1. Особености при създаване на приложения за мобилни устройства. Разпознаване на browser. Управление на viewport. Софтуерна архитектура на мобилни приложения.

Особености при създаване на приложения за мобилни устройства:
Всеки мобилен браузър поддържа някои форми на HTML.Много от тях,особено от висок клас устройства като iPhone и Windows Phone 7, поддържат последния HTML5,CSS и JavaScript стандарти и правят перфектни копия на това което ще видите в традиционен компютърен браузър.Вашият най-евтин вариант за подкрепа на мобилни браузъри е да не правите нищо.Изборът на тази опция води до много лошо мобилно браузване, но няколко причини:

Най-очевидната разлика между компютър и мобилен телефон е размера на дисплея.Средният размер на компютърен монитор е 21 инча с резолюция на дисплея не по малка от 1024 x 768.Стандартният смарт телефон е с дисплеи от 3,9 инча и резолюция по-малка от 800 x 480.Мобилният телфон не ви дава много място с което да работите.Необходимо е да използва по-малко текст и по добри икони.
Някои мобилни баузъри като Опера мини се справят с десктоп-ширината на страниците като динамично преформатират оформлението настраницата и нейните стилове.Полученият вид рядко е това, което дизайнерът е имал в предвид.Други мобилни браузъри като Сафари за iPhone или Internet Explorer за Windows Phone 7,рендват страниците с нормалните им размери и след това принуждават потребителя да увеличава на определени места и да движи уголемената страница за да прочете текста.

Ето какво трябва да запомните за големината на дисплея:
- Не разполагате с много място за работа, така че намалете екрана и опциите, също така минимизирайте размера на текста.
- Текстът който използвате трябва да бъде четлив и разбираем
- Заменете текста с икони там където е възможно
- Иконите ви трябва да изглеждат колкото се може по реалистични
- Графиките трябва да съответстват на задачите и дизайнът трябва да е ясен

Стандартният компютър има няколко начина за въвеждане на информация, но основно се използват клавиатурата и мишката.Мобилното устройство разчита главно на пръстите,като те се ползват за въвеждане на екранната клавиатура,докосване на икони и бутони, или за жестове като приплъзване или прищипване.За да компенсирате тези недостатъци, можете да предоставите звук и вибрации при натисканията на клавишите и минимизация при въвеждане на текст.Пръстите могат да са всякакви размери бутоните трябва да са достатъчно големи (35 пикселов квадрат) и визуално отделени от другите бутони или обекти за да се намелят грешките.Намерете нови начини за въвеждане на данни, като например използването на снимки.Също така трябва да се възползвате от хардуера.Windows Phone има 3 специални бутона – Back, Start, Search може да ги използвате за да поддържате реда в интерфейса и да намалите възможностите за грешка.

Разпознаване на browser:
Разпознаване на браузъра представлява процес, който най-често има за цел установяване на типа и/или версията на потребителския браузър.Осъществява се чрез код изпълняван от сървъра или от самия браузър.В по-широк смисъл този процес има за цел да установи наличието или липсата на определени(от програмиста) възможности на браузъра.
Можете да разберете дали някой посетител ползва мобилен браузър със свойството на ASP.NET – Request.Browser.IsMobileDevice. ASP.NET определя какъв браузър прави заявка и какви са неговите възможности (големина на екрана, подръжка на JavaScript и т.н.) като сравнява входящата заявка от главните низове userAgent с поредица от регулярни изрази в XML

файлове които описват общи браузъри.Информацията за съответните възможности на у-вото се съхранява в директорията : C:\Windows\Microsoft.NET\Framework\v4.0.30319\Config\Browsers (или еквивалентна в зависимост от инсталацията ви).
Със следните 2 възможности можете да подобрите разпознаването на браузър:
1.Можете да предоставяте свой собствени .browser файлове да представляват по-новите у-ва
2.Можете да използвате чужда библиотека за разпознаване на браузъри

Viewport:
Много мобилни браузъри включително Safari за iOS и Internet Explorer за Windows Phone 7, се опитват да направят заредените страници да изглеждат точно както го правят на настолен браузър.Те знаят че повечето страници са предназначени за екрани около 1000 пиксела широки и дизайнерът най-вероятно не е пригодил страницата за по малки екрани.За да разрешат този проблем обикновено те зареждат страницата на виртуално платно познато като „viewport“ обикновено около 1000 виртуални пиксела.Браузърът може да мащабира визуалния дисплей на виртуалното платно произволно, което позволява на потребителя да увеличава и движи изобразеното.

Много от най-популярните мобилни браузъри поддържат нестандартен „viewport“ мета таг, който позволява да се управлява широчината на виртуалния viewport. Например ако се добави следното към <head> отделението на страница браузърът ще разположи страницата на viewport който е 320 пиксела широк:
<meta name="viewport" content="width=320"/>
Това обикновено е много по-добър избор за мобилни телефони.Трябва да се има в предвид че има телефони с много по-висока хоризонтална резолюция.Например iPhone 4 има 640 физически пиксела на ред.Въпреки това все още има смисъл да се използва виртуален viewport от около 320 пиксела, в противен случай полученият текст ще бъде твърде малък за да се чете добре без приближаване на страницата.Може да се позволи на виртуалния viewport да варира в размер подходящ за устройството което се изполва, като добавим следния синтаксис:
<meta name="viewport" content="width=device-width"/>

Софтуерна архитектура на мобилни приложения:
Многослойната архитектура (наричана още N-слойна архитектура) е архитектура от тип клиент-сървър, в която интерфейсът, обработката на приложения и съхранението и обработката на данни са логически разделени на отделни модули. Най-разпространената форма на многослойна архитектура е трислойната архитектура.
Многослойната архитектура осигурява модел, по който разработчиците могат да създават гъвкави приложения, които могат да бъдат използвани многократно. При разделянето на едно приложение на слоеве, разработчиците имат възможността да добавят или променят отделен слой, вместо да преработват цялото приложение.
Трислойната архитектура обикновено се състои от презентационен слой (потребителски интерфейс) на най-високо ниво в приложението.Той служи за прякото взаимодействие с потребителя и изпращането на заявки към бизнес слоя. Не трябва да има директна връзка между този слой и слоят за бази данни. Бизнес слоят служи за обработка на данните и работните процеси. Той комуникира както с презентационният слой, така и с базите данни. Слоят за данни комуникира само с бизнес логиката и служи за съхранение данни и тяхното достъпване.

2. Създаване на приложения. Основни развойни средства. Емулация и изпълнение.

- Писане на приложение за Windows Phone е същото както и писане за .NET платформа
- Редактиране, компилация и debug в среда на Visual Studio
- Phone Emulator
    +Емулаторът позволява стартиране на кода в PC
    +Това е същият код, който ще се изпълнява и на телефона, но компилиран различно – за Windows PC
    +Емулацията позволява да видите работата на своята програма, как изглежда и как се държи
    +Все пак, реалното натоварване и характеристики не могат да се оценят
    +Емулаторът позволява да "движите телефона" в 3D пространството
    +Емулират се сензорните сигнали
    +Можете да записвате и повтаряте движения

Visual Studio поддържа Windows Mobile разработка на приложения от Visual Studio 5.0.
Visual Studio 2005 поддържа разработката на приложения за Smartphone 2003, Pocket PC 2003 SE, и Windows CE.
Лесно можете да добавите поддръжка за Windows Mobile 5.0 и Windows Mobile 6(Standard, Classic and Professional) като инсталирате SDKs(Software development kits) за тези платформи.SDKs включват различни инструменти, които значително подобряват процеса на разработка, емулатори, GPS, мобилен оператор емулатор и др.Visual Studio 2008 донесе няколко нови възможности и функции, както в IDE така и във функционалността.
Списъкът по долу показва някои нови функции във Visual Studio 2008:
-Поддържа тестване на елемент във Visual Studio Developer Edition и Visual Studio Team Suite – Remote Performance Monitor
-Диспечер на устройства за сигурност, част от Visual Studio 2008 IDE
-Нови Device Emulators и Device Emulator Manager
-Нов дизайн на New Project Wizard засъздаване на нови Windows Mobile проекти

Основни инструменти:
.NET CF (compact framework) е по-малък от.NET Framework, и предоставя подмножество на.NET Framework функционалността. Ето защо,повечето приложения, които се разработват за .NET CF би трябвало да работят без никакви модификации на работния плот на.NET Framework платформа.
Има две основни причини за намалената функционалност при .NET CF:
- Неподдържани десктоп функции
- Ограничена памет на мобилните устройства
Някои нови контроли са добавени и подобрени:
- Месечен календар.
Значително подобрен. - DateTimePicker.
New in .NET CF 2.0:
- DataGrid.
- LinkLabel.
- Splitter.
- WebBrowser.

Емулации:
-Device Emulator Manager –управлява емулаторите на устройството, които са инсталирани на работната станция на разработчика..

-Мрежова връзка в Device Emulator
За да се симулира реална среда, понякога трябва да има връзка с мрежата в Device Emulator.За да се направи това, трябва да се създаде приемаща мрежа, към която искате да се свържете в емулатор.
- Cellular Emulator е част от Windows Mobile SDK Tool.Той симулира присъствието на мрежа на мобилен оператор, която дава възможност на разработчиците да тестват функционалността на телефона чрез техните приложения от Device Emulator.

3. Пакетиране на готово приложение и инсталация. MarketPlace и Google Play – местата за промотиране на готови приложения. Процедура за одобрение, средства за анализ на качеството на приложение, остойностяване.

Последният етап от цикъла на развитие на проекта е да се разположи цялостния код на у-вото и да бъде оформен като цялостен пакет за дистрибуция и масово разпространение.Има няколко начина по които приложението може да бъде разположено на у-вото или емулатора:

Development IDE installer:
Най-лесният начин за разполагане на приложение е когато у-вото е свързано с разработващата работна станция чрез ActiveSync или WMDC.Първата стъпка е да се избере на устройство или емулатор ще се разполага даденото приложение.Когато направите избор може да свържете избраното у-во ръчно от лентата с инструменти или процеса на разполагане ще го направи автоматично.

Device Installer:
Не е много вероятно всяко у-во което се използва в производството да пристига при разработчика за разработка на приложение. Device Installer е метод който позволява приложенията да бъдат лесно разположени директно на у-вото, без да е необходима връзка м-у у-вото и работната станция на разработчика.Всички възли и зависимости са опаковани в един (или повече) CAB файл който се копира на у-вото и се изпълнява там. CAB изпълнението на у-вото се инсталира с предварително зададени параметри.

Windows Phone Marketplace:
-Можете да разработите и тествате приложението си на емулатора
-За да го продавате или разположите на реално устройство, следва да се регистрирате като разработчик
-Това ще ви струва $99 на година ($1 за студенти)
-Студенти могат да се регистрират като разработчици безплатно и през други среди напр. Microsoft DreamSpark

Правила в Marketplace:
-В годината на вашето регистриране, можете да публикувате неограничен брой приложения като до 100 от тях – за безплатно разпространение
-При повече безплатни - $20 за всяко
-При продажба, вземате 70% от цената
-Можете да пуснете както “demo” така и “time trialled” версии на програмите си

Одобрение в Marketplace:
-Всяка програма, предложена за продаване в marketplace, минава процес на одобрение
-Той включва: обща визия, смисленост, както и поведение на продукта. Разбира се, и документиране.

-Ако не успеете да минете процеса – получавате report и след поправка, можете да пробвате отново

Заплащания, разпространение и др. за Android:

-Google play е мястото (замени Android Market) Android приложенията се разпространяват като Android Package Files (.APK). За да Могат да се инсталират на устройство или емулатор, Android пакетът трябва да е signed. Това се прави на етап компилация в release версия. Подписът е : private release key

JDK съдържа Keytool и Jarsigner command line tools за създаването му . Можете да го изработите и чрез Android application wizard: Package Exporter →File → Export → …

За целта ще ви е нужна парола, име на приложението, организация, град и т.н. данни. В резултат - wizard компилира, подписва, компресира в zip пакета.

Алтернативи на Google play са: OEM, Amazon App Store …. За момента, Google Play е най-големият – повече от 1 200 000 apps и 40 милиарда downloads от потребители на 130 страни.

Работа с Google Play:

1. Създава се developer profile;
2. Подписва се договор
3. Плаща се регистрационна такса ($25.00)

Upload на приложението ви става като signed release package. След това качвате и своите application’s assets и други съпътстващи модули:

-Описание на екранни съответствия
-Икони, резолюции
-Поддържана графика..
-заглавие
-описание
-Категория
-Съпътстващ website
-Email
-Поддържани телефони

Вашето приложение се листва , заедно със съпътстващи детайли като: брой потребители, rating, коментари Има и Statistics. Оттам ще намерите информация за: Използвани платформи от потребителите ви; апарати; страни; езици….

Има и Android Developer Console, от където можете да наблюдавате error reports от различните си потребители (crashes or freezes).

Заплащане за приложение:

Налични са 3 опции:

1.Платено приложение
2.Free приложение, с In-App Billing (IAB) – самият download и инсталация са free, но се таксуват upgrades, опционни добавки, и др (обикновено най-необходимите)
3.Платени реклами: самото приложение е free, но се заплащат рекламите

Следене на статистика и анализи:

Google Analytic и Flurry са някои от ползваните продукти за проследяване кой ползва и как вашето приложение.

-User analytics: географско разпространение, езиково, скорост на Internet connections, ползвани screen sizes или резолюции, ..
-Application usage patterns: следи ползването на отделните Activity, сайтове, действия, прогрес на играчите (за игри) и т.н.

-Exception tracking Предлагат се и SDK за Google Analytics за мобилни устройства. След като госвалите, SDK може да се включи в проекта ви. Google Analytic Library изисква постоянен Internet достъп

За да ползвате Google Analytics service в приложението си трябва:
-Да създадете инстанция на услугата: GoogleAnalyticsTracker tracker = GoogleAnalyticsTracker.getInstance();
-За да започнете процеса на наблюдение: tracker.start("your code",this);
-За всяко действие, което искате да проследявате, повикайте trackerPageView() с подходящия параметър За мобилни, събитията се записват (logged) на телефона, така че с tracker.dispatch(); можете да ги изпратите (upload) в batch режим, когато решите.

4. Управление на състоянието при мобилни приложения с разкъсване на сесия и stateless протокол. Методи за запазване на сесийното състояние.

