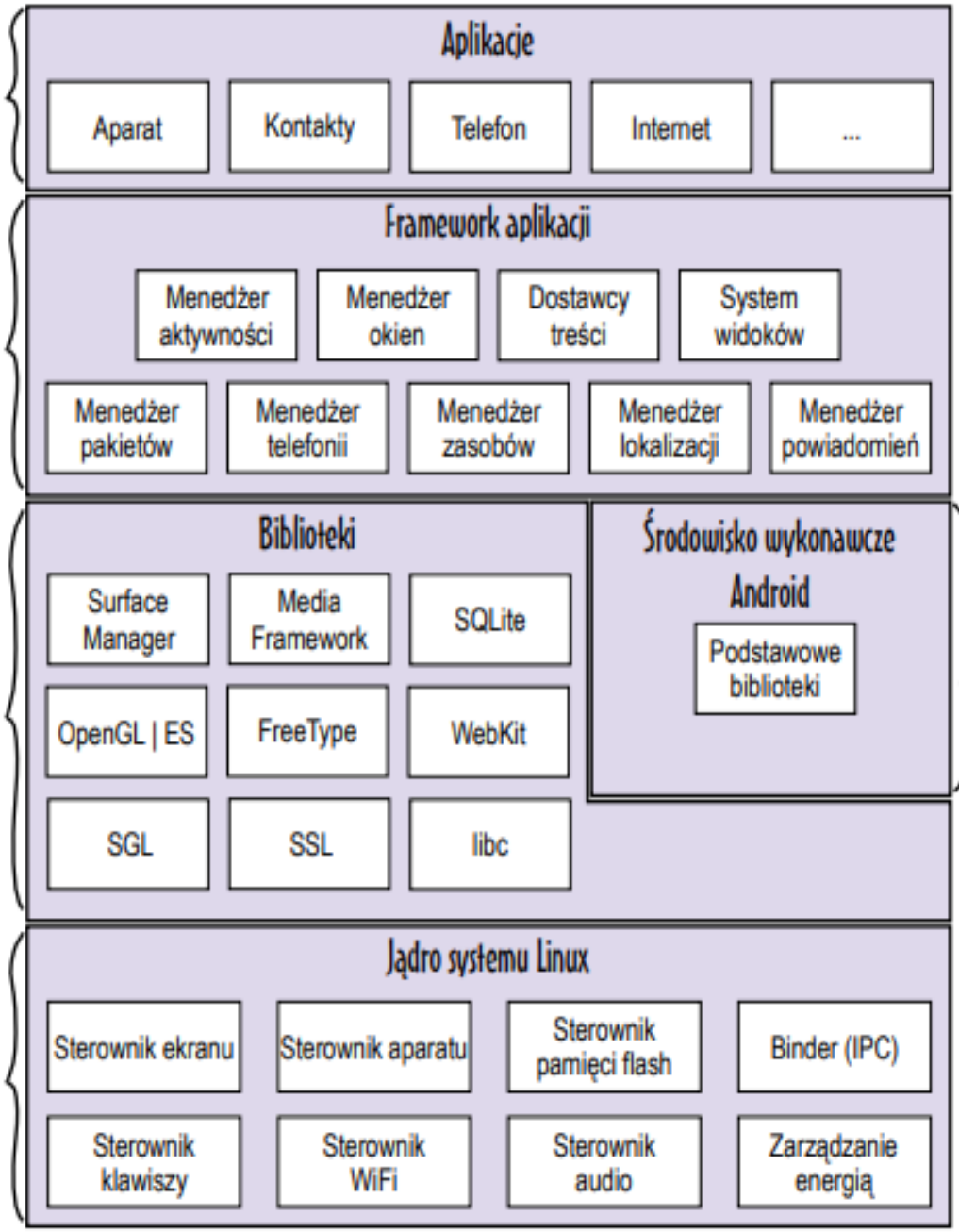
2. Budowa systemu Android

Android jest wyposażony w zestaw podstawowych aplikacji, takich jak Kontakty, Kalendarz i Mapy, oraz w przeglądarkę WWW.

