**The Red Tide**

**Documentul de Proiectare a Soluţiei Aplicaţiei Software**

**(Software Design Document)**

**29 Octombrie, 2017**

Echipă:

* PM - BOBOC Bianca Ioana, [bianca.ioana.boboc@gmail.com](mailto:bianca.ioana.boboc@gmail.com)
* TL - GHIDARCEA Anamaria, [anamaria.ghidarcea95@gmail.com](mailto:anamaria.ghidarcea95@gmail.com)
* DEV – TĂNASE Teodor Radu, [teoxyz@gmail.com](mailto:teoxyz@gmail.com)
* DEV - HUZUM-COMANICI Laura, [laurahuzumcomanici@gmail.com](mailto:laurahuzumcomanici@gmail.com)

**1. Scopul documentului**

Acest document are rolul de a descrie acurat şi complet aplicaţia proiectată – joc Tower Defense “Titlu”. Documentul serveşte drept ghid unic de construire a soluţiei pentru echipa de dezvoltare a proiectului.

**2. Conţinutul documentului**

Documentul este format din patru secţiuni esenţiale:

• Modelul datelor – prezintă principalele structuri de date folosite

• Modelul arhitectural

• Modelul interfeţei cu utilizatorul – prezintă interfaţa cu utilizatorul şi succesiunea ferestrelor acesteia

• Elemente de testare – prezintă componentele critice şi alternative de proiectare a acestora. (RAMANE SAU ??)

**3. Modelul datelor**

**3.1. Structuri de date globale**

Structura de date globală folosită este o instanţă a clasei Joc (clasa core a aplicaţiei). Cu ajutorul acestei structuri de date globale cu rol de intermediar, celelalte clase ale aplicaţiei îşi pot accesa reciproc datele interne.

Informaţiile globale despre joc, cum ar fi nivelul curent sau cantitatea de monede la momentul actual, sunt reprezentate sub forma de câmpuri a clasei core “Joc”.

Joc = Scena ? Avem ceva mai presus de Scena?

//TODO

**3.2. Structuri de date de legătură**

În mare parte, legăturile dintre diversele obicte sau module ce alcătuiesc aplicaţia se fac prin intermediul elementelor grafice. Deoarece limbajul folosit este de aşa natură (orientat pe elemente grafice), pentru a modifica comportamentul unui element, sau pentru a-i accesa proprietăţile şi metodele, este necesară o asociere/legătura cu elementul respectiv, stabilită cu ajutorul instrumentelor specifice puse la dispoziţie de mediul de dezovoltare folosit de noi (Unity).

**3.3. Structuri de date temporare**

//TODO Tipuri temporare sau obiecte temporare

**3.4. Formatul fişierelor utilizate**

Aplicaţia foloseşte un singur fişier pentru salvarea stării jocului – în momentul în care utilizatorul închide jocul, nivelul, wave-ul curent, elementele defensive plasate şi cantitatea de monede sunt salvate într-un fişier .txt. În momentul în care aplicaţia este redeschisă, jocul îşi reia starea, încărcând informaţiile din acest fişier.

**3.5. Descrierea bazei de date**

Deoarece aplicaţia noastră se bazează în mare parte pe date cu o evoluţie standard (în linii mari), nu are nevoie de o bază de date. Singurele date persistente de care avem nevoie se află în fişierul descris la punctul anterior. Nivelul, wave-ul curent, elementele defensive plasate şi cantitatea de monede sunt singurele informaţii din exterior, de care are nevoie aplicaţia în momentul pornirii. În rest, ea se bazează strict pe date şi resurse interne pentru a reface starea jocului şi pentru a continua evoluţia acestuia.