Управление на сесийно състояние:
ASP.NET предлага осъвременена и подобрена версия на сесийния обект.Този обект позволява да се изпълняват следните задачи:
-Съхранява информация специфична за потребителската сесия.
-Управлява сесийния живот през eventHandler и методи.
- Освобождава несесийните данни след определено изчакване.
Сесиното свойство на System.Web.HttpApplication класа (родителския клас на Global.asax страница) и сесийното свойство на MobilePage клас (родителския клас на мобилния си уеб форми(страница)) така ще се даде достъп до сесийния обект.Обикновено, вие ще манипулирате обекта Session или в:
1.code-behind module на Global.asax файла на вашето приложение
2.code-behind module на вашата мобилна уеб форма (страница).

Често се налага да съхраните информация за времето между client requests и server responses. За съжаление, Hypertext Transfer Protocol (HTTP) е изначало stateless, т.е. не пази такава информация. В миналото, за целта се ползваха предимно cookies , пазещи съответния session ID, както и подходяща информация, която сървърът би поискал за този потребител

Описани са 4 метода на ASP.NET за съхранение на състоянието:
1. Session state:
Позволява съхранение на клиентски променливи и обекти за време между множество заявки и отговори в рамките на сесия. ASP.NET предоставя подобрена версия на обекта Session. През него могат да се изпълняват следните задачи:
-Идентификация на user чрез уникален session ID.
-Съхраняване на собствена за потребителската сесия информация.
-Управление на сесийното състояние между повиквания на event handler methods.
-Освобождаване на сесийно обвързани данни след определено време
Пропъртито - Session на класа System.Web.HttpApplication (родителски за страницата Global.asax) както и Пропъртито - Session на класа MobilePage (родителски за страницата Web Forms) - и двата дават достъп до обект Session.
Обикновено се нуждаете от обекта Session или в
1.code-behind модула на приложението (файла Global.asax) или
2.code-behind модула на вашата страница (mobile Web Forms page). Както и в Web Forms страниците, така и файлът Global.asax поддържа модул - code-behind. Този модул е с име - global.asax.extension, където -extension указва ползвания езика. Например: за C# , code-behind модулът е Global.asax.cs. Следва фрагмент код, който добавя низ с user's start time към обект

Session с ключ - UserStartTime (от код на файла Global.asax). Показан е и начин за добавяне на елементи към обекта Session: (ползване на метода Add за достъп до ключ HelpAccess)
2. Hidden променливи:
Методът позволява съхраняване на обекти за време заявка/отговор (server round-trips) чрез предаване на нужните данни в скрити полета. **Виж т.5**
3. View state Позволява съхраняване на стойности от mobile Web Forms страници в сървъра. Тази информация се пази през инстанции на класа StateBag, който се създава в рамките на сесията. Сървърът после връща информацията към клиента. **Виж т.6**
4. Application state Подходът позволява съхраняване на променливи и обекти на приложението между множество заявки от различни клиенти. **Виж т.6**

5. Използване на Cookies за запазване състояние. Подход със скрити променливи.

Ползване на Cookies:
Cookies предоставят начин на Web servers да идентифицират клиента, свързал се в момента Потенциалът на cookies е лимитиран, особено когато става дума за приложения с безжични клиенти. Това е така, защото много от безжичните протоколи, в това число Wireless Application Protocol (WAP) не поддържат cookies. Ако сте сигурни, че устройството ви поддържа cookies или текущото proxy ги поддържа (което е така при някои WAP gateways), то cookies са си превъзходно средсво за идентификация на сесиите.

Скрити променливи:
Понякога може да ви се налага да прехвърляте малки обеми информация между отделните Web pages , без да използвате session state. Например, събирате във формата информация за потребителя. С HTML, можете да прехвърлите тази информация към друга страница – ползвайки скрито input поле (hidden). Пропъртито на класа MobilePage - HiddenVariables предоставя тази функционалност. То позволява да съхраните двойка name-value , за променлива, която впоследствие средата предава към/от сървъра и клиента във формата на hidden полета. Подходът е добър само при малки обеми данни.
Пример : … HiddenVariables.Add(TextBoxName.ID,TextBoxName.Text);

6. Универсален метод с View State за запазване на състоянието. Запазване състояние на ниво-приложение.

View State:
ASP.NET създава усещането, че състоянието на страниците се запазва при round-trips. Всъщност, страницата не се запазва между отделните заявки/ отговори, но runtime средата запазва properties на страницата в т.нар. server control view state (инстанция на класа StateBag). Когато потребител подаде заявка, runtime автоматично реконструира страницата с използване на стойностите от пропъртитата, пазени в инстанцията на StateBag. Например, дефинирали сте property във вашия code-behind клас. Това пропърти автоматично не се съхранява/възстановява всеки път, когато страницата се реконструира. Ако сте установили това property в кода си при някоя заявка, вие сигурно очаквате стойността му да е съхранена между round-trips през сървъра . Разбира се, можете да добавите това property към сесията или да го запазите с помощта на hidden variables. Ако ползвате ViewState property на MobilePage за съхраняване стойноста, то runtime автоматично поема грижата за save/ restore на стойността, вместо вас. Ако ползвате сесия със view state, следва да помните 2 неща: първо, sessions може да се унищожи (expire), което означава че вие губите вашия view state . Минутите, допустими до response от клиента се указват с атрибута – timeout на елемент sessionState във файла Web.config- 20 min са по подразбиране.

<sessionState mode="inProc" cookieless="true" timeout="20" />
Второ, изобразяваната страница в клиента и текущото състояние на сесийната информация, пазена на сървъра, могат да излязат извън синхронизация. Това може да се случи, при операция Back на browser с цел – връщане към предишна страница (например с бутона Back). Нека предположим, че потребителят е в началната страница на приложението и след това отиде към втора. Ако впоследствие, потребителя навигира обратно към първата страница, то той ще види съдържанието ѝ , докато сървърът пази session data за втората. Mobile Internet Toolkit решава този проблем като пази и малка история на view state информацията в потребителската сесия. Можете да конфигурирате размера на view state history. Подразбиращ се е 6. за да го промените – ползвайте sessionStateHistorySize атрибут на елемент mobileControls в Web.config файл:

```
<configuration>
        <system.web>
                <mobileControls sessionStateHistorySize="10"/>
        <system.web>
 </configuration>
```

Application State:
В ASP.NET, приложението е това, което обединява всички файлове, които runtime средата ще вика в рамките на virtual directory и всички нейни subdirectories. Можете да създадете variables и objects имащи обсег – цялото приложение, а не само сесиен. Класът HttpApplicationState позволява да направите това. Обектът Application е базовият за да направите инстанция на HttpApplicationState и да го експонирате през пропъртито Application на класа System.Web.HttpApplication (родителски на страницата Global.asax) и През пропъртито Application на класа MobilePage (родителски на страницата mobile Web Forms).
Използване на Application State в Global.asax Можете да дефинирате информация на ниво приложение във файла Global.asax, който винаги е в корена на виртуалната директория. Global.asax е мястото за дефиниране на event handlers, свързани с application state. Там се пази и друга информация, например session state. Дефиниране на application state data през code-behind файла за Global.asax е удачно да се прави в двата event handler метода: Application_Start и Application_End. Ако се чудите дали да ползвате session state или application state:

1. Информацията в application state е memory hungry. С други думи – приложението пази цялата application state information в паметта и не я освобождава дори ако потребителя напусне приложението.
2. Всяка нишка в многонишково приложение може да ползва application data едновременно, тъй като ASP.NET автоматично не охранява ресурсите.

7. Програмиране за Windows Phone . Silverlight и XNA архитектура. Терминология. Апаратна поддръжка. Помощни програмни средства.

Типове приложения:
Windows Phone платформата поддържа 2 отделни frameworks за разработка:
-Silverlight framework позволява реализация на event-driven, XAML-базирани приложения. Също така дава на Web програмиста силни средства за развой на интерфейси с микс от традиционни контроли, текстови елементи с високо качество, векторна графика, media, animation, и data binding - всички те работещи над много платформи и browsers. Windows Phone разширява възможностите на стария Silverlight към ползване над мобилни устройства.
-XNA Framework се използва при разработка на игрови приложения.
Това е платформа за развой на игри на Microsoft с поддръжка на 2D ефекти и 3D графика. XNA е предназначена предимно за създаване на games за Xbox 360 console. Разработчиците могат да я ползват и към приложения за PC, както и audio player, Zune HD и др.

Долната таблица листва критерии за определяне най-подходящата развойна среда (Silverlight или XNA Framework).

Text-based controls and menus	Silverlight
Event-driven application	Silverlight
Interaction with Windows Phone controls such as Panorama	Silverlight
Embedded video	Silverlight
Hosted HTML	Silverlight
Web browser compatibility	Silverlight
Vector graphics	Silverlight
Looping game framework	XNA
Visually complex applications	XNA
3D games	XNA
Advanced art assets such as textures, effects, and terrains	XNAX

Ще ползваме Silverlight. Приложенията се състоят от markup описание и код. - markup частта е на Extensible Application Markup Language, или XAML и се произнася “zammel.” XAML служи за дефиниране на потребителския layout, controls и panels. - Code-behind файловете изпълняват както initialization и логически обработки, така и handling events от контролите.

Апаратна поддръжка:
Windows Phone изисква минимален hardware за поддръжка на приложенията : WVGA (800 x 480) дисплей. Capacitive multi-touch screen. DirectX 9 hardware графични ускорители. Сензори: GPS, accelerometer, compass, light, and proximity. Цифрова камера. Наличие на бутони Start, Search, Back. Поддръжка за data connectivity с клетъчна мрежа и Wi-Fi. 256 MB (или повече) RAM и 8 GB (или повече) flash памет.

Терминология:
- Code named Metro design: Това е user interface (UI) ползван в Windows Phone. Разработчиците следва да следват този дизайн.
- Tile: това е визията, която се явява на стартовия екран. Tile визията е проектирана за изобразяване на динамична информация на дисплея.
- Status Bar: поддържа текущия статус по време на работа – например сила на сигнала. Не е задължително да е application specific.

Програмни средства за създаване на приложения:
С инсталацията на Windows Phone Developer Tools, получавате следните free tools и components: Expression Blend for Windows Phone; Visual Studio for Windows Phone; Windows Phone emulator; XNA Game Studio; Silverlight; Zune software (for deployment); .NET Framework и други. Ако имате вече инсталиран Visual Studio 2010+ (Professional или Ultimate), то можете да ползвате средата на Visual Studio за разработка , разбира се след инсталирани Windows Phone Developer Tools.

8. Създаване на приложение за Windows Phone (Silverlight структура). Основни елементи на приложението и тяхната свързаност.

Start->Microsoft Visual Studio 2010 Express for Windows Phone->New Project(Windows Phone Application template)->Finish
Приложенията на Silverlight са комбинация от markup(за интерфейса)и код. Използва се Extensible Application Markup Language или XAML, като се произнася „замел“. XAML предимно дефинира оформлението на потребителския интерфейс, неговите контроли и панели. Code-behind файловете също могат да извършват някаква инициализация и логика, но предимно се използват за поддръжката на събитията, свързани с контролите.
Файловете, който се използват при разработването на стандартно приложение са MainPage.xaml и MainPage.xaml.cs. MainPage.xaml дефинира потребителския интерфейс на приложението. XAML е XML-базиран декларитивен език, използван за създаването и оформлението на елементите на потребителския интерфейс. При разгръщането на MainPage.xaml се вижда C# code-behind файл на име MainPage.xaml.cs. Този файл е съединен с XAML файла чрез частичен клас, който съдържа логиката за XAML файла.Разделянето на потребителския интерфейс от кода позволява разделното създаване на интерфейса и логиката зад него, правейки работата на дизайнерите и програмистите по-лесна.
Когато се пусне програма App Class декларира обект от типа PhoneApplicationFrame.
Тази рамка е широка 480 пиксела и 800 пиксела висока и заема цялата повърхност на дисплея на телефона. Обектът от тип PhoneApplicationFrame после се държи малко като уеб браузър като навигира до обект наричан MainPage (главната страница).

9. Добавяне на текстови и графични компоненти. Създаване на UI. Нива на вложеност и преходи. Пример.