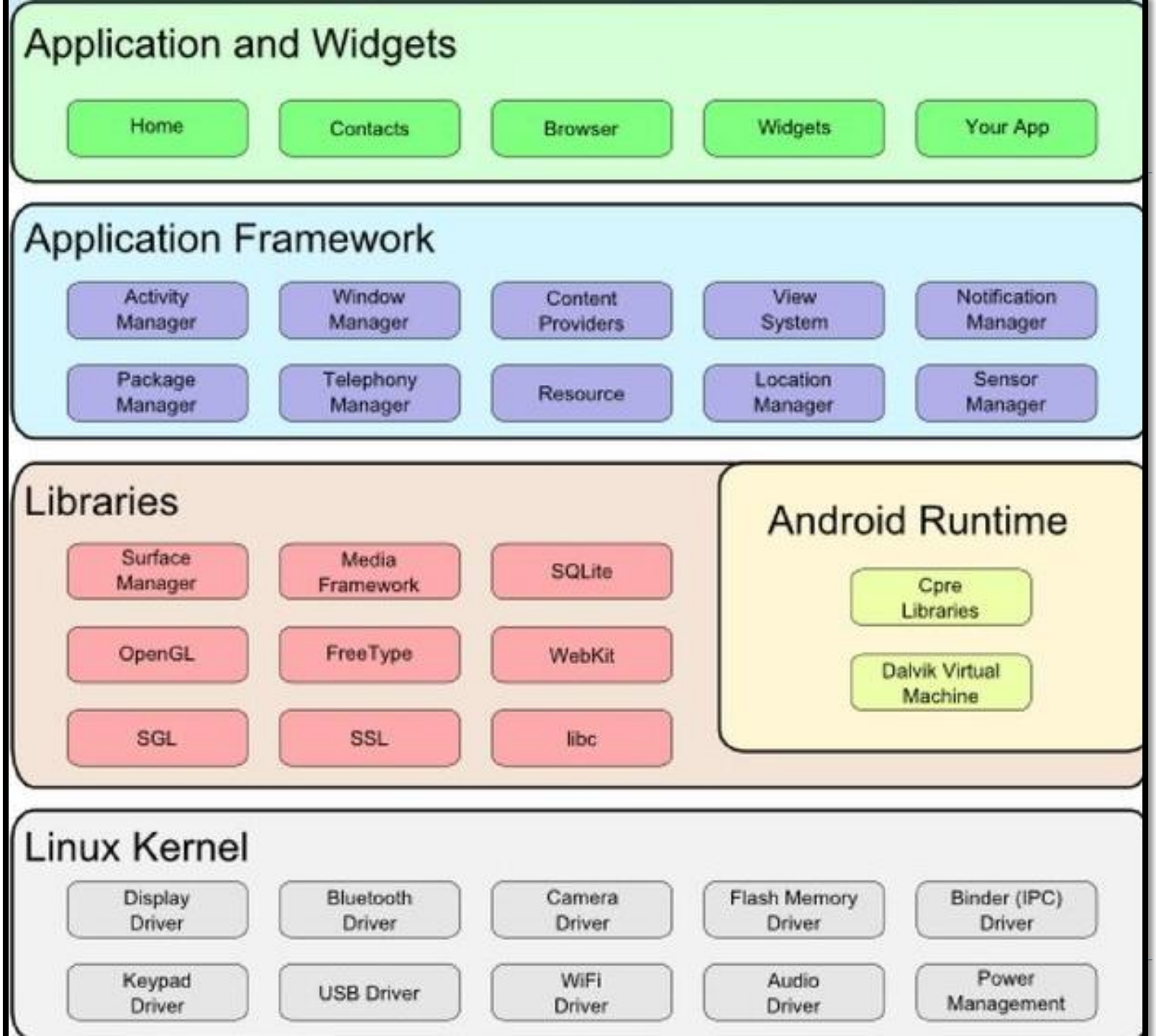
Pisząc własne aplikacje, mamy dostęp do tych samych podstawowych API, które są używane przez podstawowe aplikacje. Używając tych API, kontrolujemy wygląd aplikacji i jej działanie.

Poniżej frameworku aplikacji ukryty jest zestaw bibliotek C i C++. Dostęp do nich zapewniają API należące do frameworku aplikacji.