**4. Modelul arhitectural şi modelul componentelor**

**4.1. Arhitectura sistemului**

**4.1.1. Şabloane arhitecturale folosite**

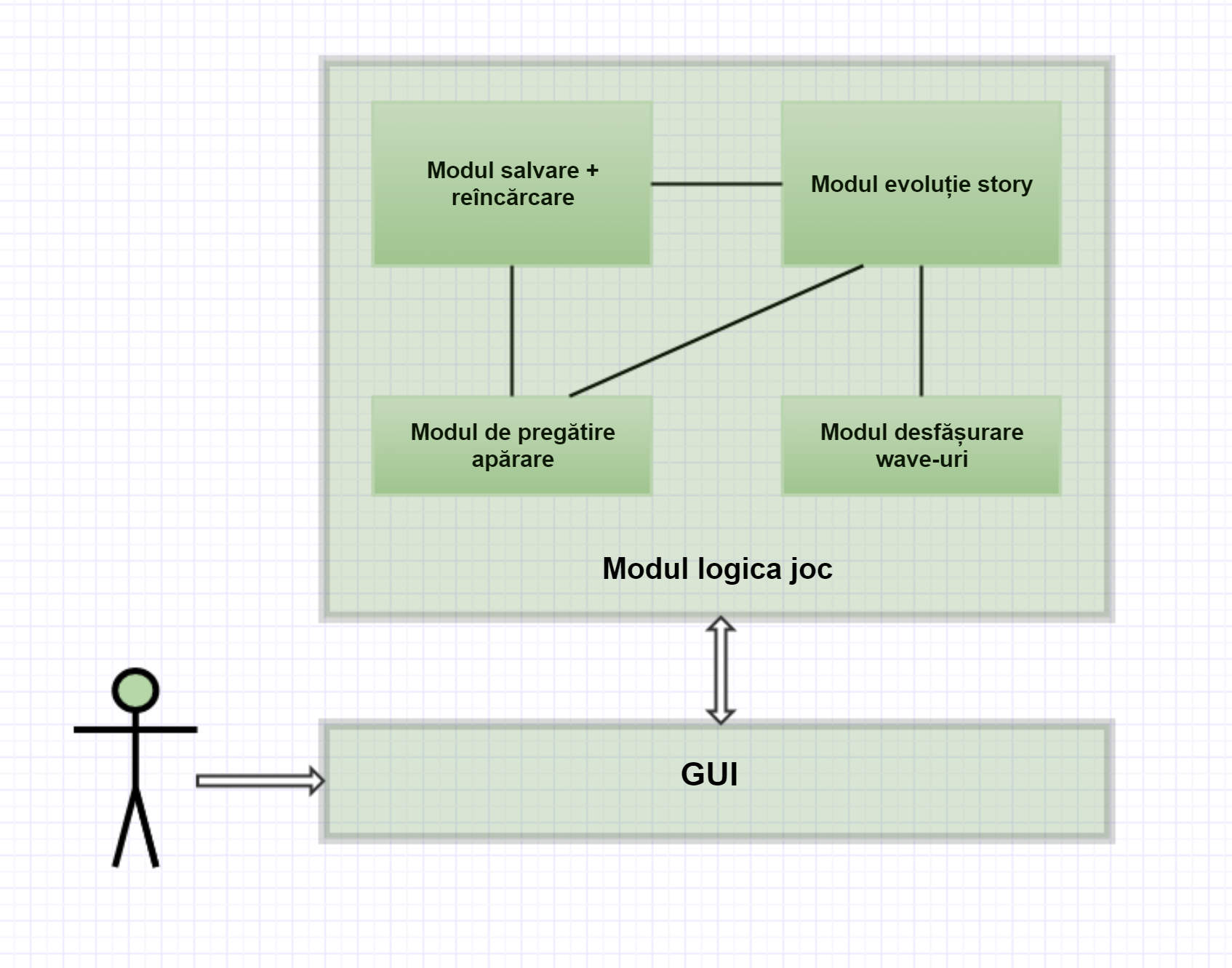
Având în vedere faptul ca aplicaţia noastră nu este distribuită şi nu are un repository complex, nu se aplică şabloanele de enterprise şi application.

Singurele şabloane despre care putem discuta sunt cele de software design:

* La şabloanele de proiectare folosite, putem menţiona doar *Prototype*.
* Folosim doua tipuri de şabloane structurale: *Decorator* (pentru extinderea funcţionalităţilor obiectelor principale) şi *Flyweight* (pentru a suporta eficient numărul mare de obiecte de acelaşi fel.
* Ca şabloane comportamentale folosim *Iterator* şi *Visitor* (pentru a accesa cu uşurinţă elementele jocului).

**4.1.2. Diagrama de arhitectură**

Diagrama de arhitectură de mai jos descrie modulele aplicaţiei şi relaţiile de interacţiune dintre acestea.



**4.2. Descrierea componentelor**

Aplicaţia constă din următoarele module interconectate:

* Modulul GUI (Graphical User Interface) - Este responsabil cu desenarea şi randarea optimă a interfeţelor grafice ale aplicaţiei.
* Modulul de Logică joc - Este responsabil de coordonarea celorlalte module copii astfel încât rezultatele rulării şi interacţiunea utilizatorului cu interfata grafică să aibă efectele dorite. Acest modul cuprinde 4 submodule (după cum se observă în schema arhitecturală) pe care le voi descrie în continuare.
* Modulul Salvare + Reîncărcare – Este responsabil de salvarea parametrilor cheie necesari pentru refacerea stării jocului, în momentul închiderii. De asemenea, este responsabil de refacerea stării jocului cu ajutorul acestor parametri în momentul redeschiderii jocului.
* Modulul desfăşurării wave-urilor – Se ocupă efectiv de valurile de inamici şi de interacţiunea lor cu mediul înconjurător. Acest modul nu implică contribuţia utilizatorului, decât în cazul intervenţiei armei dinamice.
* Modulul de pregătire apărare – Acest modul implică în totalitate comenzi de la utizator, care are posibilitatea să pregătească harta pentru urmatorul val de inamici. În cadrul acestui modul poate cumpăra arme noi (statice sau dinamice) sau poate upgrada arme vechi.
* Modulul de evoluţie story – Acest modul se ocupă de consecinţele rezultatatelor valurilor asupra poveştii jocului. Adică, în funcţie de cum rezistă obiectivul la valuri se avansează sau nu la nivelul următor, sau se declară sau nu jocul pierdut.

**4.3. Restricţiile de implementare**

Având in vedere faptul că aplicaţia noastră este monolitică, toate modulele aplicaţiei vor fi dezvoltate utilizând limbajul de programare Unity.