Добавяне на TextBlock:
Нека добавим прост TextBlock който ще изобрази съобщението "Hello, World!"
- Ако MainPage.xaml не е вече активиран, double-click MainPage.xaml в Solution Explorer.
- В менюто View , избирате опция Other Windows → Toolbox. Появява се Toolbox прозореца.
- Resize или pin на този прозорец (Toolbox) – ще можете да наблюдавате едновременно Toolbox и телефона в Design view.
- От Toolbox, с влачене поставяте TextBlock в main panel на телефона . гледайки XAML view, забелязвате TextBlock елемент, добавен в Grid content panel.
- От View menu, избирате Other Windows и след това избирате Properties Window.
Можете да редактирате properties на TextBlock. Например : FontSize="36" Всички размерности в Silverlight са в pixels. Специфицирайки 36, вие имате шрифт, който от дъното на горен ред, до върха на долен се разпростира на 36 pixels. Шрифтовете са всъщност по-сложни. Нашият TextBlock ще има височина повече от 48 pixels—около 33% по-голяма от указаната във FontSize. Това допълнително разстояние (наричано leading) служи да гарантира, че редове няма да се припокриват. Традиционно, font sizes се обозначават в points. Point е около 1/72 от inch. Как става преобразуването от pixels → points? На 600 dots-per-inch (DPI) принтер, например, 72-point шрифт ще бъде 600 pixels висок.
Дисплеите обикновено имат резолюция около 100 DPI. По подразбиране, Windows приема , че видео дисплеите имат резолюция 96 DPI. При това предположение, font sizes и pixels са свързани със следните формули: points = 3/4 × pixels; pixels = 4/3 × points

Добавяне на графика:
Добавяте графика през класовете Shape . Можете да създадете прости форми:
-напр. Rectangles, или по-сложни – напр. Polygons.
-Brushes служат за оцветяване на обект. В началото поставяте т.нар. StackPanel около обекта TextBlock. Panel е контейнер, групиращ и подреждащ UI елементите. Всяко приложение следва да има поне 1 Panel. StackPanel подрежда всеки елемент последователно, или вертикално, или хоризонтално – според Orientation. Grid и Canvas панелите позволяват по-точно позициониране на елементи. Ще създадем Ellipse. Елипсата ще се появи след TextBlock в StackPanel. Ще укажем Height и Width за Ellipse, както и Fill. За Fill ще специфицираме Brush . Можете да ползвате както Design view, така и XAML view: - В XAML view, намирате добавения TextBlock. - Дописвате следния XAML.

```
<StackPanel>
        <TextBlock FONTSIZE="50" TEXT="HELLO, WORLD!" />
        <Ellipse Fill="Blue" Height="150" Width="300” Name="FirstEllipse" />
</StackPanel>
```

Създаване на UI:
Главната страница се състои от една панорама Panorama с три PanoramaItems(панорамни елемента).Първия елемент се състои от ListBox които предоставя главно меню за приложението. Когато ползвателя избере един от елементите в ListBox-а ние навигираме до кореспондиращата страница- това е страницата с колекцията за или Рецепти или Факти или Коктейли.Точно преди да навигирате ние трябва да се подсигурим, че сме заредили кореспондиращите данни в Рецепти,Факти и Коктейли(части от приложението за мангото):

```
switch (CategoryList.SelectedIndex)
{
        case 0:
        App.ViewModel.LoadRecipes();
        NavigationService.Navigate(    new Uri("/RecipesPage.xaml", UriKind.Relative));
        break;
        ... additional cases omitted for brevity
}
```

Когато ползвателя избере един елемент от списъка с сезони акценти (Seasonal Highlights) в UI,ние разглеждаме избрания елемент за да видим дали е рецепта или коктейл и да се насочим до индивидуалната страница за Коктейл или Рецепта, преминавайки през ИД то на елемента като част от заявката за навигиране:

```
SeasonalHighlight selectedItem = (SeasonalHighlight)SeasonalList.SelectedItem;
String navigationString = String.Empty;
if (selectedItem.SourceTable == "Recipes")
{
 App.ViewModel.LoadRecipes();
 navigationString = String.Format("/RecipePage.xaml?ID={0}", selectedItem.ID);
}
else if (selectedItem.SourceTable == "Cocktails")
{
 App.ViewModel.LoadCocktails();
 navigationString = String.Format("/CocktailPage.xaml?ID={0}", selectedItem.ID);
}
NavigationService.Navigate(new System.Uri(navigationString, UriKind.Relative));
```

Потребителя може да навигира от менюто на главната страница до един от трите страници със списъци. Всяка от тези страници се свързва по данни с една от колекцията в ViewMode за да покаже списък от елементи: Рецепти,Факти и Коктейли. Всяка от тези страници предлага прост

ListBox където всеки елемент от списъка съдържа конторил за изображение за снимката и TextBlock за име на елемента Фигурата показва FactsPage:
Когато потребителят избере индивидуален елемент от списъка с Рецепти,Факти или Коктейли, ние го пращаме до идивидуалната страница за Рецепти,Факти или Коктейли минавайки по ИДто на индивидуалния елементи и заявката за навигация. Отново тези страници са почти еднакви, като предлагат в себе си Изображение и някъв текст доло.
Кодът зад всяка страница е почти идентичен и прост. В OnNavigatedTo() отмяната ние извличаме индивидуалния ИД на елемента от низа на заявката, намираме този елемент от ViewModel колекцията и го правим свързан по данни.
Кодът за страницата рецепти RecipePage е малко по сложен от другите- добавения код в тази страница изцяло е свързан с HyperlinkButton бутонът позициониран на горния десен ъгъл на страницата:
Когато потребителя натисне копчето закрепи HyperlinkButton на индивидуалната страница на рецептата ние закачаме този елемент като плочка към стартовата страница на телефона. Действието закачане изпраща потребителя към началната страница и деактивира приложението. Когато плочка е закрепена по този начин тя се анимира периодично обръщайки се между отпред и отзад както е показано:
Потребителят може да натисне на закрепената плочка която навигира директно до елемента в приложението. Когато достигне страницата бутона“ прикрепи“ ще има изображение „откачи“ . Ако той окача страницата той ще бъде премахнат от стартовата страница, и приложението ще продължи.
За да стане това програмистът трябва да :

1. Дефинира таг в настоящата страница маркиран като „ прикрепен“
2. Дефинира код обработчик на събитиети PinUnpin_Click() дефиниращ начина и заместването на „Откачането“

10. Добавяне на бутон и обработчик (handler). Добавяне на exception handler. Публикуване на готово приложение в Marketplace.

Добавяне на Button:
Нека добавим button от тип Control. Silverlight притежава богата библиотека от control с Button, TextBox, ListBox, и много други. Има 2 етапа на това добавяне. Първият е добавяне на елемент Button към XAML описанието. Вторият е да се добави логика за обработка на събитията , генерирани при взаимодействие на потребител – например – click за Button. - В XAML view, допишете следния XAML след <Ellipse /> tag:
<BUTTON HEIGHT="150" Width="300" Name="FirstBUTTON" Content="Tap" />
Visual Studio създава автоматично event handlers - В Design View, селектирате бутона.
- в Properties window, избирате Events tab. Списък от събития се появява за бутона . Селектирате нужния event .
Файлът с code-behind кода (MainPage.xaml.cs) се отваря и откривате FirstButton_Click event handler.

Публикуване в Marketplace:
След като завършите приложението, за да го публикувате за free download или продажба, следва да го изпратите в Windows Phone Marketplace
Това става през App Hub, където минавате сертификационен процес. След сертифициране на приложението, се генерират Marketplace pages в инсталирания при вас софтуер - Zune.

11. Особености при изобразяване на мобилен дисплей. Видове дисплеи и софтуерни решения.

При първоначалното пускане на Windows Phone 7, устройствата трябва са с размер на дисплея 480 × 800 пиксела. В бъдеще, е възможна и поддръжката на екрани с размер 320 × 480.
Тези размери са познати като „голям“ и „малък“. Най-големият общ знаменател на хоризонталните и вертикални размери на двата дисплея е 160, поради което могат да се представят като сбор от множество квадрати с размер 160x160 пиксела. Разбира се, телефоните могат да се завъртат в панорамен режим, като някои приложения могат да изискват определена ориентация.
Всички размери в Silverlight са в пиксели. При размер на шрифта 36, се оказва, че реалният размер е 48 пиксела, поради разстоянието, необходимо за предотвратяване на сливането на текста. Обикновено размерите на шрифта се изразяват в точки, като една точка е близо 1/72-ра от инча. Преобразуването от пиксели в точки става по следния начин:
При принтери с 600 точки за инч(DPI), 72-точков шрифт ще е 600 пиксела! Обикновено, дисплеите са с около 100 DPI(най-често 96), което определя размера на точките- 1 точка= ¾*пиксел, а 1 пиксел= 4/3*точка.
При мобилните устройства с високи резолюции се стига до DPI с размери 264(за 480x800).

12. Обработка на изключения в приложение. Съобщения на екрана. Вграждане на ресурси. Навигация между страници в приложение.

Обработка на изключения в приложение:
Windows Phone платформата поддържа обработката на изключения, като за целта се използват try-catch блокове. В try се изписва фрагментът от кода, при изпълнението на който е възможно възникването на изключение, а в catch блока се дефинира какво трябва да се случи ако възникне изключение (извеждане на съобщение за грешка и пр.). Възможно е и добавяне на блок finally, в който се дефинират действия които да се изпълнят след минаването през try или try и catch блоковете.

Съобщения на екрана:
Съществуват два типа съобщения: обикновени съобщения и съобщения с възможност за отговор от страна на потребителя. Обикновените съобщения извеждат някакъв стринг на екрана, като този стринг може да се използва за осведомяване на потребителя за извършването на някакво действие. Синтаксисът им е следния: MessageBox.Show("Стринг");
При извеждане на такова съобщение на потребителя е възможно само да натисне бутон ОК след като е прочел текста, съдържан в стринга.
Втория тип съобщения са такива, при които потребителя има възможност да направи избор. Синтаксисът е следния: MessageBox.Show("Текст_на_съобщението", "Заглавие_на_съобщението", MessageBoxButton.OKCancel); При този тип съобщения, потребителя може да избере бутон ОК или Cancel и в зависимост от това приложението да изпълнява различни методи. Този тип съобщения връщат като резултатен тип MessageBoxResult, като по този начин може да се зареди променлива и да се използва в кода.

Вграждане на ресурси:
Вградените ресурси са външни за проекта файлове (текстови и графични), които се импортират в проекта на приложението. Това става с десен клик върху името на проекта Add→Add New Item, след което се избира файлът, който ще се импортира. След това се отварят свойствата (properties) на ресурса и се задава стойност Embedded Resource на свойството Build Action. По този начин, вградените файлове могат да бъдат използвани свободно в рамките на приложението.

Навигация между страници в приложение:
За навигация между страници в приложение се използва класа NavigationService и по- точно функцията му Navigate. Синтаксис: NavigationService.Navigate(new Uri("/име_на_страница.xaml", UriKind.Relative)); За връщане назад: NavigationService.GoBack();

13. Работа с данни. Източници на данни. Терминология. Пример за привързване на данни към контрол. Едно и дву-посочно привързване.

Работа с данни:
.NET CF поддържа работа с различни източници на данни.Концепцията е същата като при .NET Framework и най-важното е че ADO.NET се поддържа в по-голямата си функционалност. Windows Mobile приложенията могат да използват SQL Server CE, XML, SQL Server, уеб услуги като източници на данни.XML е общ формат, при който повечето източници на данни могат да внасят и изнасят данни.Въпреки че може да се използва от повечето източници на данни, не се препоръчва като средство за съхраняване на данни, защото има голям излишък, не е оптимизиран и е много по-бавен от релационна база данни, защото няма индекси.Въпреки това, ако няма друга възможност XML може да се ползва за съхранение на данни на у-вото.SQL Server може директно да бъде достъпен като средство за съхраняване на данни.Уеб услугите са добър избор за прехвърляне на данни в n-редна среда, когато няма директен достъп до SQL Server (за директен достъп до данни или за копиране).

Локални файлове:
Има текстови и XML файлове. Локалните файлове могат да се компилират като resource или content.
-Ресурсни файлове: Resource files се вграждат в пакета на проекта (.xap). Предимствата на използване на resource file е че той винаги е заедно с приложението. За сметка на това, приложението се стартира по-бавно. Достъпът до ресурсни файлове е през Application.GetResourceStream(). Използвате ресурсен файл ако:

+ Не се интересувате от application startup time.
+ Не е нужен update на ресурсния файл, дълго време след компилацията.
+ Искате да опростите дистрибутирането на приложението си, чрез намаляване на между-файловите връзки.

- Content файлове: С оглед на производителността - content файловете са предпочитани пред resource files. Content files се включват в пакета на приложението (.xap) . Въпреки , че те не са компилирани в асемблито, в метаданнововто описание е отразена връзката с content файла. Достъпът до content file се осъществява например чрез : Xelement.Load()

Пример за привързване на данни към контрол:
Data Binding на контрол на елемент. Пример. Следният код показва пример за свързване на контрола на една единица данни. Целта е свойството на Text на контрола textBox. Източникът е от класа на информация за музика:Recording.
XAML

```
<GRID X:NAME="CONTENTPANEL" GRID.ROW="1" MARGIN="12,0,12,0">
<TextBox VerticalAlignment="Top" IsReadOnly="True" Margin="5" TextWrapping="Wrap"
Height="120" Width="400" Text="{Binding}" x:Name="textBox1" />
</ Grid>
```

Code-behind

```
public MainPage() {
InitializeComponent();
// Set the data context to a new recording
textBox1.DataContext = new Recording("Chris Sells", "Chris Sells Live",
```

```
new DateTime(2008, 2, 5)); }
// A SIMPLE BUSINESS OBJECT
public class Recording {
{ public Recording() { }
public Recording(string artistName, string cdName, DateTime release) {
Artist = artistName; }
Name = cdName;
ReleaseDate = release;
public string Artist { get; set; }
public string Name { get; set; }
public DateTime ReleaseDate { get; set; }
// Override the ToString method.
public override string ToString()
{ return Name + " by " + Artist + ", Released: " + ReleaseDate.ToSortDateString(); } }
```

Двупосочно привързване (Two-Way Binding):
Ако сменяте стойност в полето Name (тип TextBox), сигурно бихте желали и обратно - да можете да промените и съответните данни от DataContext обекта. Тогава - two-way binding: Следва да модифицираме програмата – искаме two-way binding над Name property. Правим следното: <TextBox x:Name="Name" TextWrapping="Wrap" d:LayoutOverrides="Height" Grid.Column="1" HorizontalAlignment="Left" Width="200" VerticalAlignment="Center" Text="{Binding Name, Mode=TwoWay}" />

14. Работа с Isolated storage. Съхраняване на файлове и settings.

Isolated Storage:
За съхраняване и ползване на user-specific information, може да се ползва и isolated storage. В приложенията за Windows Phone няма директен достъп до файловата система на ОС. Все пак, можете да използвате isolated storage за съхранение и ползване на данни, локално на устройството. Има 2 начина за ползване на isolated storage:

- Записвате/четете данните във формат key/value с помощта на клас IsolatedStorageSettings
- Записвате/четете файлове с клас IsolatedStorageFile.

- This storage is called isolated because each application on the phone has its own area. One application cannot access the storage of another
- The data is stored in the mass storage of the phone. A program can store very large amounts of data in isolated storage.
- A program can create and use as many files as the application requires
- It is also possible to create folders within isolated storage so an application can organise data as required
- The data will be persisted when the application is not running
- If the application is removed from the phone all its isolated storage is deleted

Метод за запис във файл в Isolated Storage:

```
private void saveText(string filename, string text) {
   using (IsolatedStorageFile isf = IsolatedStorageFile.GetUserStoreForApplication() {
      using (IsolatedStorageFileStream rawStream = isf.CreateFile(filename)) {
         StreamWriter writer = new StreamWriter(rawStream);
         writer.Write(text);
         writer.Close(); } } }
```

Метод за четене от файл в Isolated Storage:

```
try {
  using (IsolatedStorageFileStream rawStream = isf.OpenFile(filename, System.IO.FileMode.Open)) {
     StreamReader reader = new StreamReader(rawStream);
     result = reader.ReadToEnd();
     reader.Close(); } }
catch {
  return false; }
```

Your applications can create many files in isolated storage. They can also build up a directory hierarchy within the storage. You can perform stream based input/output with files in the isolated storage.