Poniżej wszystkich pozostałych warstw platformy Android umieszczone jest jądro systemu Linux. To ono odpowiada za obsługę sterowników oraz podstawowych usług, takich jak zabezpieczenia i zarządzanie pamięcią.



Środowisko uruchomieniowe Androida jest wyposażone w zestaw podstawowych bibliotek, implementujących przeważającą część języka programowania Java. Każda aplikacja na Androida jest wykonywana we własnym procesie.


Application and Widgets
Home
Contacts
Browser
Widgets
Your App
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Notification Manager
Package Manager
Telephony Manager
Resource
Location Manager
Sensor Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Cpre Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Bluetooth Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
USB Driver
WiFi Driver
Audio Driver
Power Management

Aplikacje mobilne

Linux Kernel - (jądro linuksowe) - Android opiera się na wersji jądra 2.6 dla podstawowych usług systemowych, takich jak bezpieczeństwo, zarządzanie pamięcią, zarządzanie procesami, stos sieciowy i model sterownika. Jądro działa również jako warstwa abstrakcji pomiędzy sprzętem i resztą stosu oprogramowania.

Android Runtime - (środowisko uruchomieniowe) - Android zawiera zbiór bibliotek, które dostarczają większość funkcji dostępnych w bibliotekach podstawowych języka Java. Każda aplikacja działa we własnym procesie, z własnej instancji maszyny wirtualnej Dalvik. Dalvik została napisana tak, że na urządzeniu można uruchomić wiele maszyn wirtualnych. Dalvik VM wykonuje pliki wykonywalne (.dex) skonstruowane tak, aby zużywały minimalną ilość pamięci

Aplikacje mobilne

Libraries - (biblioteki) - Android zawiera zestaw bibliotek C / C++ używanych przez różne elementy systemu. Możliwości te są udostępnione programistom poprzez Application Framework.

Application Framework - (framework aplikacji) - programiści mają pełny dostęp do tego samego API, którego używają aplikacje podstawowe systemu.

Aplikacje mobilne

Framework oferuje:

- widoki (Views), które można wykorzystać do budowania aplikacji,
- dostawców treści (ContentProviders), które pozwalają aplikacjom dostęp do danych z innych aplikacji, (np. takich jak Kontakty), lub dzielenie się swoimi danymi.
- menedżer zasobów (Resource Manager), umożliwiający dostęp do zasobów, takich jak, grafika i pliki.
- Notification Manager, który umożliwia wszystkim aplikacjom wyświetlanie powiadomień w pasku stanu
- ActivityManager, który zarządza cyklem życia aplikacji

Struktura aplikacji na Androida

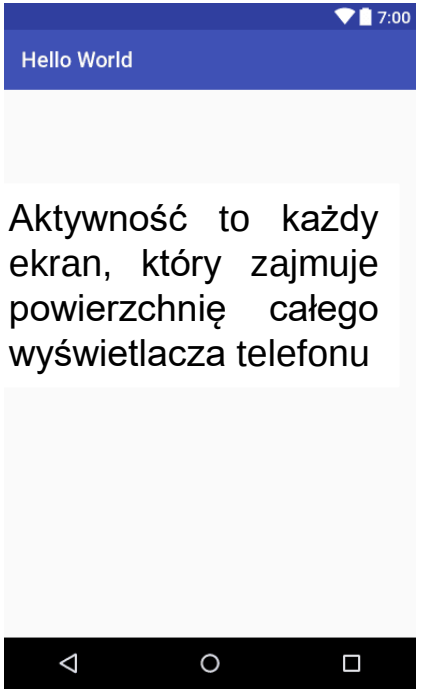
Aplikacje na system Android składają z komponentów powiązanych ze sobą w luźny sposób, komunikujących się ze sobą za pośrednictwem Intencji (Intent).

