**Курсов проект по дисциплината**

**”Технология на софтуерното производство”**

На тема

**Информационно табло на аеропорт**

Разработил

**Тодор Кирилов Пенчев**

Фак. № 20621541

Срок за представяне: май, 2023г.

Водещ преподавател:

/доц.В.Божикова/

ЗАДАНИЕ

**за курсов проект по дисциплината**

**”Технология на софтуерното производство”**

**Цел на курсовия проект:**

Изисква се да се разработи „Програмно обезпечение на информационно табло на аеропорт”.

Изисквания:

* Да изобразява състоянието на полетите
* Да се изобразява следната информация за полетите
  + Номер на полет
  + Пункт за излитане
  + Време на излитане / кацане
  + Статус
* Да обезпечи
  + Добавяне и отстраняване на информация за полети
  + Внасяне на изменния в състоянието на таблото, ако стане някакво събитие

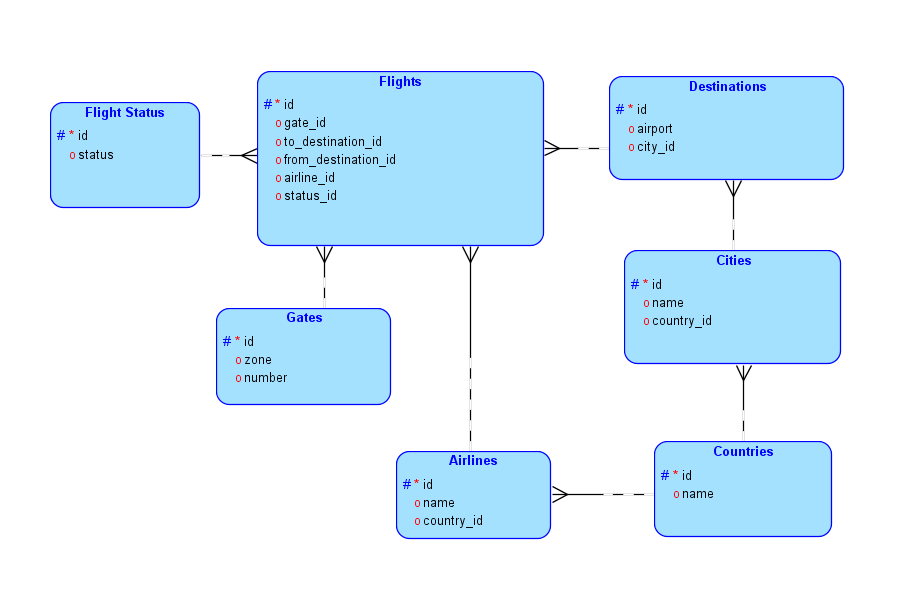
**Реализация на проекта**

**Проектиране**

Базата данни се състои от следните таблици:

* Airlines (Авиолинии)
  + Id – Уникален идентификатор в таблицата
  + Name – име на авиолинията
  + Country\_id – идентификатор на държавата, към която авиолинията принадлежи
* Cities (Градове)
  + Id – Уникален идентификатор в таблицата
  + Name – име на град
  + Country\_id - идентификатор на държавата, към която авиолинията принадлежи
* Countries (Държави)
  + Id – Уникален идентификатор в таблицата
  + Name – име на държавата
* Destinations (Дестинатции – комбинация от град и летище)
  + Id – Уникален идентификатор в таблицата
  + Airport – име на летище
  + City\_id – идентификатор на града, към която авиолинията принадлежи
* Flight Status (Състояния на полета)
  + Id – Уникален идентификатор в таблицата
  + Status – Статус на полета
* Gates (Пунктове за излитане)
  + Id – Уникален идентификатор в таблицата
  + Zone – зона на летището
  + Gate\_number – номер на
* Flights (Полети)
  + Id – Уникален идентификатор в таблицата
  + Gate\_id
  + To\_destination\_id
  + From\_destination\_id
  + Airline\_id
  + Status\_id