Using Settings Isolated storage:

- Creating files in isolated storage is useful, but often a program only wants to store name/value pairs
- Examples of this:
  - Username
  - Home directory
  - Color and display preferences
- The Isolated storage in Windows Phone also provides setting storage
- The settings storage works like a Dictionary collection
- A Dictionary holds a collection of a particular type which is indexed by key values of another type
- Programs can look up items based on the value of the key

Метод за запис в Isolated Storage Settings:

```
private void saveText(string filename,  string text) {
 IsolatedStorageSettings isolatedStore = IsolatedStorageSettings.ApplicationSettings;
 isolatedStore.Remove(filename);
 isolatedStore.Add(filename, text);
 isolatedStore.Save(); }
```

- You can save objects other than strings
- Each object must have a unique name
- Your program must call the Save method to persist the settings information when it has been added to the settings object

Метод за четене от Isolated Storage Settings:

```
private bool loadText(string filename,  out string result) {
 IsolatedStorageSettings isolatedStore = IsolatedStorageSettings.ApplicationSettings;
 result = "";
 try {
    result = (string)isolatedStore[filename];  }
 catch  {
    return false;  }
 return true; }
```

- Reading is the reverse of writing
- Your program must provide the key of the item it wants to load
- Note that the saved item will be returned in the form of an object reference which your program must cast to the required type
- The settings storage does not provide a ContainsKey method

15. Web- услуги, протоколи и технологии около web- услугите за мобилни приложения.

Терминология:
При използване на web services , имате : услуги, формати, технологии. Следват някои термини , специфични за web services.
- web service: Units of application logic that provide data and services to other applications. Applications access web services using standard web protocols and data formats such as HTTP, XML, and SOAP, independent of how each web service is implemented.
- REST: (Representational State Transfer Protocol) . A protocol for exposing resources on the web for access by clients.
- POX: (Plain Old XML) A term used to describe basic XML.
- JSON: (JavaScript Object Notation) A lightweight format for exchanging data. It's designed to be human-readable, but also easily parsed by a computer.
- OData: (Open Data Protocol) A web protocol for querying and updating data.
- SOAP: (Simple Object Access Protocol) A lightweight protocol intended for exchanging structured information in a decentralized, distributed environment.

Технологии при Web услугите:
В Silverlight за Windows Phone може да ползвате : HTTP класове WCF услуги WCF Data Services (OData services) Windows Azure Services HTTP Classes Можете директно да ползвате web services или resources из мрежата директно с: Класовете :HttpWebRequest / HttpWebResponse или WebClient Тези класове имат функционалност за изпращане на заявки към web услуга с използване на HTTP протокол. Silverlight не позволява хостване на HTTP-базирани услуги, така че тези класове са полезни когато се ползва съществуваща web service. HTTP услугата следва да е хоствана някъде и отговорът ще е XML или JSON. Ако желаете да създадете своя web услуга, Silverlight позволява това през WCF.

WCF услуги:
Windows Communication Foundation (WCF) е framework за създаване и достъп до услуги. WCF позволява да експонирате клас като услуга и изпълнявате обмена на обекти между Silverlight и тази услуга. В Silverlight за Windows Phone application ползвате SLsvcUtil.exe tool или Add Service Reference на Visual Studio Те създават локален proxy клас за тази услуга. Proxy класът позволява достъп до услугата, както ако тя е в локален клас.

WCF Data Services (OData услуги):
WCF Data Services, известни и като ADO.NET Data services, е framework за достъп до данни през т. нар. : representational state transfer (REST) протокол. WCF Data Services управлява целия HTTP обмен, сериализация и други задачи по експониране на данни като услуга. Приложението може да ползва тези данни през стандартния HTTP protocol , да провежда заявки, да изпълнява create, update и delete над данните от data service, в същия domain или в друг domain. OData функционалността за Windows Phone се поддържа от OData Client library.

Windows Azure Storage Services:
Ползвате Windows Azure за съхранение и възстановяване на данни за вашето Windows Phone приложение, в случаите на недостиг на запомняща среда в устройството . Windows Azure storage services създават persistent, durable storage в cloud и позволяват поддръжка на неограничен по обем, еластичен, запомнящ масив. Достъпът до Windows Azure storage е подобен на достъпа до web service.

16. Привързване към база данни. Създаване на таблици през код. Вградени заявки и обработки от приложението.

Локална БД и LINQ to SQL Secondary Tiles:
Последователност от задачите по създаване на приложението:
- Създаване на базово приложение в среда на Visual Studio: Ползваме Windows Phone Silverlight and XNA Application template.
- Отделно създаване на база данни с рецепти, коктейли и факти.
- Подобряване на приложението - свързване и ползване на базата данни (data binding).
- Създаване на нови UI страници и привързване с източник на данни за всяка. - Указване на т.нар .''Secondary Tiles'' с цел да се позволи потребител да фиксира (pin) определена рецепта към стартовата страница на телефона.

Имаме поддръжка от страна на Windows Phone SDK на local databases. Т.е. приложението може да съхранява данни в локален файл (local database file - .sdf) на телефона. създаването на базата данни може да е:

А. В кода, като част от самото приложение или
Б. като отделно помощно приложение.

За да създадем БД в кода си – следва да имаме клас , производен на стандартния DataContext . Класът е същия DataContext , който ще ползваме както в помощното приложение, което създава БД, така и в основното приложение, консумиращо я. В помощното приложение, следва да укажем местоположение на БД - isolated storage, тъй като това е единственото място, където можем да пишем от код на приложение на телефона. Класът съдържа набор полета от тип Table по едно за всяка таблица на БД. Атрибутите Column описват колоните на БД както и database schema properties като например data type или размер (INT, или друг), дали колоната позволява нулева стойност, дали е ключ и т.н. Дефинираме класове Table за всяка отделна таблица на БД по същия начин.
В помощното приложение трябва да опишем и клас ViewModel - посредник между View (потребителския интерфейс) и Model (данните) като за целта използваме класа DataContext. Във ViewModel има поле DataContext и множество колекции за отделните данни в таблицата (Recipes, Facts или Cocktails). Данните са статични, така че колекция от тип List<T> е достатъчна. За същата цел - се нуждаем и от get property accessors, не от set modifiers. Трябва да подготвим и public метод, който ще се вика в UI — за създаване на БД и всички данни в нея. В метода ще създадем базата (ако вече не е била създадена) и после – поотделно всяка таблица . Ще ги попълним със статичните данни.
Например, за да създадем таблица Recipe:
1.Създаваме множество инстанции на класа Recipe, кореспондиращи на редовете в таблицата;
1.Добавяме редовете към DataContext;
2.Накрая „commit“ на данните към БД. Същата последователност трябва да се приложи и към таблиците Facts и Cocktails.
На подходящо място в помощното приложение —например button click handler—ще повикаме този метод (CreateDatabase). При стартиране на помощното приложение, файлът с БД ще се създаде и разположи в isolated storage. Последната ни задача е да извлечем този файл на десктопа, така че да можем да го използваме от приложението в режим емулация. Ползваме Isolated Storage Explorer tool - command-line tool който е част от инсталацията на Windows Phone SDK .Важен метод в ViewModel е метода LoadData. В него се инциализира БД и се генерират LINQ-to-SQL заявки за да се заредят данни през DataContext в паметта. Бихме могли да заредим с данни и трите таблици, но с цел повишаване на бързодействието в startup – отлагаме това до момента когато съответната страница от приложението е отворена. Единствените данни, които трябва да заредим в startup са тези за таблицата SeasonalHighlight, тъй като те се изписват още в main page. Ето защо, имаме 2 заявки за селектиране на редове от таблици Recipes и Cocktails , които съответстват на сезона и ги комбинират

17. Архитектура ViewModel. Наблюдаеми колекции. Използване и предимства. Класова поддръжка.

Клас ViewModel - посредник между View (потребителския интерфейс) и Model (данните) като за целта използваме класа DataContext. Във ViewModel има поле DataContext и множество колекции за отделните данни в таблицата. Данните са статични, така че колекция от тип List<T> е достатъчна. За същата цел - се нуждаем и от get property accessors, не от set modifiers.
Важен метод в ViewModel е метода LoadData. В него се инциализира БД и се генерират LINQ-to-SQL заявки за да се заредят данни през DataContext в паметта. Бихме могли да заредим с данни и трите таблици, но с цел повишаване на бързодействието в startup – отлагаме това до момента когато съответната страница от приложението е отворена.

Observable Collections:
Represents a dynamic data collection that provides notifications when items get added, removed, or when the whole list is refreshed.
Properties: Count (Gets the number of elements actually contained in the Collection<T>), Item (Gets or sets the element at the specified index) and Items (Gets a IList<T> wrapper around the Collection<T>).
Events: CollectionChanged (Occurs when an item is added, removed, changed, moved, or the entire list is refreshed) and PropertyChanged (Occurs when a property value changes).

18. Sink обекти и работа с тях по обработка на видео или аудио вход.

Обекти Source и Sink:
Алтернатива в ползването на VideoBrush е свързване на обекта CaptureSource към AudioSink, VideoSink или FileSink обекти. Терминът “sink” е в смисъл на получател (receptacle) . Класът FileSink се ползва когато искаме наше видео или звук да се съхранят в isolated storage . Ако ви е нужен достъп до текущо записваните битови поредици на видео или звук, в реално време – ползвате класовете VideoSink и AudioSink.
Тези класове са абстрактни (abstract). Наследявате единия или двата и припокривате методите им: OnCaptureStarted(), OnCaptureStopped(), OnFormatChange() OnSample(). В наследниците на VideoSink или AudioSink първо викате OnFormatChange(), после - OnSample() OnFormatChange() връща информация как да се интерпретират получаваните данни. И за VideoSink, и за AudioSink, повикването на OnSample() изработва информация за timing на снимките и връща масив от битове. При VideoSink, например, тези битове са редове и колони от pixels за всеки frame на видеото.

19. Работа с камера. Разглеждане на снимки. Създаване на видео-клип.

Интерфейсът за webcam се състои от дузина класове на System.Windows.Media namespace. Обикновено се започва от статичния CaptureDeviceConfiguration клас. Викат се GetDefaultVideoCaptureDevice() и GetDefaultAudioCaptureDevice() . CaptureDeviceConfiguration() връща инстанция на VideoCaptureDevice и AudioCaptureDevice. За видео, се изработва: device's name + форматни спецификации (размери в pixels за всеки фрейм на видеото, цветова гама, фреймове/секунда). За аудио, се изработват и броя канали, bits per sample както и wave format, който за момента е Pulse Code Modulation (PCM). Ако CaptureDeviceConfiguration.AllowedDeviceAccess property е true, потребителят може да ползва устройството.
В приложението можем да създадем и CaptureSource object, който комбинира звук и видео в общ поток. Тогава можем да стартираме методите Start() и Stop(). Можете да повикате и

CaptureImageAsync() на CaptureSource за да обработвате отделен video кадър като WriteableBitmap обект

Опция VideoBrush:
Основен атрибут на CaptureSource е VideoBrush. Служи за попълване на контроли с цвят за backgrounds или foregrounds. Следва извадка от приложение - StraightVideo което ползва VideoCaptureDevice, CaptureSource и VideoBrush за да изобрази live video, въведено от камерата. Показана е част от MainPage.xaml . отбележете: - Задаване на landscape mode (препоръчителна за видео),
-Дефиниране на VideoBrush за Background property
-Дефиниране на Grid и
-Button за достъп до камерата.

Writeable Bitmap:
С методи на клас WriteableBitmap може да се пусне или обработи по кадри записано видео. Размерът на кадъра е най-често 640x480 pixels за телефон, но може да бъде променян. обработване на кадър с методи на клас WriteableBitmap: Ако искате да обработвате кадри , постъпващи в real-time от видео-източник, можете да ползвате функционалност от CaptureImageAsync() на класа CaptureSource. Наследникът на VideoSink е получил видеото. Искате да отделите от кадър част – равностранен триъгълник:
Отделената част може да се размножи и завърти подходящо с функционалност на WriteableBitmap и като се отмести и насложи – да формира някаква нова форма.

22. Сензор GPS. Ориентация в 3D пространството.

Работа с GPS Location Information:
Най-общо казано, следва да сте създали GPS listener в приложението си, както и достъп до доставените от listener данни, за да можете да ги изобразите. В Windows Phone SDK се използва клас Compass. В кода го конфигурирате (например интервали на обновяване). Event handler следва да е описан за обработката: _compass.CurrentValueChanged += new EventHandler<SensorReadingEventArgs<CompassReading>>(…);

Сензор - Ориентация
Най-простия от разглежданите сензори – SimpleOrientationSensor. Той позволява грубо ориентиране в 3D пространството. За инстанциирането му – виквате статичен метод: SimpleOrientationSensor simpleOrientationSensor = SimpleOrientationSensor.GetDefault();
По всяко време след това, можете да получите стойност – указваща текущата ориентация (чрез обекта от тип SimpleOrientationSensor): SimpleOrientation simpleOrientation = simpleOrientationSensor.GetCurrentOrientation(); SimpleOrientation представлява изброим тип с 6 члена: NotRotated, Rotated90DegreesCounterclockwise, Rotated180DegreesCounterclockwise, Rotated270DegreesCounterclockwise, Faceup и Facedown.
При регистриран интерес, получавате нотификация за всяка промяна в ориентацията: simpleOrientationSensor.OrientationChanged += OnSimpleOrientationChanged; Този event се запалва само при промяна на ориентацията, така че няма да има ефект при стационарен компютър –той е неподвижен. При необходимост за прочитане текуща стойност – викате метода GetCurrentOrientation(). Event handler се изпълнява в отделна нишка, така че за да може тя да взаимодейства с нишката поддържаща потребителския интерфейс, следва да ползвате обект CoreDispatcher.