Podstawowe komponenty to:

- Aktywności (*activities)*
- Usługi (*services),*
- Dostawcy treści (*content providers)*
- Odbiorniki komunikatów (*broadcast receivers)*

Aplikacje mobilne

- ✓ **Aktywności (*activities*)** - *oznaczają warstwę wizualną aplikacji oraz dołączone do niej funkcjonalności.*
  - *Dana aktywność reprezentuje pojedynczy ekran zawierający interfejs użytkownika, tak więc jedna aplikacja może zawierać wiele wywołujących się nawzajem aktywności.*
  - *Aplikacje mogą również uruchamiać aktywności zawarte w innych aplikacjach – pod warunkiem, że mają odpowiednie zezwolenia.*



Aplikacje mobilne

Widoki – (View) – to obiekty tworzące interfejs użytkownika. Za pomocą widoków użytkownik komunikuje się z aplikacją (z aktywnościami). Na widok składa się układ kontrolek zawartych w odpowiedniej kompozycji (layout).

- Budowa graficznego interfejsu użytkownika aplikacji Android opiera się na hierarchii obiektów **View i ViewGroup.**
- **Obiekty View** są zazwyczaj widocznymi elementami interfejsu, np. przyciskami i polami tekstowymi,
- **obiekty ViewGroup** to niewidoczne kontenery określające sposób rozmieszczenia elementów widoków potomnych, np. w siatce albo pionowej liście.

Aplikacje mobilne

- ✓ Usługi *(services),* *w przeciwieństwie do aktywności, nie posiadają reprezentacji graficznej.*
  - *Działają one w tle i wykonują czasochłonne zadania lub operacje na rzecz zdalnych procesów, np. pobierają pliki z Sieci i aktualizują źródła danych, wysyłają powiadomienia, itd.*

Aplikacje mobilne

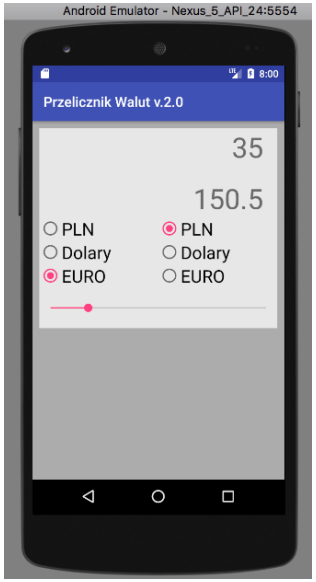
- ✓ Intencje (*intent*) - *to rodzaj komunikatów wysyłanych do komponentów aplikacji.*
  - *Służą do wywoływania aktywności (zarówno własnego programu jak i zewnętrznych zarejestrowanych w systemie).*
  - *Aplikacja posiada w pliku manifest.xml filtr intencji – precyzujący na jakie intencje jest ona w stanie zareagować.*
  - *Intencja może użyć aplikacji która nie jest aktualnie uruchomiona – wystarczy, by była zarejestrowana w systemie.*

.

Aplikacje mobilne

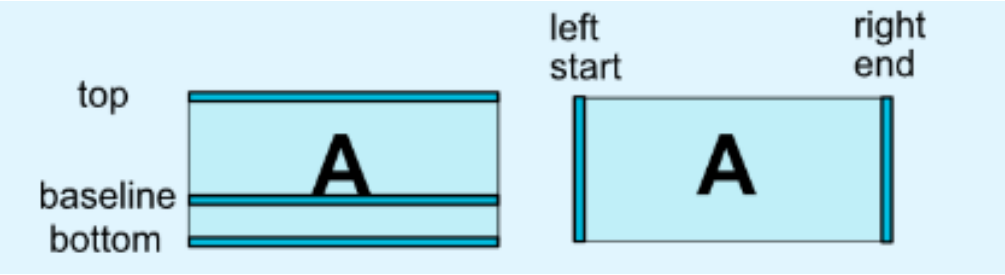
- ✓ **Dostawcy treści (*content providers*)** *to komponenty służące do udostępniania i współdzielenia danych pomiędzy aplikacjami.*
  - *Pracują one w podobny sposób do relacyjnych baz danych.*
  - *Informacje są udostępniane w formie struktur przypominających tabele baz danych.*
- ✓ **Odbiorniki komunikatów (*broadcast receivers*)** *umożliwiają tworzenie aplikacji sterowanych zdarzeniami (event-driven applications).*
  - *Nie wyświetlają interfejsu użytkownika, ale mogą tworzyć komunikaty na pasku powiadomień, informujące np. o otrzymaniu wiadomości lub zakończeniu pobierania pliku.*
  - *Dzieje się to w skutek aktywowania przez system lub inne aplikacje.*

LAYOUT APLIKACJI



Constraint Layoutu

Constreint Layout daje możliwość podczepienia każdej z krawędzi widoku do jednej z dwóch krawędzi dowolnego innego widoku, linii pomocniczej lub krawędzi ekranu.