1. Логически модел на базата данни



1. Релационен модел на базата данни

A picture containing text, diagram, screenshot, font

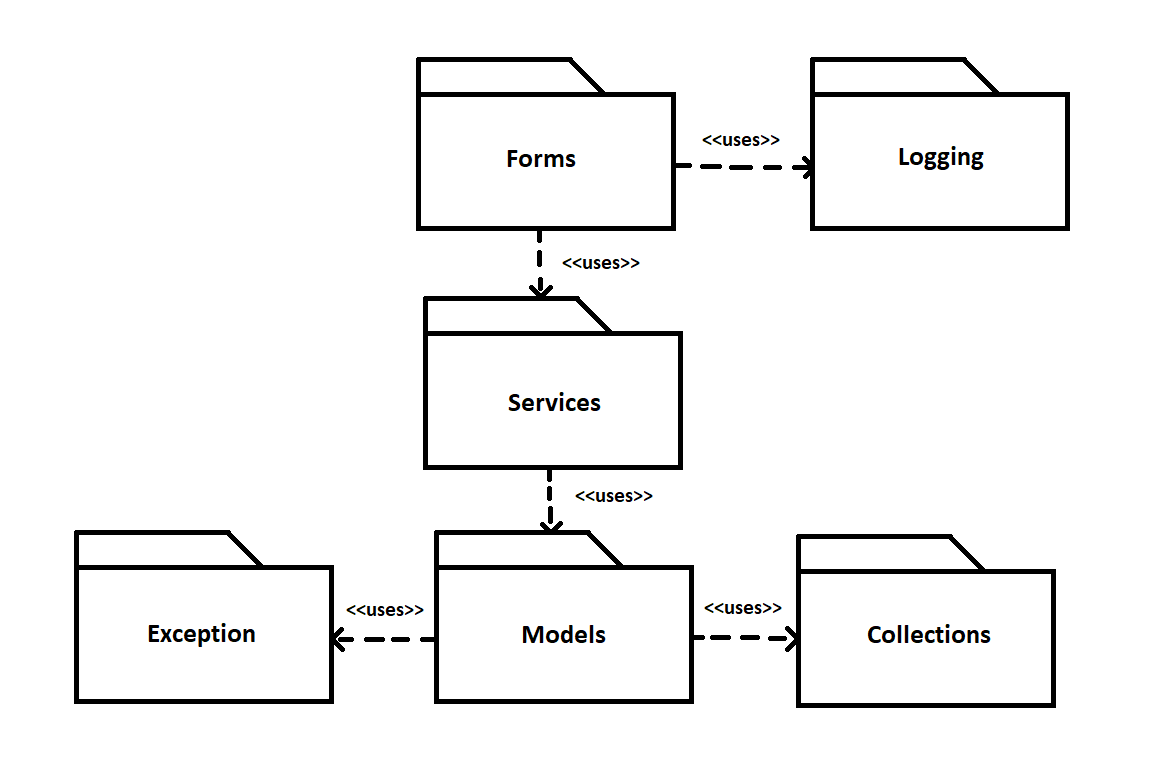
Description automatically generated

1. Модули на програмата:

* Collections – Отговаря за превръщане на идващите записи от базата данни в списъци (List), кото приложението да използва удобно
* Exceptions – отговаря за възможните изключения по време на изпълнение на програмата
* Forms – отговаря за интерфейса на приложението и всички събития, които могат да произтекат от неговото използване
* Logging – отговаря за показване на събитията, които могат да произтекат по време на изпълнение на програмта
* Models – отговаря за представяне на базата данни в кода на програмата и отговаря за SQL заявките
* Services – отговаря за бизнес логиката на приложението

1. UML диаграми

* Диаграма на модулите (Package Diagram)