Посоката север (Compass):
Въпреки че Accelerometer дава информация кое е долу, той не ни информира напълно за ориентацията на устройството в 3D пространството. Да предположим - сме стартирали

програма AccelerometerAndSimpleOrientation. Държим устройството в някакво странно положение. Accelerometer показва кое е долу. Нека се завъртим в кръг. Таблетът се е завъртял на 180 градуса в пространството , но Accelerometer отново показва същата стойност, тъй като посоката „долу“ е останала същата спрямо устройството.
Като сме завъртели таблета в кръг, какво се е променило? - Отговорът е : посоката „север“спрямо таблета. Ето защо, сензорът Compass е важен: дава допълнителна информация за ориентацията. Комбинирайки Compass и Accelerometer, можете да изработите цялостна информация за ориентацията на таблета в 3D пространството.
Класът Compass е структуриран като Accelerometer. Така и класът CompassReading има property HeadingMagneticNorth – даващо ъгъл в градуси между компютъра и посока север. Ъгълът ще е около 0, ако държите екрана на таблета успореден на земята и върхът му, сочи север. Ако завъртите екрана в посока изток, ъгълът нараства.

OrientationSensor = Accelerometer + Compass:
Ротацията в три-размерното пространство може да се отчете по различни начини,всички от които са взаимно-свързани. Класът OrientationSensor е подобен на Inclinometer, в смисъл, че комбинира информация отново от accelerometer и от compass с цел да създаде представа за ориентацията в 3D пространството. OrientationSensor прави това с помощта на инстанциите на 2 други класа: SensorQuaternion и SensorRotationMatrix.
Quaternions са интересна тема от математиката. Точно както имагинерните числа могат да изобразят ротацията в дву–размерно пространство , така и quaternion служат за математическо описание на ротациите в 3-дименсионното пространство. rotation matrix е регулярна трансформационна матрица с липсващи: последна колона и последен ред. (Регулярната три-дименсионна трансформационна матрица има 4 реда и 4 колони.) Класът SensorRotationMatrix дефинира 3 реда и 3 колонна матрица. Подобна матрица не е в състояние да опише транслация или перспектива, както и мащабиране или огъване на обект. Но лесно се ползва за описване на ротации в 3D пространството.
Когато работим с ротационната матрица на класа OrientationSensor, можем да преобразуваме в две различни, 3D координатни системи:
-Едната, свързана с устройството
-Другата – със Земята. В 3D-координатната система, свързана с компютъра:
-Оста Y сочи към върха на екрана,
-Оста X сочи надясно и
-Оста Z сочи навън (нагоре) от екрана. в координатната система, свързана със земята: - оста Y сочи север; - ос X сочи изток и - ос Z сочи от Земята - нагоре. Двете координатни системи са успоредни, когато компютърът е в легнало положение, с екран нагоре и върхът му сочи север. Ротационната матрица описва взаимното изместване на Земята и компютъра, което е обратно на ротацията, описвана с Ойлерови ъгли.

23. Сензор за измерване на ускорение и посоки.

Ускорение (acceleration), сила, гравитация и вектори:
Сензорът SimpleOrientationSensor има достъп и до hardware наричан accelerometer. Това е устройство, измерващо ускорение и на пръв поглед едва ли е много необходимо. Ако , обаче, си припомним втория закон на Исак Нютон - за движението: F= ma ще установим, че в повечето време accelerometer измерва гравитацията и дава отговор на въпроса “къде е долу?” Достъпът до accelerometer hardware е през клас Accelerometer. За да се инстанциира той – ползваме static method както и при SimpleOrientationSensor: Accelerometer accelerometer = Accelerometer.GetDefault(); По всяко време можете да вземете текуща стойност от Accelerometer: AccelerometerReading accelerometerReading = accelerometer.GetCurrentReading();

Класът AccelerometerReading съдържа 4 properties: AccelerationX от тип double AccelerationY от тип double AccelerationZ от тип double Timestamp от тип DateTimeOffset Трите double стойности, заедно формират 3D vector, който сочи от устройството към земята.
Можете да привържете event handler към обекта Accelerometer: accelerometer.ReadingChanged += OnAccelerometerReadingChanged; Подобният event в SimpleOrientationSensor беше OrientationChanged. Както и там, ReadingChanged handler се изпълнява в отделна нишка.
Ако компютърът е неподвижен, AccelerationX, AccelerationY, и AccelerationZ properties на класа AccelerometerReading дефинират вектор, сочещ центъра на земята. (обозначенията на вектор са обикновено с boldface coordinates - (x, y, z) за да могат да се отличават от запис на точка. Например, ако таблет се държи прав - acceleration vector сочи в „–Y“ посока. Големината на вектора е 1, така че може да се запише (0, –1, 0). Ако устройството лежи на плоска повърхност, с екрана нагоре, векторът е (0, 0, –1). Величината 1 е в стойности „ g“ - (земно ускорение).
Inclinometer е единият от двата класа, комбиниращи и обработващи данни от accelerometer и compass (другият – следва). Класът изработва информация за : yaw, pitch и roll ъгли, които са термини, взети от авиацията. Ротационната матрица, описва ротация на Земята спрямо компютъра. Това означава, че ако искате да опишете ротацията на компютъра, спрямо Земята, трябва да инвертирате матрицата.

24. Работа с карти (Bing Maps) и свързване с позиция и промяна в ориентацията.

Класът Geolocator не представя сензор. Той си е и в друго пространство: Windows.Devices.Geolocation. Все пак, той следва да се стартира, когато искате информация за промяна на географско положение и определяне на това положение текущо. В явен вид трябва да укажете в Capabilities section на Package.appx файла manifest, че вашето приложение изисква Location information. Тогава Windows 8 иска потвърждение от потребителя при първоначално стартиране на програмата.
Най-често Geolocator location се ползва при свързване с географски карти. Контролът Bing Maps не е вграден в Windows 8, но може да се свали. Ще ви е нужен credentials key, който може да се получи от www.bingmapsportal.com. Нека покажем нещо по-сложно: карта, която се върти, съобразно ориентацията на таблета. При това, можем да свържем (както и се прави в професионалните програми) посоката север на картата с действителния - север (или това, което таблета смята за север). За да се постигне това, по-добре е вместо Bing Maps control, да се ползва Bing Maps SOAP service, чрез която услуга могат да се свалят отделни порции (tiles) и те да се стиковат така че да се получи добро изображение от частта на картата. За услугата също е нужен credentials key.
За целта се създава proxy class , който Visual Studio генерира. За добавяне на този proxy към програмата, кликвате десен бутон над project name в среда на Solution Explorer на Visual Studio и избирате - Add Service Reference. Въвеждате адреса - URL на тази Imagery Service (можете да го откриете на http://msdn.microsoft.com/en-us/library/cc966738.aspx заедно с още 3 URL на други Web services, свързани с Bing Maps. Услугата поддържа 2 типа заявки: GetMapUriAsync GetImageryMetadataAsync Първата позволява извличане на статична карта за искана локация. Втората – позволява получаване на информация, нужна за сваляне на порции от картата (map tiles), които впоследствие могат да се асемблират в цялостна карта. Тази услуга е по-подходяща за замисленото ни приложение.
Web услугата се достъпва в Loaded handler. 2 повиквания към нейни функции следва да се направят: една за извличането на maps metadata, нужни за road view и другата за aerial view. Най-съществената част от споменатите метаданни е URI template , нужен за свалянето на отделните порции от картата. Нашият клас -ViewParams ще има и 2 полета – за minimum и maximum zoom levels. Знаем, че zoom level може да заема стойности от 1 до 21:
Свалянето на двете изображения се изпълнява в асинхронни повиквания, така че получаването на метаданните за това да е независимо едно от друго. Те могат да се изпълняват в едно и

също време. Това се постига най-удачно с повикване на метода Task.WaitAll(), който изчаква докато всички Task items се довършат.
Когато двете повиквания на Web service се изпълнят успешно, могат да се стартират нашите :Geolocator и Inclinometer. Inclinometer се използва единствено за изработване на стойност за yaw, нужна при ротиране на картата , както и за ротиране на стрелката , сочеща север.
След завършването на Loaded handler, приложението ни има 2 URI templates, които то ще ползва при свалянето на отделните порции от картата. Отделните порции (tiles) са базовия елемент на Bing Maps системата и представляват bitmaps изображения, които са винаги квадрат от 256 pixels. Всеки tile е асоцииран със своя longitude, latitude и степен на zoom. При Level 1, цялата Земя— или по-точно частта от Земята с географска ширина между + и - 85.05 градуса, се е вместила в 4 tiles. Да поясним тези порции (tiles) малко по-подробно. Всяка е квадрат - 256 pixels, така че на екватора всеки пиксел отговаря на 49 мили. При Level 2, 16 tiles покриват цялата Земя.

25. Определяне на идентификатор за порция от картата и извличането на квадрант.

Всяка порция (tile) при Level 1 покрива еднаква област от картата, както 4 tiles в Level 2. последователността продължава в същия ред: Level 3 има 64 tiles, Level 4 - 256 tiles, и така до Level 21, който покрива Земята с повече от 4 trillion tiles—2 милиона хоризонтална посока и 2 милиона във вертикална . Това осигурява резолюция при екватора от 3 инча земна повърхност за 1 pixel в изображението . 3 размерноти са необходими в процеса: географска дължина, ширина и zoom level. С оглед постигане на максимум бързодействие, порциите (tiles) покриващи съседни области се съхраняват за оптимизиран достъп на сървъра. Схемата за номериране се нарича - quadkey. Всеки tile притежава уникален quadkey номер. URI templates получени при Bing Maps Web service съдържат ключ “{quadkey}” , с референции към нужните порции (tile). броят цифри, обозначаващи quadkey идентификатора е равен на zoom level. Така,че порция от ниво Level 21 се дефинира с 21-цифров quadkey.
Цифрите в „ quadkey“ идентификатора са винаги 0, 1, 2 или 3. в двоичен формат - 00, 01, 10 и 11. първият бит е вертикалната координата, вторият - хоризонталната. Така че, битовете определят относителната геогр. Ширина и дължина: видяхме, че всеки елемент (tile) от Level 1 съответства на 4 елемента (tiles) в Level 2. т.е има определено съотношение между елементите на картата : родители - деца. Идентификаторът „quadkey“ на дъщерен елемент винаги започва със същите като на родителя цифри, като добавя нова, според разположението спрямо родителя.
Следователно, можете да изработите родителския „quadkey“ идентификатор, от дъщерния – просто със замазване на последна цифра. За да ползвате Bing Maps Web услуга, е необходимо да изработите идентификатора „quadkey“ от географскта ширина и дължина на мястото. Следва да сте подготвили свой метод за това: да преобразува дължина и ширина, получени от Geolocator в относителните стойности (double) в 0 и 1, които ще подадете във вид на цяло число. Идентификаотрът „quadkey“ служи за извличане на tile с исканата longitude и latitude, но точното местоположение е някъде в областта. Местоположението (пикселът) вътре в областта (tile) може да се определи от младшите 8 цифри на целите числа - за longitude и за latitude, след цифрите, необходими при формирането на quadkey идентификатора. Вече сме близо да края. Тъй като цялата страница трябва да изобразява няколко 256-pixel-квадратни области (tiles), както и поради факта че всички те подлежат на ротация и поради нуждата текущата локация на клинта да епозиционирана в центъра на екрана, някъде в централния tile, обработчикът на SizeChanged (handler) трябва да определи колко елемента (tiles) са нужни и оттам – колко елемента на приложението от тип Image трябва да се създадат. Тогава в поле, именовано например sqrtNumTiles (което означава “the square root of the number of tiles.”) се съхранява броя елементи, които следва да се свалят. Например, за екран с резолюция - 1366 на 768 – областите, които се изобразяват са 9. но общия брой на нужните, поради гореизброени

причини географски елемента за сваляне (и съответно Image елемента на приложението) се определя като стойността се повдигне на квадрат – т.е. 81.

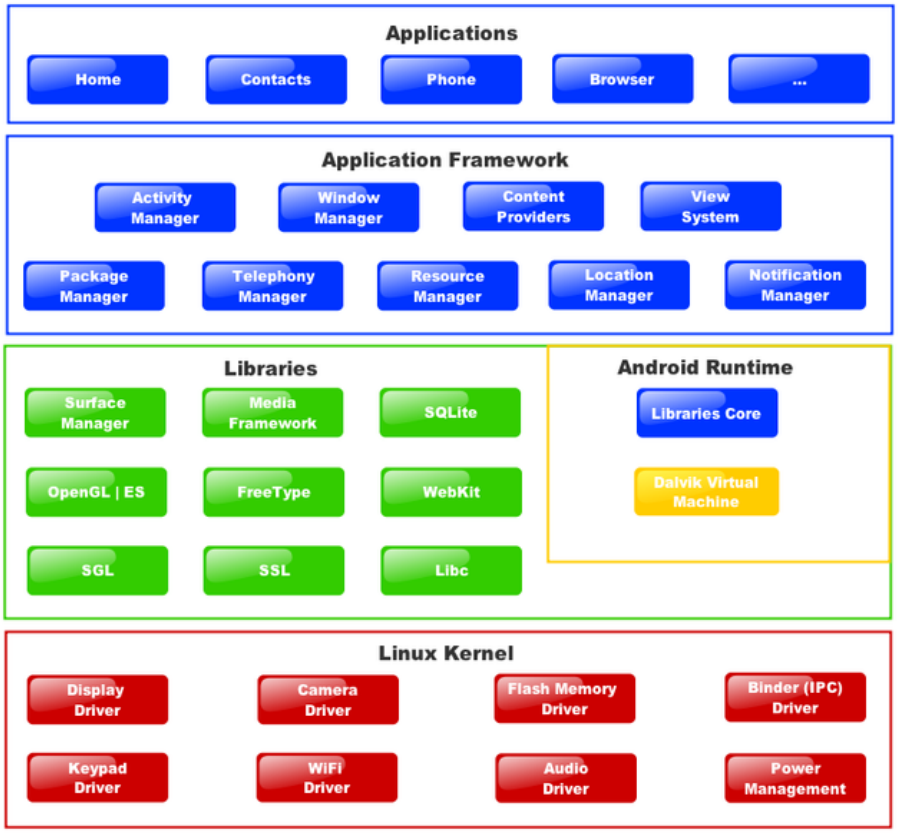
26. Операционна система Android. Архитектура на операционната среда Android. Софтуерни слоеве.