Źródło: https://developer.android.com

Podstawowe polecenia:

- layout_constraintTop_toBottomOf
- layout_constraintBottom_toBottomOf
- layout_constraintBottom_toTopOf
- layout_constraintTop_toTopOf
- layout_constraintStart_toStartOf
- layout_constraintStart_toEndOf
- layout_constraintEnd_toEndOf
- layout_constraintEnd_toStartOf

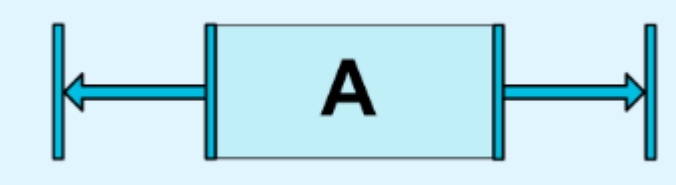
Parametrem każdego z tych poleceń jest Id innego widoku

Constraint Layoutu

Wyśrodkowanie vs. rozciągnięcie na cały ekran

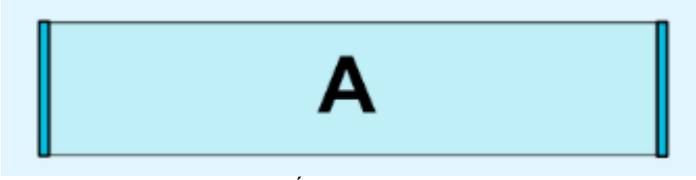
wrap_content – „dopasuj się do zawartości”

Wyśrodkowanie w poziomie



android:layout_width="**wrap_content**"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintLeft_toLeftOf="parent"

Dopasowanie do szerokości ekranu



Źródło: https://developer.android.com

android:layout_width=„**0dp**”
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintLeft_toLeftOf="parent

parent – „rodzic” kontener, w którym znajduje się widok
(najczęściej oznacza cały ekran)

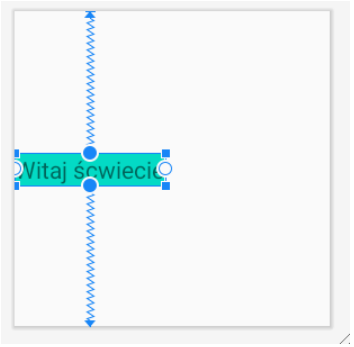
Ustawienie szerokości obiektu na 0dp oznacza, że powinna być ona wyliczona z innych ustawień

Constraint Layoutu

Wyśrodkowanie vs. rozciągnięcie na cały ekran

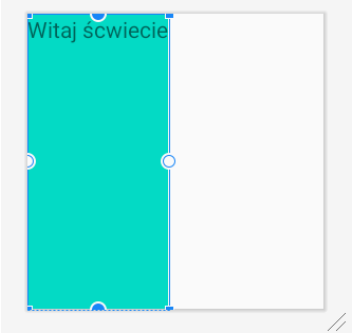
Wyśrodkowanie w pionie

```
android:layout_height="wrap_content"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintTop_toTopOf="parent"
```



Dopasowanie do wysokości ekranu

```
android:layout_height="0dp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintTop_toTopOf="parent"
```

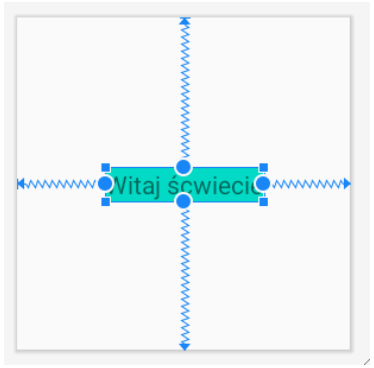


Ustawienie wysokości obiektu na 0dp oznacza, że powinna być ona wyliczona z innych ustawień