**Програмиране**

1. Models - модул за връзка с базата данни
   1. **Model** – абстрактен клас, който имплементира общи функции за всички модели, които ще представят таблиците от базата данни в програмата. Като атрибути съдържа данните за връзка с access файлът и самата връзка.
      1. List<List<string>> **runSelect**(string query) – изпълнява SELECT заявки и обръща резултатите от OleDbDataReader в List<List<string>> чрез ползването на класа ListParser и неговия метод readerToList()
      2. Void **runQuery**(string query) – изпълнява всички заявки, при които не се очаква връщане на записи от базата данни, като INSERT, UPDATE, DELETE чрез метода oleDbRunQuery.
      3. Void **oleDbRunQuery**(string query) – Отваря връзката към базата данни, изпълнява заявката и затваря връзката
      4. List<List<string>> **getAll**() – връща всички записи за съответната таблица
      5. Void **delete**() – изтрива запис по посочено id
      6. String **getTableName**() – връща името на таблицата, която моделът представлява
      7. List<List<string>> getAllWithRelations() – изпълнява заявка, която връща всички резултати в дадена таблица, заедно с релационните полета от други таблици
      8. Void **save**() – абстрактен метод, който в наследниците ще създава INSERT заявките
      9. Void **update**() - абстрактен метод, който в наследниците ще създава UPDATE заявките
      10. String **getId**() – връща id-то на обекта, което се използва при заявките за UPDATE и DELETE
   2. Airline
      1. Наследява абстрактният клас Model
      2. Имплементира абстрактните методи по спечифичен начен за съответната таблица
      3. В конструтора се задава името на таблицата в полето на класа tableName
      4. Имплементира getters и setters
      5. Всички останали класове в съответния модул са аналогични на този, наследявайки класът Model
   3. City
   4. Destination
   5. Flight
   6. FlightStatus
   7. Gate
2. Collections - модул за превръщане обработка на записите върнати от SQL използваеми списъци в C#
   1. ListParser – Клас, който обръща записите от OleDbDataReader в списък от списъци
      1. List<List<string>> **readerToList**(OleDbDataReader reader) – единсвеният метод на класа, който може да бъде извикан статично. Той чете записите от reader и ги записва в списък, като всеки елемент на списъка е друг списък, който съдържа полетата на съответния запис от базата данни
3. Services – модул отговарящ за бизнес логиката на приложението
   1. ComboBoxTools – модул отговарящ за необходимостите на използваните в приложението ComboBox елементи
      1. **ComboBoxItem** – клас, който представлява създаден от нас елемент на ComboBox. Необходимо за разработването на приложението беше елементите на ComboBox да могат да имат както текстово представяне за потребителя, така и да съдържат скрита стойност в себе си. За това този клас има string полета Text и Value. Първата представлява стойноста, която ще се вижда за потребителя, а втората стойноста, която ще използва приложението.
      2. **SelectedItems** – Клас, който спомага задаването на избран елемент в ComboBox
         1. Void **set**(ComboBox, string selectedValue) – единствения метод на класа, който може да бъде използван статично. По подаден ComboBox и стойност, открива индексът на тази стойност и задава този индекс за SelectedIndex
   2. GridTools – Модъл отговарящ за работата с DataGridView
      1. Row – Клас с единствен статичен метод **isSelected**, който може да провери дали има избран елемент от посочен DataGridView
      2. Table – Клас, който помага при използването на DataGridView
         1. Void **loadData**(DataGridView grid, Model model) – Взима всички записи за съответния подаден Model и ги зарежда в подадения DataGridView използвайки методът loadRowsIntoGridView
         2. Void **loadRowsIntoGridView**(DataGridView grid, List<List<string>> rows) – изчиства данните с grid.Rows.Clear() И след това с цикъл създава нови редове в
   3. **Service** – абстрактен клас, който представя необходимите функции за работа с базата данни и операциите по INSERT, UPDATE и DELETE. Целта му е да създаде ново ниво на абстракция и да не работи с моделите директно във формите