Общи сведения:
През 2005 г. Google Inc. закупува операционната система Android от Android Inc. и започва нейното развитие. Пускането и на пазара на 5 ноември 2007 г. е съпроводено с основаването на Open Handset Alliance - консорциум от 84 хардуерни, софтуерни и телекомуникационни компании, решени да развиват отворените стандарти при мобилните устройства. Google пуска по-голяма част от кода под свободен лиценз.
Операционната система Android използва в основата си модифицирана версия на ядрото на Linux.
За развитието на Android се грижат голям брой софтуерни разработчици, които създават така наречените "apps" - малки приложения, които разширяват функционалността на системата. Приложенията могат да бъдат сваляни от различни сайтове в Интернет или от големи он-лайн магазини като Android Market (ново име Google Play) - магазинът на Google. По данни към януари 2011 г. за Android има над 200 000 приложения. Приложенията се пишат предимно на Java, Python или Ruby.

Архитектура:
Основата на Android е ядрото на Linux (версия 2.6). То е отговорно за управлението на паметта и процесите, както и за мрежовите връзки. Тук са разположени и драйвърите.
Директно над ядрото се намира т. нар. "Runtime Environment". Тя съдържа най-важните библиотеки по време на изпълнение и най-важната функционалност на езика Java. Тук се съдържа и виртуалната машина Dalvik Virtual Machine (DVM). Тя се различава от класическите виртуални машини на Java (Java Virtual Machine, JVM) по това, че е оптимизирана за мобилни уреди с малко памет. Оптимизацията позволява и едновременното изпълнение на няколко виртуални машини на същия уред.
Android съдържа няколко библиотеки на C/C++:

- Surface Manager (управлява достъпа до дисплея)
- OpenGL ES (приложно-програмен интерфейс за 3D компютърна графика), SGL (приложно-програмен интерфейс за 2D компютърна графика)
- Media Framework (управление на мултимедийно съдържание, на основата на OpenCORE, поддържа формати като MPEG4, H.264, MP3, AAC, AMR, JPG PNG и др.)
- FreeType (библиотека за рендъринг на пикселни и векторни шрифтове)
- SSL (криптиране)
- SQLite (бази данни)
- WebKit (рендъринг на HTML)
- Libc (версия на стандартната C-библиотека за Android)

Приложният фреймуърк (Application Framework) ползва библиотеките на C/C++ и предлага стандартизиран приложно-програмен интерфейс за програмистите на приложения (Apps).
Android се доставя с няколко приложения, сред които са комуникационните приложения за телефониране, електронна поща, SMS и браузър, както и Google Maps, календар и приложение за управление на контактите.



27. Компоненти на приложение в Android. Комуникации между компонентите.

Ето и някои от по важните компоненти за едно Андроид приложение.

- Activity – Представя тип слой от приложението например екран, който потребителя ще вижда. Едно приложение може да има няколко слоя активности и може да превключва между тях по време на работа на проложението. Потребителския интерфейс на тези слоеве е изграден от widget класове които са свързани от "android.view.View" Слоевете view се управляват от ViewGroups.
- Services – Услугите ви осигуряват фонови задачи без наличието на интерфейс. Те могат да известяват потребителя посредством notification framework в Андроид.
- Content Provider – Доставя данни от приложението . Благодарение на този компонент можете да споделяте данни с други приложения. Андроид съдържа SQLite база данни, която може да служи като доставчик на тези данни.
- Intents – Това са асинхронни съобщения, които позволяват приложението да иска функционалност от други услуги или дейности. Приложението може директно да извика услуга или дейност (explicit intent) или да извести “системата за регистрация на услуги и приложения на Андроид” за тези си намерения (implicit intents). Например приложението може да извика чрез Intent приложението за контакти. Приложението се регистрира посредством Intent филтър (IntentFilter). Компонента Intent е много мощен инструмент благодарение на който можете да създавате свободно свързани приложения.
- Broadcast Receiver - Получава системни съобщения, както и асинхронни съобщения implicit intents (за справка виж малко по на-горе). Може да се използва за реакция при променящи се условия в системата. Приложението може да се регистрира като приемник (Broadcast Receiver) за определени събития, и може да се стартира при наличието на такова (събитие).

28. Манифест на приложение. Android SDK и процес на разработка. Публикуване на приложение.

The Manifest File:
- You must declare all its components in a manifest file called AndroidManifest.xml which ressides at the root of the application project directory.
- This file works as an interface between Android OS and your application.
- If you do not declare your component in this file, then it will not be considered by the OS.
Following is the list of tags which you will use in your manifest file to specify different Android application components:
<activity> - elements for activities;
<service> - elements for services ;
<receiver> - elements for broadcast receivers;
<provider> - elements for content providers.

Процес на разработка:
Разработка на приложения за Android устройства се улеснява от група инструменти, които се предоставят с SDK.Можете да получите достъп до тези инструменти чрез плъгин на Eclipse наречен ADT (Android Development Tools) или от командния ред. Разработка с Eclipsee предпочитаният метод, защото може пряко да се позовете на инструменти,от които се нуждаете, по време на разработката на приложения.
Основните стъпки за разработване на приложения с или без Eclipse са едни и същи:
1.Настройте Android Virtual Devices или хардуерни устройства. Трябва да създадете Android Virtual Devices(AVD) или да се свържете хардуерни устройства, на които ще инсталирате приложения. Един Android Virtua lDevice(AVD) е емулатор конфигурация, която ви позволява да симулирате действително устройство чрез определяне на хардуерни и софтуерни опции да бъдат емулирани от Emulator Android. Най- лесният начин за създаване на AVD е да използвате графичния AVD мениджър, който се стартира от Eclipse.
2. Създаване на проекта Android. Един проект Android съдържа всичкия сорс код и ресурсни файлове за вашето приложение. Той е построен в apk пакет, които можете да инсталирате на устройства с Android.
3. Изгради и стартирай вашето приложение.
4. Debug на вашето приложение с SDK debuging инструменти. Инструментите са осигурени с Android SDK. Eclipse вече идва в комплект със съвместим дебъгер.
5. Тествайте вашето приложение с тестване и **Instrumentation framework**.
AnAVD е конфигурацията на устройството за Android емулатор, който ви позволява да моделирате различни конфигурации на Android- устройства. Когато стартирате AVD мениджъра в Eclipse или стартирате Android инструмент на командния ред.

Android SDK:
Първо- създавате своя SDK.

Platform	***Package***	***Size***
Windows	**android-sdk_r12-windows.zip**	**36486190 bytes**
	installer_r12-windows.exe (Recommended)	**36531492 bytes**
Mac OS X (intel)	**android-sdk_r12-mac_x86.zip**	**30231118 bytes**
Linux (i386)	**android-sdk_r12-linux_x86.tgz**	**30034243 bytes**

Eclipse IDE Android Development Tools(ADT) е плъгин за Eclipse IDE, която е предназначена да ви даде мощна интегрирана среда, в която да се изградите Android приложения.

29. Android приложение. Файлова структура. Създаване и изпълнение. R клас. Примери.

AndroidAPI- то включва повече от 70 пакета, споведе от 400 класа
Първо приложение: Hello,World
Като разработчик, вие знаете, че първото впечатление за development frameworka-а е колко лесно еда се напише"Hello, World". Е,на Android, това е доста лесно. Това е особено лесно, ако използвате Eclipse IDE, защото този плъгин управля вавашия проект, значително ще ускори вашият цикъл на разработка
1.Вие ще стартирате вашето приложение в Emulator Android. Преди да можете да стартирате емулатора, трябва да създадете Android Virtual Device (AVD). AVD определя системен образ и настройките на устройството, използвани от емулатора
2. Създаване на нов проект Android: След като сте създали AVD можете да преминете към следващата стъпка и да започне нов проект Android в Eclipse.
Вашият Android проект вече е готов. Той трябва да се вижда в пакет Explorer в ляво. Отворете файла HelloAndroid.java, разположени вътре HelloAndroid→SRCcom.example.helloandroid Тя трябва да изглежда така: package com.example.helloandroid; import android.app.Activity; import android.os.Bundle; public class HelloAndroid extends Activity { /** Called when the activity is first created. */ @Override public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.main); } } Забележете, че този клас се базира на Activity класа. Activity е един обект приложение, което се използва за извършване на действия.Приложението може да има много отделни дейности, но потребителят взаимодейства с тях един по един. TheonCreate() метод се извиква от системата Android, когато вашето Activity започва- това е мястото, където трябва да се изпълни цялата инициализация и UI настройка.

R клас:
Файлът R.java е индекс на всички ресурси, определени във файла.
Можете да използвате този клас в изходен код като един по-кратък начин да реферира на ресурсите, които сте включили във вашия проект.
It is an automatically generated file and you should not modify the content of the R.java file.

30. Организация на ресурси. Достъп до ресурси.

Организация:
- You should place each type of resource in a specific subdirectory of your project's res/ directory.
- anim/ XML files that define property animations. They are saved in res/anim/ folder and accessed from the R.anim class.
- color/ XML files that define a state list of colors. They are saved in res/color/ and accessed from the R.color class.
- drawable/ Image files like .png, .jpg, .gif or XML files that are compiled into bitmaps, state lists, shapes, animation drawables. They are saved in res/drawable/ and accessed from the R.drawable class.
- layout/ XML files that define a user interface layout. They are saved in res/layout/ and accessed from the R.layout class.
- menu/ XML files that define application menus, such as an Options Menu, Context Menu, or Sub Menu. They are saved in res/menu/ and accessed from the R.menu class.
- raw/ Arbitrary files to save in their raw form. You need to call*Resources.openRawResource()* with the resource ID, which is *R.raw.filename* to open such raw files.
- xml/ Arbitrary XML files that can be read at runtime by calling *Resources.getXML()*. You can save various configuration files here which will be used at run time.

- values/ XML files that contain simple values, such as strings, integers, and colors. For example, here are some filename conventions for resources you can create in this directory:
    - ◦*arrays.xml* for resource arrays, and accessed from the R.array class.
    - ◦*integers.xml* for resource integers, and accessed from the R.integerclass.
    - ◦*bools.xml* for resource boolean, and accessed from the R.bool class.
    - ◦*colors.xml* for color values, and accessed from the R.color class.
    - ◦dimens.xml for dimension values, and accessed from the R.dimen class.
    - ◦*strings.xml* for string values, and accessed from the R.string class.
    - ◦*styles.xml* for styles, and accessed from the R.style class.

Достъп:
- When your Android application is compiled, a **R** class gets generated, which contains resource IDs for all the resources available in your **res/** directory.
- You can use R class to access that resource using sub-directory and resource name or directly resource ID.

31. Компонент Activity в Android приложение. Създаване на Activity. Стек на Activity.

Activities — Your application's presentation layer. The UI of your application is built around one or more extensions of the Activity class. Activities use Fragments and Views to layout and display information, and to respond to user actions. Compared to desktop development, Activities are equivalent to Forms.
- Each Activity represents a screen that an application can present to its users.
- The more complicated your application, the more screens you are likely to need.
- To move between screens you start a new Activity (or return from one).
- Most Activities are designed to occupy the entire display, but you can also create semitransparent or floating Activities.

Създаване: (skeleton code)

```
package com.paad.activities;

import android.app.Activity;
import android.os.Bundle;

public class MyActivity extends Activity {

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
  }
}
```

Activity Stacks:
- The state of each Activity is determined by its position on the Activity stack, a last-in–first-out collection of all the currently running Activities.
- When a new Activity starts, it becomes active and is moved to the top of the stack.
- If the user navigates back using the Back button, or the foreground Activity is otherwise closed, the next Activity down on the stack moves up and becomes active.

Activity States:
- Active — When an Activity is at the top of the stack it is the visible, focused, foreground Activity that is receiving user input. When another Activity becomes active, this one will be paused.
- Paused — In some cases your Activity will be visible but will not have focus; at this point it's paused. This state is reached if a transparent or non-full-screen Activity is active in front of it. When paused, an Activity is treated as if it were active; however, it doesn't receive user input events.
- Stopped — When an Activity isn't visible, it "stops." The Activity will remain in memory, retaining all state information; however, it is now a candidate for termination when the system requires memory elsewhere. When an Activity is in a stopped state, it's important to save data and the current UI state, and to stop any non-critical operations.
- Inactive — After an Activity has been killed, and before it's been launched, it's inactive. Inactive Activities have been removed from the Activity stack and need to be restarted before they can be displayed and used.

32. Изграждане на графически потребителски интерфейс. Основни елементи. Работа с Layout. Фрагменти. Контроли.

Fundamental Android UI design:
- Views — Views are the base class for all visual interface elements (commonly known as *controls* or *widgets*). All UI controls, including the layout classes, are derived from View.
- View Groups — View Groups are extensions of the View class that can contain multiple child Views. Extend the ViewGroup class to create compound controls made up of interconnected child Views. The ViewGroup class is also extended to provide the Layout Managers that help you lay out controls within your Activities.
- Fragments — Fragments, introduced in Android 3.0 (API level 11), are used to encapsulate portions of your UI. This encapsulation makes Fragments particularly useful when optimizing your UI layouts for different screen sizes and creating reusable UI elements. Each Fragment includes its own UI layout and receives the related input events but is tightly bound to the Activity into which each must be embedded.
- Activities — Activities represent the window, or screen, being displayed. Activities are the Android equivalent of Forms in traditional Windows desktop development. To display a UI, you assign a View (usually a layout or Fragment) to an Activity.

Layout:
- Defines how elements are positioned relative to each other (next to each other, under each other, in a table, grid, etc.)
- Can have a different layouts for each ViewGroup.

Layout Managers:
- Behave like containers for other views.
- Implements strategies to manage size, position of its children.
- Layout managers used in android:
    - ◦Linear Layout: Aligns all the children in one direction (Horizontally or Vertically); Children are stacked one after another We may nest multiple linear layouts or linear layout within some other layout.
    - ◦Relative Layout: Display child views in relative positions; We may specify position in relation with parent or siblings of a view; Eliminates the need of nested views; Many nested linear layouts can be converted into one Relative Layout.

◦Table Layout: Keep all the child views in a table; In Table Layout, TableRow represent one row; All children in a TableRow are columns; Useful to display data in rows and columns; Not useful for designing complete user interfaces.

◦Grid Layout: Places all of its child views in a rectangular grid; By default you we may define rowCount & colCount and all child views in a grid layout behaves like a matrix; We can manually define which row/col a certain object belongs to using layout_row & layout_column property; Useful for displaying image galleries, grid data and similar things.

