# BLStream Fingerprint

# Table of Contents

# 1. Preface

BLStream Finger print is a set of practices applied in the company.

# 2. Project management

## 2.1. Definition of Done

## 2.2. Daily Standup

## 2.3. Demo

## 2.4. Scrum/Kanban Board

## 2.5. Planning Meeting

## 2.6. Retrospectives

## 2.7. Retrospectives shared with the customer

## 2.8. Project Practices Charter shared with the customer

## 2.9. Collocated Team (all team members PM included)

## 2.10. 3rd party libraries licences listed, approved

## 2.11. Clean Backlog

## 2.12. Responsive Product Owner

## 2.13. Grooming

## 2.14. BurnUp / BurnDown chart

# 3. Development

## 3.1. Easy infrastructure setup

from nothing to running in <1h

## 3.2. Easy application setup

from nothing to running in <1h

## 3.3. Concurrency in application code accounted for

## 3.4. GUI Style Guide defined

## 3.5. Application Monitoring

## 3.6. Unit Tests

Unit testing is at the core of engineering practices in BLStream. It's not just a practice, it is a foundation on which many other more sophisticated practices are built. It would not only be impossible to apply techniques like Continuous Integration and many other kinds of automation, but also it would make code refactorings and generally code maintenance much harder and error prone.

Unit testing

## 3.7. Scalability requirements known and accounted for

## 3.8. Performance requirements known and accounted for

## 3.9. Static code analysis (backend)

## 3.10. Application events logging

## 3.11. OWASP Top 10 in Definition of Done

### 3.12. Authorization model defined

### 3.13. Continuous Integration

### 3.14. Continuous Delivery

### 3.15. Continuous Deployment

### 3.16. Documentation tracked in VCS

### 3.17. Documentation generated during CI

### 3.18. Parts of the documentation generated automatically

### 3.19. Automatic documentation of the executed tests

### 3.20. Documentation scope agreed

### 3.21. JS application framework

### 3.22. JS Build process

### 3.23. JS modules dependency management

### 3.24. JS Unit test

### 3.25. CSS builder

### 3.26. Static code analysis (JavaScript)

### 3.27. Truly RESTful interfaces

### 3.28. HTML validation

### 3.29. Code Reviews

### 3.30. Pair Programming

### 3.31. Test Driven Development

### 3.32. Database schema versioning

### 3.33. Database data versioning

### 3.34. Concurrency for DB writes

### 3.35. Version Control System

### 3.36. Branching strategy

# 4. Quality assurance

## 4.1. Radiator

## 4.2. Defect Tracking System

## 4.3. Defined bug lifecycle

## 4.4. At least 1 QA for every 4 developers

## 4.5. Bug report template

## 4.6. Bug triage meeting

## 4.7. Smoke

## 4.8. Integration

## 4.9. Functional / Acceptance

## 4.10. Spelling

## 4.11. Security

## 4.12. Performance

## 4.13. Exploratory

## 4.14. Usability

## 4.15. Versioned repository of the test scenarios

## 4.16. Pair testing