      1. AirlineService – разширява класът Service и имплементира абстрактните методи
         1. Void **setFields**(Dictionary<string, string> values) – задава стойности на атрибутите на класа, които по-късно ще бъдат използвани за работа с моделите
         2. Void **save**() – създава обект от модел Airline и задава съответните му стойности (id, name, countryId), след което изпълнява save() методът на класът, който добавя нов запис таблицата чрез INSERT
         3. Void **update**() – аналогично на save() методът, но за операция UPDATE
         4. Void **delete**() – по задаено id, намира записът в базата данни и го изтрива чрез DELETE
         5. Всички останали класове в съответния модул са аналогични на този, наследявайки класът Service
      2. CityService
      3. CountryService
      4. DestinationService
      5. FlightService
      6. GateService
      7. StatusService
4. Forms
   1. AirlinesForm
      1. Атрибут **airlineService** – Dependancy Injection на услугата AirlineService
      2. Атрибут **values** – Dictionary, което ще съдържа стойностите на полетата, които се взимат от потребителския интерфейс и се подават при операции save() и update()
      3. **AirlinesForm\_Load**(object sender, EventArgs e) – зарежда необходимите данни във формата чрез методите arilinesLoadData() и loadCountries(). Също така задава handler за събитието RowStateChanged на използвания DataGridView
      4. **addAirline\_Click**(object sender, EventArgs e) – взима стойностите въведени от потребителя и ги подава на airlineService, след което и изпълнява операцията по запазване. Накрая обновява данните DataGridView
      5. **updateAirline\_Click**(object sender, EventArgs e) – аналогично, но в допълнение се подава и id-то на избрания ред от таблицата за да се изпълни методът update()
      6. void **getFieldValues**() – Взима стойностите от TextBox-овете или ComboBox-овете и ги задава във values
      7. **deleteAirline\_Click**(object sender, EventArgs e) – задейства операцията по изтриване на запис
      8. **airlineGridView\_RowStateChange**(object sender, EventArgs e) – Handler за избран ред от DataGridView. При задействането му се взимат стойности от таблицата и се задават в полетата за въвеждане
      9. Void **airlineLoadData**() – зарежда данните за таблицата използвайки Table.loadData(airlinesGridView, new Airline())
      10. Void **loadCountries**() – зарежда списъкът за държави, от който се избира при добавяне/обновяване на запис
   2. CitiesForm
   3. CountriesForm
   4. DestinationsForm
   5. GatesForm
   6. StatusesForm
5. Logging
   1. Notifications
6. Exceptions
   1. InsertException
   2. UpdateException
   3. DeleteException

**Ръководство за потребителя**

1. Начален екран

A screenshot of a computer

Description automatically generated with medium confidence

На началния екран се виждат две таблици. Първата съдържа пристигащите полети, а втората съдържа отлитащите. Информацията, която ми преставя е час на пристигане/отлитане, дестинация, от която пристига/за която заминава, номер на полет, пункт на летището и статус на полета. Това всъщност е таблото, което се изисква според заданието

В долния десен край се намира бутон за презареждане/актуализиране на информацията.

Бутоните Airlines, Cities, … служат за отваряне на формите, от които се оправляват съответните таблици с информация

1. Airlines Form

A screenshot of a computer

Description automatically generated with medium confidence

Изгледът в този екран е всъщност изгледът, който ще наблюдаваме и в другите форми. Голямата част от екрана е заета от таблица с информацията от базата данни. В десния край се виждат полета за въвеждане на данни в горния край и бутони за управление в долния край. Бутонът Add запазва информацията в таблица от базата данни. Бутонът Update обновява избраният ред от таблицата, а Delete изтрива съответния ред. За операциите Update и Delete трябва да се избере ред от таблицата.Операциите са аналогични за всички останали форми.