Layout Params**:**

- Define attributes available to all the child controls within Layout Manager.
- All type of layout managers have various layout params that define position, weight, gravity, etc. for a child within that certain layout manager, for instance:
    - ◦In LinearLayout.LayoutParams we have:
        - + Gravity (android:layout_gravity)
        - + Weight (android:layout_weight)
    - ◦In RelativeLayout.LayoutParams we have:
        - + Layout Above (android:layout_above)
        - + Layout Top (android:layout_alignTop)
        - + and many more...

Fragments**:**

- Fragments enable you to divide your Activities into fully encapsulated reusable components, each with its own lifecycle and UI.
- Each Fragment is an independent module that is tightly bound to the Activity into which it is placed.
- Fragments can be reused within multiple activities.
- Fragments provide a way to present a consistent UI optimized for a wide variety of Android device types, screen sizes, and device densities.
- Each Activity includes a Fragment Manager to manage the Fragments it contains.
- Fragment Transactions can be used to add, remove, and replace Fragments within an Activity at run time.
- Using Fragment Transactions, you can make your layouts dynamic — that is, they will adapt and change based on user interactions and application state.
- Each Fragment Transaction can include any combination of supported actions, including adding, removing, or replacing Fragments.

Basic Input Controls**:**

- Input controls are used to take data from user.
- Most commonly used controls in Android Ecosystem are:
    - ◦Text Fields
    - ◦TextView
    - ◦Buttons (Button, ImageButton, RadioButton, ToggleButton)
    - ◦Checkboxes
    - ◦Spinners
    - ◦ImageView

33. Управление на събитията в Android. Пример.

Handling User Interaction Events:
- For your new View to be interactive, it will need to respond to user-initiated events such as key presses, screen touches, and button clicks.
- Android exposes several virtual event handlers that you can use to react to user input:
    - ◦onKeyDown — Called when any device key is pressed; includes the D-pad, keyboard, hang-up, call, back, and camera buttons
    - ◦onKeyUp — Called when a user releases a pressed key
    - ◦onTrackballEvent — Called when the device's trackball is moved
    - ◦onTouchEvent — Called when the touchscreen is pressed or released, or when it detects movement

Event Handling:
- Decide what Widgets who's events to process
- Define an event listener and register it with the View.
    - ◦View.OnClickListener (for handling "clicks" on a View), View.OnTouchListener (for handling touch screen events in a View), and View.OnKeyListener (for handling device key presses within a View)

Пример:
- Step 1: Add button
- Step 2: Register Event Handler
    - +TWO OPTIONS – separate class to handle event(s), OR have the Activity containing the button do the event handling
- Step 3: Implement Event Handler…for a Button means implementing the View.OnClickListener interface

- Here code to handle is inside Activity itself:

```
public class ExampleActivity extends Activity implements OnClickListener {
protected void onCreate(Bundle savedValues) { ...
Button button = (Button)findViewById(R.id.corky); //STEP 1
button.setOnClickListener(this); //STEP 2 - registration }
// Implement the OnClickListener callback //STEP 3 –event handler
public void onClick(View v) { // do something when the button is clicked } ... }
```

34. Работа с компонента Intents. Неявно и явно стартиране на Activity в Android приложение. Broadcast събития.

Intents:
Intents are used as a message-passing mechanism that works both within your application and between applications.
- Explicitly start a particular Service or Activity using its class name.
- Start an Activity or Service to perform an action with (or on) a particular piece of data.
- Broadcast that an event has occurred.

Using Intents to Launch Activities:
- To create and display an Activity, call startActivity, passing in an Intent, as follows:

```
startActivity(myIntent);
```

- The startActivity method finds and starts the single Activity that best matches your Intent.
- Explicitly Starting New Activities:

```
Intent intent = new Intent(MyActivity.this, MyOtherActivity.class);
startActivity(intent);
```

- each time you call startActivity, a new Activity will be added to the stack;
- pressing back (or calling finish) will remove each of these Activities, in turn.
- An Activity started via startActivity is independent of its parent and will not provide any feedback when it closes.

Implicit Intents:
- Mechanism that lets anonymous application components service action requests.
- You can ask the system to start an Activity to perform an action without knowing which application, or Activity, will be started.
- Dial a number:

```
Intent intent = new Intent (Intent.ACTION_DIAL, Uri.parse("tel:93675359"));
startActivity(intent);
```

- Launch a website:

```
Intent intent = new Intent (Intent.ACTION_VIEW, Uri.parse("http://codeandroid.org"));
startActivity(intent);
```

Using Intents to Broadcast Events:
- You can also use Intents to broadcast messages anonymously between components via the sendBroadcast method.
- As a system-level message-passing mechanism, Intents are capable of sending structured messages across process boundaries.
- As a result, you can implement Broadcast Receivers to listen for, and respond to, these Broadcast Intents within your applications.
- Broadcast Intents are used to notify applications of system or application events, extending the event-driven programming model between applications.

35. Провайдер на съдържание и споделяне на данни. Достъп да данни в Android приложение. Пример.

Content provider:
- Provider offer data encapsulation based on URI's.
- Any URI which starts with content:// points to a resources which can be accessed via a provider.
- A URI for a resource may allow to perform the basic CRUD operations (Create, Read, Update, Delete) on the resource via the content provider.
- A provider allows applications to access data.
- Content providers manage access to a structured set of data.
- They encapsulate the data, and provide mechanisms for defining data security.
- Content providers are the standard interface that connects data in one process with code running in another process.
- When you want to access data in a content provider, you use the ContentResolver object in your application's Context to communicate with the provider as a client.
- The ContentResolver object communicates with the provider object, an instance of a class that implements ContentProvider.
- The provider object receives data requests from clients, performs the requested action, and returns the results.
- You don't need to develop your own provider if you don't intend to share your data with other applications.
- However, you do need your own provider to provide custom search suggestions in your own application.
- You also need your own provider if you want to copy and paste complex data or files from your application to other applications.

- Android itself includes content providers that manage data such as audio, video, images, and personal contact information.
- You can see some of them listed in the reference documentation for the android.provider package.
- With some restrictions, these providers are accessible to any Android application.
- A content provider manages access to a central repository of data.
- A provider is part of an Android application, which often provides its own UI for working with the data.
- Content providers are primarily intended to be used by other applications, which access the provider using a provider client object.
- Together, providers and provider clients offer a consistent, standard interface to data that also handles inter-process communication and secure data access.
- A content provider presents data to external applications as one or more tables that are similar to the tables found in a relational database.
- A row represents an instance of some type of data the provider collects, and each column in the row represents an individual piece of data collected for an instance.

Достъп до данни:
- An application accesses the data from a content provider with a ContentResolver client object.
- This object has methods that call identically-named methods in the provider object, an instance of one of the concrete subclasses of ContentProvider.
- The ContentResolver object in the client application's process and the ContentProvider object in the application that owns the provider automatically handle inter-process communication.
- ContentProvider also acts as an abstraction layer between its repository of data and the external appearance of data as tables.

Пример:
- To get a list of the words and their locales from the User Dictionary Provider, you call ContentResolver.query().
- The query() method calls the ContentProvider.query() method defined by the User Dictionary Provider.
- The following lines of code show a ContentResolver.query() call:

```
// Queries the user dictionary and returns results
mCursor = getContentResolver().query( UserDictionary.Words.CONTENT_URI, // The content URI of the words table
mProjection, // The columns to return for each row
mSelectionClause // Selection criteria
mSelectionArgs, // Selection criteria
mSortOrder); // The sort order for the returned rows
```

36. AdapterView и Adapters. Работа с адаптери. Примери.

AdapterView:
- The AdapterView is a ViewGroup subclass whose child Views are determined by an Adapter that binds AdapterView object to data of some type.
- Typically you are going to use subsclasses of AdapterView class instead of using it directly
- Example subclasses of AdapterView class
    - ◦ListView
    - ◦Spinner
    - ◦Gallery
- Two main responsibilities of AdapterView:
    - ◦Filling the layout with data (with a help from Adapter)

◦Handling user selections - when a user selects an item, perform some action
+ AdapterView.OnItemClickListener
+ AdapterView.OnItemLongClickListener
+ AdapterView.OnItemSelectedListener

Adapter:
You can populate an AdapterView such as ListView or GridView by binding the AdapterView instance to an Adapter, which retrieves data from an external source and creates a View that represents each data entry.
- The Adapter provides access to the data items.
- The Adapter is also responsible for making a View for each item in the data set.
- Types of Adatpers - they implements ListAdatper interface
◦ArrayAdatper
◦CursorAdatper
◦There are a few more

37. SQLite база данни. Класове с работа с SQLite в Android приложение. Основни операции.

SQLite Database:
- Embedded RDBMS
- ACID Compliant
- Size – about 257 Kbytes
- Not a client/server architecture: Accessed via function calls from the application
- Writing (insert, update, delete) locks the database, queries can be done in parallel
- Android provides some useful helper classes: SQLiteDatabase, Cursor, ContentValues….
- Datastore – single, cross platform file (Definitions, Tables, Indexes, Data)
- Quite different than the normal SQL data types

android.database.sqlite:
- Contains the SQLite database management classes that an application would use to manage its own private database.
- Contains classes and interfaces to explore data returned through a content provider.
- Contains the methods for: creating, opening, closing, inserting, updating, deleting and quering an SQLite database.
- These methods are similar to JDBC but more method oriented than what we see with JDBC (remember there is not a RDBMS server running).
- The main thing you are going to use here is the Cursor interface to get the data from the resultset that is returned by a query.
- Classes: SQLiteCloseable, SQLiteCursor, SQLiteDatabase, SQLiteProgram, SQLiteQuery, SQLiteQueryBuilder, SQLiteStatement;

Основни операции:
- openOrCreateDatabase():This method will open an existing database or create one in the application data area;
- long insert(String table, String nullColumnHack, ContentValues values);
-int update(String table, ContentValues values, String whereClause, String[ ] whereArgs);
- int delete(String table, String whereClause, String[] whereArgs);

- Queries: Method of SQLiteDatabase class and performs queries on the DB and returns the results in a Cursor object

Cursor c = mdb.query(p1,p2,p3,p4,p5,p6,p7)

◦p1 : Table name (String)
◦p2 : Columns to return (String array)
◦p3 : WHERE clause (use null for all, ?s for selection args)
◦p4 : selection arg values for ?s of WHERE clause
◦p5 : GROUP BY (null for none) (String)
◦p6 : HAVING (null unless GROUP BY requires one) (String)
◦p7 : ORDER BY (null for default ordering)(String)
◦p8 : LIMIT (null for no limit) (String)

38. Изграждане на мобилни Web приложения с HTML5 Frameworks .

HTML5 allows mobile developers to write code only once creating **cross-platform mobile applications** that can be opened with any mobile browser. An **HTML5 mobile app** is a web page, or series of web pages, designed to work on a tiny screen and on a widest range of devices and operating systems.
According to a recent IDC research over 80% of all mobile apps will be wholly or in part based upon HTML5 by 2015. Moreover a HTML adoption survey conducted between developers shows that 63% of them are currently using HTML5 and 31% plan on using it in the future.
Compared to native apps, HTML5 apps have some advantages and limitations:

Advantages:

- Working on all mobile devices and OS
- Offline support
- Contents contained in the app can be found by search engines
- Updates and bugs fixing are immediately available to all users
- Distribution and support is much easier
- No dependency on app stores
- Faster and cheaper to develop

Limitations:

- Limited access to native device functionality (camera, contacts, calendar, notifications)
- Browser fragmentation
- No secure offline storage
- Difficult to discover as it is not listed in any app store.
- No guarantee of safety and security of the app
- Performance can be slower than native apps

Nowadays a lot of softwares give the possibility to developers to easily build mobile web apps based on HTML5:

- Sencha Touch: a high-performance HTML5 mobile application framework. Built for enabling world-class user experiences, Sencha Touch enables developers to build powerful apps that work on iOS, Android, BlackBerry, Windows Phone, and more.

+ Pros: 50 built-in components, themes for every popular mobile platform, model–view–controller system.
+ Cons: Sometimes slow on Android and not working properly on Blackberry. It's very hard to debug and fix errors.

- jQuery Mobile: Instead of writing unique apps for each mobile device or OS, the jQuery mobile framework allows you to design a single highly-branded web site or application that will work on all popular smartphones, tablets and desktop platforms.

+ Pros: If you know basic HTML it's super easy to use. Drag-and-drop UI builder. Compatible with almost every device.

+ Cons: Sites may looks similar due the limited options of themes customization.

- The-M-Project: a Mobile HTML5 JavaScript Framework that helps you build great mobile apps, easy and fast. The framework follows the popular model-view-controller (MVC) software architecture pattern, a huge selling point for many developers. It support all devices: smartphones, tablets and even desktops.

+ Pros: it comes with a build tool called Espresso!, offline support, excellent demos

+ Cons: It's a new project at its alpha stage so some things are not yet implemented.

- DaVinci Studio: this app framework enables you to drop and drag different elements with a WYSIWYG (What You See Is What You Get) authoring environment.

+ Pros: Compatible with open source frameworks, emulator to simulate N-screen environments, extensive functions & widgets.

+ Cons: It's very new on the market so some bugs has still to be fixed.

- Wink: (Webapp INnovation Kit) a toolkit which will help you build great mobile web apps for iOS (iPod, iPhone, iPad), Android, BlackBerry, Bada and now Windows Phone 7. Moreover it is adapted for Firefox mobile and Opera mobile.

+ Pros: it keeps the file size minimum, big library of widgets as 3D wall, sliding panels, flip page, coverflow, progress bar and much more.

+ Cons: the documentation looks a bit poor and cold.

39. Крос-платформени приложения. Платформата „Mono". Избор на архитектура на приложението.