Constraint Layoutu

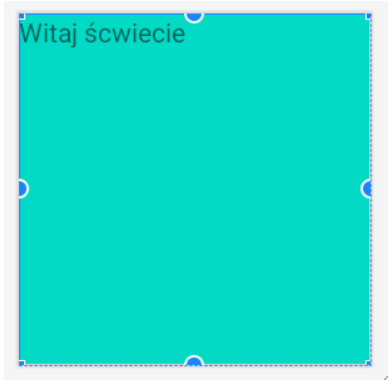
Wyśrodkowanie

```
android:layout_height="wrap_content"
android:layout_width="wrap_content"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```



Rozciągnięcie na cały ekran

```
android:layout_height="0dp"
android:layout_width="0dp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```



Ustawienie wysokości i szerokości obiektu na 0dp oznacza, że powinny być one wyliczone z innych ustawień

Constraint Layoutu

Guideline – linia pomocnicza - pozwala na ustawienie linii odniesienia do pozycjonowania pozostałych widoków. Może zostać ustawiona procentowo lub na konkretną wartość.

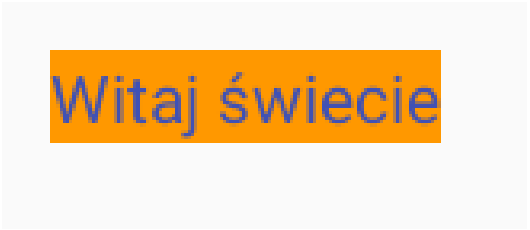
```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_begin="20dp" />
```

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.5" />
```

Definiowanie wyglądu elementów

Kolory

```
android:background="#FF9800"
android:textColor="#3F51B5"
```



Jednostki

px – piksele: odradzane ze względu na różną gęstość pikseli na wyświetlaczach.

mm – milimetry: jednostka bazująca na wielkości wyświetlacza.

in - cale: jednostka bazująca na wielkości wyświetlacza.

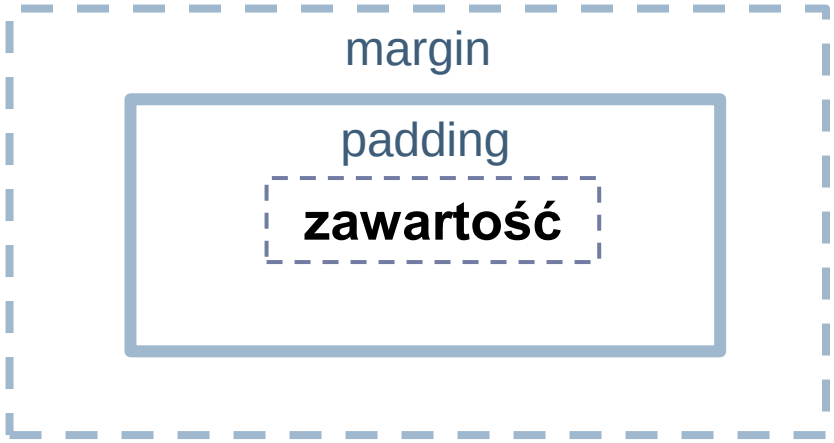
pt – punkty: 1pt = 1/72 cala

dp – Density-independent Pixels: polecana - jednostka niezależna od gęstości pikseli. Uniezależnia element layoutu od rozdzielczości oraz wielkości wyświetlacza. Jednostka ta bazuje na wyświetlaczu 160dpi (1dp jest odpowiednikiem 1 piksela przy gęstości 160dpi) Stosunek piksel-dp jest za każdym razem dobierany do gęstości wyświetlacza.

sp - Scale-independent Pixels: Jednostka podobna do dp. Podczas skalowania pod uwagę brane są również ustawienia użytkownika (Settings.System.FONT_SCALE). Należy stosować do definiowania wielkości czcionek oraz wszystkich wymiarów z nimi związanych.

Definiowanie wyglądu elementów

Model pudełkowy widoku (elementu typu view)



Od modelu pudełkowego znanego z HTML i CSS różni się on brakiem elementu „border” czyli możliwości definiowania obramowania. Wynika to stąd, że mamy do czynienia z predefiniowanymi kontrolkami.

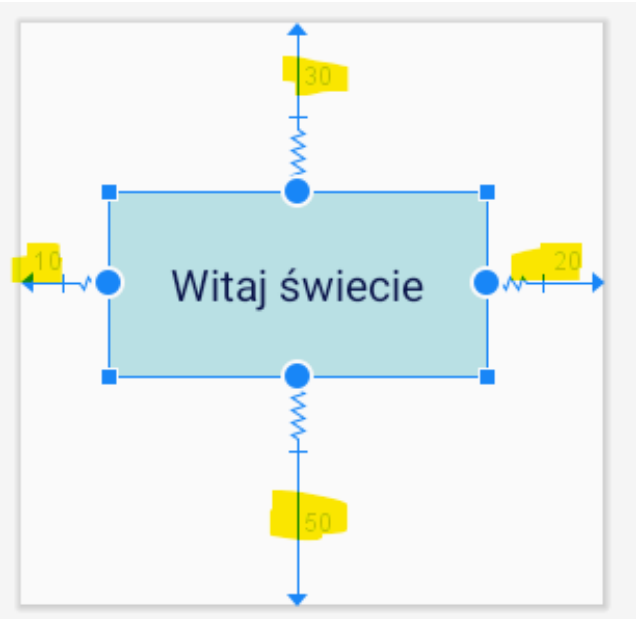
Definiowanie własnych obramowań jest oczywiście możliwe, lecz nieco bardziej skomplikowane (wymaga przedefiniowania wyglądu kontrolki)

Definiowanie wyglądu elementów

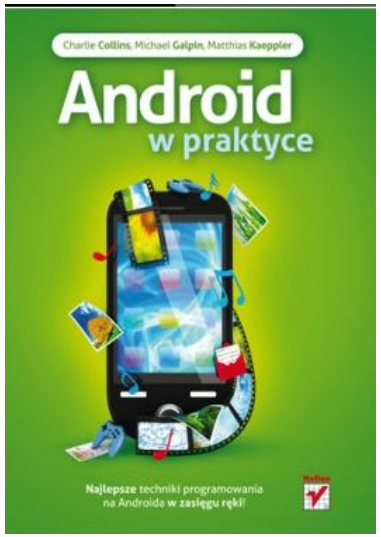
Model pudełkowy widoku (elementu typu view)