Платформата Mono:

•Mono is an open source implementation of Microsoft's .NET Framework based on the ECMA standards for C# and the Common Language Runtime;

•Runs on multiple platforms - Linux, OS X, BSD, iPhone, Android, PlayStation 3, Wii, Xbox 360, and Microsoft Windows, including x86, x86-64, ARM, s390, PowerPC, SPARC, IA64, MIPS and much more;

•Multiple languages - Develop in C# 5.0 (including async, LINQ and dynamic), VB 8, Java, Python, Ruby, Eiffel, F#, Oxygene, and more;

•Xamarin is the company sponsoring mono development and support. Commercial products based on mono are MonoTouch and MonoAndroid for developing cross platform mobile applications;

•Mono supports everything till .NET 4.0 except WPF, WWF and limited WCF;

•To verify if your .NET application is compatible with mono, run Mono Migration Analyzer;

Choosing the right architecture:

-Always in the network;

-Storage and processor limitations . Servers, XML mediator, SOAP transfer

-Securing data on the device: if available – Lightweight Directory Access Protocol (LDAP) Username+password, data encryption; data destruction

-Building scalable applications

-Writing extendible modules: for devices (barcode, camera, geo-location etc.)

-Choosing the right software architecture:

-Native application

-Web application (HTML 5; CSS)

-Application for different platforms: for iOS; Android; Windows Phone

40. Среда за крос-платформени приложения с Mono за Mac, Windows Phone, iOS, Android. Развойна среда Eclipse.

OS/ product	Mac OSX	Windows
iOS/ MonoTouch	X	
Android/Mono for Android	X	X
Webkit / ASP.NET		X
Windows Phone		X

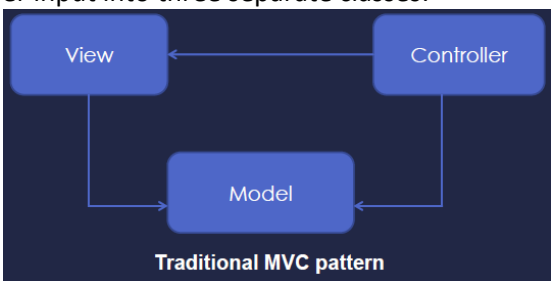
Setting up the development environment:
A. Installing development tools:
1.Installing MS Visual Studio (2010 up) for: Windows Phone 7; Mobile web (WebKit) ; Console samples ; apps for Android also , but Mono for Android is better.
2. Download:
- Latest Service Pack for 2010 +
- Install IIS to be able to develop
- WebKit samples

B. Installing device frameworks (SDK + emulators and simulators):
1.Installing the Windows Phone SDK: by downloads, free. When completed, start Visual Studio→New → Project → Visual C# and you will find Silverlight for Windows Phone
-You can start also Windows Phone Emulator for testing websites without starting VStudio 2010.
2. Preparing for iOS development (iPod/iPhone/iPad) - you need iOS SDK. MonoTouch uses them to generate code from compiled .NET assemblies.
-Download Xcode developper tools + iOS SDK. All SDKs are bundled within Xcode development environment and can be downloaded when having Apple ID You have here a simulator not an emulator (the code generated for a simulator is different from the code for target device. If emulator – the code is identical)
-Installing MonoTouch. It exists 2 versions: trial and licensed. In the trial is not included the software needed to generate an app to run on the physical device.
3. Preparing for Android development - Installation for both Windows and Mac OSX
-Tools are Java based
-Installing Java JDK:
-For Mac OSX JDK is included natively
-For Windows must be downloaded (32-bit version of Java JDK)
- Installing the Android SDK: must be downloaded – the suitable version (2.2; 3.x; 4.x)
- Configure the emulator – add a virtual device. Must be done explicitly!
4. Installing Eclipse - it's preferable to install 1.Eclipse development and 2. Android plug-ins
- install Eclipse Classic (free)
- Install Android plug-in
- Install Mono for Android
- Install MonoCross Project Templates for Visual Studio
- Install the MonoCross Templates for MonoDevelop

Installing MonoDevelop for Mac:
-iOS SDK is required and Mac OSX 10.6 or later OS
-Download the latest version of Mono framework.
-You have the Framework. Have to install MonoDevelop for mobile platforms (iOS and/or Android)
- After having installed development tools, install the device frameworks (SDK)

41. Модифициран крос-платформен Mono-модел на Model-View-Controller (MVC) приложения.

The Model-View-Controller (MVC) pattern separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes:



MonoCross MVC pattern includes separated Interfaces: decouples the platform-specific presentation views from the common code in the shared application.

Defining a simple model:

An example program within MVC model:

•You want to work with a customer in a management system Here is the Model object (business objects being acted in the app)

•file: project_name/Model/Customer.cs

Now you need to create the controller that will load the model and make some decisions about processing of the model for presentation in your views…

- Now that you've defined your model object, you need to create a controller that can load the model from your system-of-record and make any business decisions about processing of the model for presentation in your views.
- Controllers in a MonoCross application are inherited from the abstract MXController<T> class.
- The MXController<T> class implements the IMXController interface, which defines the contract for a MonoCross controller. The interface consists of the properties, methods, and events necessary for the MonoCross framework to manipulate the model and trigger the rendering of platform-specific views.
- Each controller class can handle all the business and data logic for a particular model type. You can create controllers at any level of your model hierarchy, but it is a best practice to create your controllers at the level that makes the most sense for the specific needs of your application and for the use cases described by your application's workflow requirements.

42. Създаване на платформен контейнер в Mono MVC архитектура: за конзола, за iOS, Android, Windows Phone, WebKit.

Adding a Platform Container:

- Inside the platform container you define views for every platform, so shared controllers render them as defined into the app workflow)
- First you have to inherit from abstract class MXContainer for each platform supported.
- You put there your application views:

```
public class MXConsoleContainer : MXContainer
{
public MXConsoleContainer(MXApplication theApp):base(theApp){}
…
// there is the place for initialization of the app; instantiating the controller (so views are registered)}
```

Having a container class, in the Main() you have to initialize it (or them) and to add views:

1)Create an instance of the shared app;

2)register the views by AddView(), passing instance of each view +ViewPerspective name:

For the example, you have 2 views:
first -displays a list of customers;
the second - displays the details of a single customer (instance of CustomerViews class):

```
class Program
{ static void Main(string[] args)
{ MXContainer.Initialize(new CustomerManagment.App());
MXConsoleContainer.AddView<Customer>(newViews.CustomerView(),
ViewPerspective.Default);
MXConsoleContainer.AddView<Customer>(newViews.CustomerEdit(),
ViewPerspective.Update);
…… }
```

Implementing an iOS platform container:
MonoTouch for iOS is like a slimmed-down version of .NET desktop + libraries of native iOS APIs that Apple provides to C/Objective-C developpers (Cocoa Touch). That's not an abstract cross-platform UI. Instead this provides binding capabilities to the native API of the device. There is a binding from MonoTouch to Cocoa Touch classes (only a wrapper).
-First of all you have to write a code to initialize the container (MXTouchContainer class) and AddView() for it.
-After you have to write code for different application's views (for example: ListView, Customer View, Customer Edit View); there are MXTouchView… base classes for the purpose.

Implementing an Android container:
1 . Initialize the Container. Android Application consist of 1 or more activities. Any of them can be the entry point. The developer must specify which activity is the entry point, there are no main.
By property you specify the main activity class and describe functionality for events (On..() methods).
Using MXDroidContainer.AddView<View_Name>(..) you add views.
Using MXDroidContainer.Navigate(..) you change views.
2. You have to describe All Views successively naming by property like this:
[Activity(Label = "Customert List" , Icon = "@darwable/icon")]
having activity name for the view, you are ready to create class and methods for it.
For example :
ListView,
CustomerView,
CustomerEditView
MonoCross provides base classes from which you can derive your views.

Implementing a Windows Phone Container:
Windows Phone is version of Silverlight. Every page needs its XAML file describing view layout, formatting, data binding. Dynamic views are not supported.
Execution flow is message driven. Starting point is in Application derived class (App).
1.Initializing.
You have to write a class App : Application {..}
Like in the previous into this class must be :
MXPhoneContainer.Initialize(..);
MXPhoneContainer.AddView<View_Name>(..); …
MXPhoneContainer.Navigate(..);
Class Methods must be described.
2. Binding Views:
Description is in XAML files + into a normal file with the class methods code for every view.

Implementing a WebKit container:
MonoCross include capability to create container implemented for MS ASP.NET MVC framework.
The WebKit container runs on the server and not on the client (unlike other platforms
like jQuery and jQTouch).
Views and controller render an HTML for every page sent down to the client
1.Initializing a Container with WebKit (occures during the Session Start and runs for each new client that connects to the web server). Like ASP.NET Global.asax.cs file is needed for each new Session code like this exists:
MXWebKitContainer.Initialize(…);
MXWebKitContainer.AddView<View_Name>(..);
2.Building Views with WebKit .html file (Root.html) exists for each interface. There are HTML, CSS, JavaScripts. MonoCross WebKit container provides 2 base classes from witch you derive you views: MXWebKitView<View_name> and MXWebKitDialogView<View_Name>.
In this classes (Methods exist for this) you define the content of the correspondent HTML. Example:
button.Controls.Add(image);
a.Controls.Add(em);

43. Крос-платформени, мобилни приложения с Cordova (PhoneGap). Философия, възможности. Архитектура на приложение.

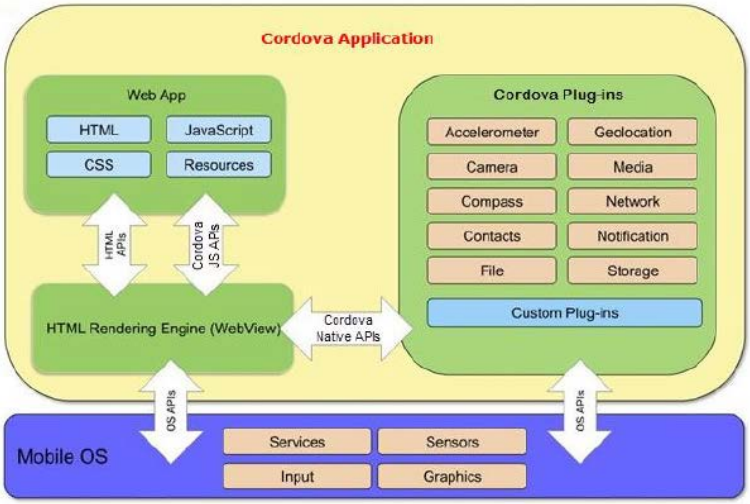
PhoneGap:
- Apache Cordova was originally called Phonegap build by Nitobi
- Open-source & free software from the beginning (MIT License), Apache License now
- Nitobi then aquired by Adobe and donated the PhoneGap codebase to the Apache Software Foundation (ASF)
- PhoneGap is still a product of Adobe. It is a distribution of Apache Cordova. Think of Apache Cordova as the engine that powers PhoneGap.
- PhoneGap is an open source framework for building cross-platform mobile applications with HTML, CSS, and JavaScript.
- This is an ideal solution for web developers interested in mobile development as it allows them to leverage existing skills rather than start form scratch with a device-specific compiled language.
- This is also an ideal solution for those interested in creating an application that can run on multiple devices with the same code base.

Platform Support:
IOS, Android, Windows Phone, BlackBerry, Palm webOS, Tizen+bada, Symbian.

Apache Cordova is a platform for building natively installed mobile applications using HTML, CSS and JavaScript.

Tools for development:
- Any HTML & JS editor
- Platform SDK e.g. Android SDT, Android SDK, BB SDK, Xcode, Visual Studio Mobile.
- Platform Emulator (usually provide along with SDK)
- JS/HTML GUI Mobile framework e.g. JQuery, Sencha Touch, dojo Mobile
- Browser e.g. Firefox with Bugzilla extension, Chrome Browser

Архитектура на приложение:



When building Cordova applications, you can employ two distinct patterns:
- Multipage applications: In multipage applications, multiple HTML pages are used to represent the various screens of your application. Navigation between pages uses the standard browser mechanics, with links defined by anchor tags. Each HTML page includes script references to the Cordova JavaScript code and your application JavaScript.
- Single-page applications: In single-page applications, a single HTML file references Cordova and your application JavaScript. Navigation between the various pages of your application is achieved by dynamically updating the rendered HTML. From the perspective of the phone browser, the URL remains the same and there's no navigation between pages.
- The multipage approach has some disadvantages: First, when the browser navigates from one page to the next, it has to reload and parse all the JavaScript associated with the new page. There's a noticeable pause as the Cordova lifecycle. Second, because your JavaScript code is being reloaded, all application state is lost.
- The single-page pattern overcomes the issues associated with the multipage approach. The Cordova and application JavaScript code is loaded just once, resulting in a more responsive UI and removing the need to pass application state from one page to the next.

44. Мобилни приложения с Cordova: среда и помощни средства. Структура на приложение. Процес на разработка.

Структура на приложение:
●The PhoneGap applications are hybrid
●They are neither truly native nor purely web based
●All layout rendering is done via the web view instead of Objective-C
●Much of the functions of HTML5 are supported
●Applications built with PhoneGap are not just like normal mobile web sites.
PhoneGap Applications
●PhoneGap applications are able to interact with mobile device hardware, such as the Accelerometer or GPS, in ways that are unavailable to normal web applications.
●PhoneGap applications are also built and packaged like native applications, meaning that they can be distributed through the Apple App Store or the Android Market.

User Interface:
- The user interface for Apache Cordova applications is created using HTML, CSS, and JavaScript.

- The UI layer is a web browser view that takes up 100% of the device width and 100% of the device height.
- The web view used by application is the same web view used by the native operating system

Apache Cordova API:
- Provides an application programming interface (API)
- enables you to access native operating system functionality using JavaScript.
- APIs for Accelerometer, Camera, Compass, Media, FileSystem, etc
- Extendable using native plug-in

Tools for development:
- Any HTML & JS editor
- Platform SDK e.g. Android SDT, Android SDK, BB SDK, Xcode, Visual Studio Mobile.
- Platform Emulator (usually provide along with SDK)
- JS/HTML GUI Mobile framework e.g. JQuery, Sencha Touch, dojo Mobile
- Browser e.g. Firefox with Bugzilla extension, Chrome Browser

Developing Cordova Applications:
- You can add your HTML, JavaScript and CSS files to the www folder and they'll be included in your project and accessible via the browser control when your application executes. You can use any of the standard JavaScript/HTML5 libraries or frameworks in your Cordova application, as long as they're compatible with the phone's browser.
- The Cordova APIs are documented on the Cordova Web site. One important thing to note is that you must wait for the device-ready event before making use of any of the other API methods. If you inspect the index.html file generated from the template, you can see that it waits until the device is ready before updating the UI.