```
android:padding="20dp"
android:layout_margin="10dp"
```

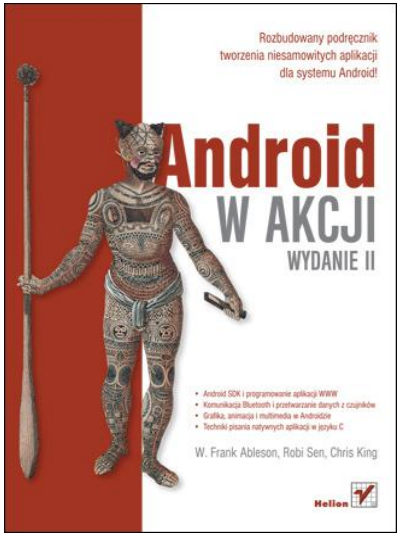
```
android:padding="20dp"
android:layout_marginLeft="10dp"
android:layout_marginRight="20dp"
android:layout_marginTop="30dp"
android:layout_marginBottom="50dp"
```

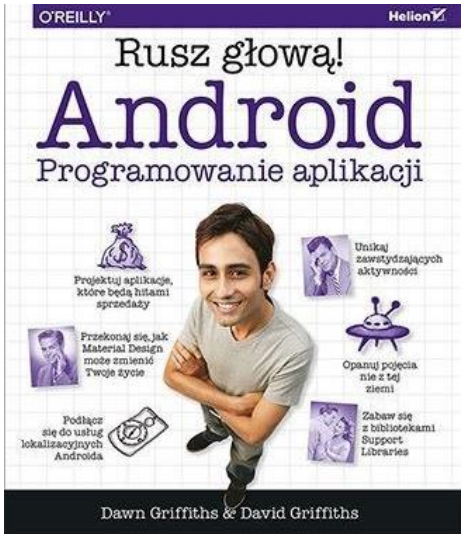


Literatura



https://developer.android.com





https://javastart.pl/baza-wiedzy/android/

https://forum.android.com.pl